Theoretical Computer Science 412 (2011) 83–96

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Complexity of multi-head finite automata: Origins and directions

Markus Holzer, Martin Kutrib *, Andreas Malcher

Institut für Informatik, Universität Giessen, Arndtstraße 2, 35392 Giessen, Germany

ARTICLE INFO

Keywords:
Multi-head finite automata
Descriptional complexity
Computational complexity

ABSTRACT

Multi-head finite automata were introduced and first investigated by Rabin and Scott in 1964 and Rosenberg in 1966. Since that time, a vast literature on computational and descriptional complexity issues on multi-head finite automata documenting the importance of these devices has been developed. Although multi-head finite automata are a simple concept, their computational behavior can be already very complex and leads to undecidable or even non-semi-decidable problems on these devices such as, for example, emptiness, finiteness, universality, equivalence, etc. Additionally the conversions between different types of multi-head finite automata induce in most cases size bounds that cannot be bounded by any recursive function, so-called non-recursive trade-offs. These strong negative results trigger the study of subclasses and alternative characterizations of multi-head finite automata for a better understanding of the nature of non-recursive trade-offs and, thus, the borderline between decidable and undecidable problems. In the present paper, we tour a fragment of this literature.

1. Introduction

Languages accepted by multi-tape or multi-head finite automata were introduced in [50,52]. Since that time, many restrictions and generalizations of the original models were investigated and studied (see, for example, [63]). Obviously, one-head automata, regardless of whether they work one- or two-way, or whether they are deterministic or nondeterministic are equivalent to right-linear context-free grammars and, thus, capturing the lowest level of the Chomsky hierarchy, the family of regular languages. On the other hand, two-way deterministic and nondeterministic *multi-head* finite automata are probably best known to characterize the complexity classes of deterministic and nondeterministic logarithmic space. In fact, in [14] it was shown that the question of the equality of deterministic and nondeterministic logarithmic space is equivalent to the question of whether every language accepted by some nondeterministic two-way three-head finite automaton (the corresponding language class is denoted by $\mathscr{L}(2\mathrm{NFA}(3))$) is accepted by some deterministic two-way multi-head finite automaton—here $\mathscr{L}(2\mathrm{DFA}(k))$ refers to the class of languages accepted by two-way k-head deterministic finite automata:

$$\mathsf{DSPACE}(\log n) = \mathsf{NSPACE}(\log n) \quad \text{if and only if} \quad \mathscr{L}(2\mathrm{NFA}(3)) \subseteq \bigcup_{k\geq 1} \mathscr{L}(2\mathrm{DFA}(k)).$$

Later this result was improved in [60] by showing that the relation remains valid also for one-way two-head nondeterministic finite automata instead of two-way three-head ones. Observe, that both one- and two-way multi-head finite automata induce a strict hierarchy of language families with respect to the number of heads, regardless of whether they work deterministically or nondeterministically [40,64].

* Corresponding author. Tel.: +49 641 99 32144.
E-mail addresses: holzer@informatik.uni-giessen.de (M. Holzer), kutrib@informatik.uni-giessen.de (M. Kutrib), malcher@informatik.uni-giessen.de (A. Malcher).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.08.024

Although multi-head finite automata are very simple devices, their computational behavior is already highly complex. But what about the size of multi-head finite automata opposed to their computational power? Questions on the economics of description size were already investigated in the early days of theoretical computer science and build a cornerstone of descriptional complexity theory [39,49,58]. During the past decade descriptional complexity has gained some new interest in the theoretical computer science community and became a widespread area. In terms of descriptional complexity, a known upper bound for the trade-off between different descriptional systems answers the question, how succinctly a language can be represented by a descriptor of one descriptional system compared to an equivalent description of another descriptional system. An example of this kind is the well-known result that one-way nondeterministic finite automata can offer exponential state savings compared to one-way deterministic finite automata, that is, given an n-state one-way nondeterministic finite automaton, then 2^n states are sufficient, and necessary in the worst case for a one-way deterministic finite automaton to accept the same language [39,41,49]. On the other hand, when dealing with more complicated devices such as, for example, pushdown automata that accept regular languages, a qualitative phenomenon is revealed, namely that of non-recursive trade-offs. There the gain in economy of description can be arbitrary, that is, there are no recursive functions serving as upper bounds for the trade-off. This phenomenon was first discovered in [39], and in the meantime, a lot of deep and interesting results on non-recursive trade-offs have been found for powerful enough computational devices almost everywhere (see, for example, [12,15,16,32,33,54,62]).

As previously mentioned *one-head* finite automata characterize the family of regular languages. Since this characterization is constructive, almost all elementary questions such as, for example, emptiness, finiteness, infiniteness, universality, inclusion, equivalence, etc., are decidable for one- and two-way finite automata, regardless of whether they work deterministically or nondeterministically. Mostly, precise exponential bounds (in terms of number of states) are known for simulations of one-head finite automata by each other [19,29,39,41,49,56]. Probably, the most important open question for one-head automata is to determine how many states are necessary and sufficient to simulate 2NFA(1) with 2DFA(1). This problem has been raised by Sakoda and Sipser in [53]. They conjectured that the upper bound is exponential. The best lower bound currently known is $\Omega(n^2/\log n)$. It was proved in [1], where also an interesting connection with the open problem whether the computational complexity classes $\mathsf{L} = \mathsf{DSPACE}(\log n)$ and $\mathsf{NL} = \mathsf{NSPACE}(\log n)$ are equal is given. However, not only are the exact bounds of that problem unknown, but we cannot even confirm the conjecture that they are exponential. The picture was complemented by the sophisticated studies on unary languages. The problem of Sakoda and Sipser has been solved for the unary case in [8]. Further partial solutions are obtained in [10,21,35,57].

On the other hand, when dealing with multi-head finite automata in general, this means, automata with *strictly* more than one head, then the phenomenon of non-recursive trade-offs comes into play. In [33] it is shown that even the simulation of 1NFA(2) by 1DFA(k), for $k \geq 2$, causes non-recursive trade-offs. Similar results hold for two-way devices as well [28]. An immediate consequence from the proofs of these results is that almost all of the aforementioned elementary questions become undecidable—in fact they are shown to be not even semi-decidable. Furthermore, because of these non-recursive trade-offs pumping lemmas and minimization algorithms for the automata in question do not exist. The main ingredient to obtain the results on non-recursiveness is a proof scheme, which was developed in [15,16]—for a unified form of this technique we refer to [32,33].

These strong negative results trigger the study of subclasses of multi-head finite automata for a better understanding of the nature of non-recursive trade-offs and, thus, the borderline between decidable and undecidable problems. From the legion of possible research directions we focus on three alternative and intermediate computational models, namely (1) multi-head automata accepting bounded languages, that is, languages of the form $a_1^* a_2^* \dots a_n^*$, for distinct letters $a_1, a_2, \dots, a_n$, (2) data-independent or oblivious multi-head finite automata, and (3) parallel communicating finite automata. Here, data-independent multi-head finite automata are machines, where the positions of the input heads depend on the length of the input only, and parallel communicating automata are finite state devices which work together on a common input tape according to a given communication protocol. While the former research on bounded languages dates back to [11], the latter two topics were recently investigated in [3,17,18]. In fact, for some of these models, some of the aforementioned elementary questions turn out to become decidable, while for others undecidability remains. There the border between decidability and undecidability turns out to be very thin and can be boiled down to a parameter related to the underlying device such as, for example, the number of head reversals. At this point it is worth mentioning that recently it was shown that even stateless one-way multi-head finite automata have a non-decidable emptiness problem [24]. In fact, these devices are the most simple ones, since they have one internal state only.

In the present paper we tour a fragment of the literature on computational and descriptional complexity issues of multi-head finite automata—it obviously lacks completeness, as one falls short of exhausting the large selection of multi-head finite automata related problems considered in the literature. We give our view of what constitute the most recent interesting links to the problem areas considered.

2. Multi-head finite automata

We denote the set of non-negative integers by $\mathbb{N}$. We write Σ^* for the set of all words over the finite alphabet Σ. The empty word is denoted by λ, and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. The reversal of a word w is denoted by w^R and for the length of w we write $|w|$. We use $\subseteq$ for inclusions and $\subset$ for strict inclusions. We write 2^S for the powerset of a set S.

Let $k \geq 1$ be a natural number. A two-way k-head finite automaton is a finite automaton having a single read-only input tape whose inscription is the input word between two endmarkers. The k heads of the automaton can move freely on the tape but not beyond the endmarkers. A formal definition is:

Definition 1. A *nondeterministic two-way k-head finite automaton* (2NFA(k)) is a system $M = \langle S, A, k, \delta, \triangleright, \triangleleft, s_0, F\rangle$, where

1. S is the finite set of *internal states*,
2. A is the set of *input symbols*,
3. $k \geq 1$ is the *number of heads*,
4. $\triangleright \notin A$ and $\triangleleft \notin A$ are the *left and right endmarkers*, respectively,
5. $s_0 \in S$ is the *initial state*,
6. $F \subseteq S$ is the set of *accepting states*, and
7. δ is the partial transition function mapping $S \times (A \cup \{\triangleright, \triangleleft\})^k$ into the subsets of $S \times \{-1, 0, 1\}^k$, where 1 means to move the head one square to the right, -1 means to move it one square to the left, and 0 means to keep the head on the current square. Whenever $(s', (d_1, d_2, \ldots, d_k)) \in \delta(s, (a_1, a_2, \ldots, a_k))$ is defined, then $d_i \in \{0, 1\}$ if $a_i = \triangleright$, and $d_i \in \{-1, 0\}$ if $a_i = \triangleleft$, for $1 \leq i \leq k$.

A 2NFA(k) starts with all of its heads on the first square of the tape. It halts when the transition function is not defined for the current situation. Nevertheless, if necessary, by adding some new states we always can modify a given 2NFA(k) such that it halts in distinguished states with all heads on the right endmarker.

A *configuration* of a 2NFA(k) $M = \langle S, A, k, \delta, \triangleright, \triangleleft, s_0, F\rangle$ at some time $t \geq 0$ is a triple $c_t = (w, s, p)$, where $w \in A^*$ is the input, $s \in S$ is the current state, and $p = (p_1, p_2, \ldots, p_k) \in \{0, 1, \ldots, |w| + 1\}^k$ gives the current head positions. If a position p_i is 0, then head i is scanning the symbol $\triangleright$, if it satisfies $1 \leq p_i \leq n$, then the head is scanning the p_ith letter of w, and if it is $n + 1$, then the head is scanning the symbol $\triangleleft$. The *initial configuration* for input w is set to $(w, s_0, (1, \ldots, 1))$. During its course of computation, M runs through a sequence of configurations. One step from a configuration to its successor configuration is denoted by $\vdash$. Let $w = a_1 a_2 \ldots a_n$ be the input, $a_0 = \triangleright$, and $a_{n+1} = \triangleleft$, then we set

$$(w, s, (p_1, p_2, \ldots, p_k)) \vdash (w, s', (p_1 + d_1, p_2 + d_2, \ldots, p_k + d_k))$$

if and only if $(s', (d_1, d_2, \ldots, d_k)) \in \delta(s, (a_{p_1}, a_{p_2}, \ldots, a_{p_k}))$. As usual we define the reflexive, transitive closure of $\vdash$ by $\vdash^*$. Note, that due to the restriction of the transition function, the heads cannot move beyond the endmarkers.

The language accepted by a 2NFA(k) is precisely the set of words w such that there is some computation beginning with $\triangleright w \triangleleft$ on the input tape and ending with the 2NFA(k) halting in an accepting state:

$$L(M) = \{w \in A^* \mid (w, s_0, (1, \ldots, 1)) \vdash^* (w, s, (p_1, p_2, \ldots, p_k)), s \in F, \text{ and } M \text{ halts in } (w, s, (p_1, p_2, \ldots, p_k))\}.$$

If in any case δ is either undefined or a singleton, then the k-head finite automaton is said to be *deterministic*. Deterministic two-way k-head finite automata are denoted by 2DFA(k). In case the heads never move to the left, the k-head finite automaton is said to be *one-way*. Nondeterministic and deterministic one-way k-head finite automata are denoted by 1NFA(k) and 1DFA(k), respectively. The family of all languages accepted by a device of some type X is denoted by $\mathscr{L}(X)$, where $X \in \{1\text{DFA}(k), 1\text{NFA}(k), 2\text{DFA}(k), 2\text{NFA}(k)\}$.

The power of multi-head finite automata is well studied in the literature. Obviously, for one-head machines, regardless of whether they work one- or two-way, or whether they are deterministic or nondeterministic, we obtain a characterization of the well-known family of regular languages REG. A natural question is to what extent the computational power depends on the number of heads. For (one-way) automata the proper inclusion $\mathscr{L}(1\text{NFA}(1)) \subset \mathscr{L}(1\text{DFA}(2))$ is evident. An early result is the inclusion which separates the next level, that is, $\mathscr{L}(1\text{DFA}(2)) \subset \mathscr{L}(1\text{DFA}(3))$ [25]. The breakthrough occurred in [64], where it was shown that the language

$$L_n = \{w_1\$w_2\$\ldots\$w_{2n} \mid w_i \in \{a, b\}^* \text{ and } w_i = w_{2n+1-i}, \text{ for } 1 \leq i \leq n\}$$

can be accepted by a 1DFA(k) if and only if $n \leq \binom{k}{2}$. Thus, L_n can be used to separate the computational power of automata with $k + 1$ heads from those with k heads in the one-way setting:

Theorem 2. *Let $k \geq 1$. Then*

1. $\mathscr{L}(1DFA(k)) \subset \mathscr{L}(1DFA(k + 1))$ *and*
2. $\mathscr{L}(1NFA(k)) \subset \mathscr{L}(1NFA(k + 1))$.

By exploiting the same language the computational power of nondeterministic classes can be separated from the power of deterministic classes [64]. To this end, for any n, the complement of L_n was shown to be accepted by some one-way two-head nondeterministic finite automaton. Since the deterministic language families $\mathscr{L}(1\text{DFA}(k))$ are closed under complementation, none of them includes the complement of L_n, if $n > \binom{k}{2}$. In particular, the inclusions $\mathscr{L}(1\text{DFA}(k)) \subset \mathscr{L}(1\text{NFA}(k))$ are proper, for all $k \geq 2$. Moreover, it can be shown that there is a witness language that is independent of the number of heads. This implies $\mathscr{L}(1\text{NFA}(2)) \setminus \bigcup_{k\geq 1} \mathscr{L}(1\text{DFA}(k)) \neq \emptyset$.

In order to compare one- and two-way multi-head finite automata classes, let $L = \{w \mid w \in \{a, b\}^* \text{ and } w = w^R\}$ be the mirror language. It is well known that the mirror language is not accepted by any 1NFA(k), but its complement belongs to $\mathscr{L}(1\text{NFA}(2))$. The next corollary summarizes the inclusions.

Corollary 3. *Let $k \geq 2$. Then*

1. $\mathscr{L}(1DFA(k)) \subset \mathscr{L}(2DFA(k))$,
2. $\mathscr{L}(1DFA(k)) \subset \mathscr{L}(1NFA(k))$, *and*
3. $\mathscr{L}(1NFA(k)) \subset \mathscr{L}(2NFA(k))$.

From the complexity point of view, the two-way case is the more interesting one, since there is a strong relation to the computational complexity classes $\mathsf{L} = \mathsf{DSPACE}(\log n)$ and $\mathsf{NL} = \mathsf{NSPACE}(\log n)$. In fact, in [14] the following characterizations have been shown.

Theorem 4. $\mathsf{L} = \bigcup_{k\geq 1} \mathscr{L}(2DFA(k))$ *and* $\mathsf{NL} = \bigcup_{k\geq 1} \mathscr{L}(2NFA(k))$.

Concerning a head hierarchy of *two-way* multi-head finite automata, in [40] it was proven that $k+1$ heads are better than k. More precisely, for each $k \geq 1$, there is a language accepted by some deterministic (nondeterministic) finite automaton with $k+1$ heads which is not accepted by any k-head deterministic (nondeterministic) finite automaton. Moreover, the witness languages are over an alphabet with a sole letter only, they are *unary*.

Theorem 5. *Let $k \geq 1$. Then there are unary languages that show the inclusions $\mathscr{L}(2DFA(k)) \subset \mathscr{L}(2DFA(k+1))$ and $\mathscr{L}(2NFA(k)) \subset \mathscr{L}(2NFA(k+1))$.*

Whether nondeterminism is better than determinism in the two-way setting, that is, whether the inclusion $\mathscr{L}(2\mathrm{DFA}(k)) \subseteq \mathscr{L}(2\mathrm{NFA}(k))$ is proper for $k \geq 2$ is an open problem. In fact, in [60] it was shown that the equality for at least one $k \geq 2$ implies $\mathsf{L} = \mathsf{NL}$. More generally, in [60] the following statement was proven:

$$\mathsf{L} = \mathsf{NL} \quad \text{if and only if} \quad \mathscr{L}(1\mathrm{NFA}(2)) \subseteq \bigcup_{k\geq 1} \mathscr{L}(2\mathrm{DFA}(k)).$$

Due to a wide range of relations between several types of finite automata with different resources, the results and open problems for k-head finite automata apply in a similar way for other types. Here, we mention deterministic two-way finite automata with k pebbles (2DPA(k)), with k linearly bounded counters (2DBCA(k)), and with k linearly bounded counters with full-test (2DBCFA(k)). In order to adapt the results, we present the hierarchy

$$\mathscr{L}(2\mathrm{DFA}(k)) \subseteq \mathscr{L}(2\mathrm{DPA}(k)) \subseteq \mathscr{L}(2\mathrm{DBCA}(k)) \subseteq \mathscr{L}(2\mathrm{DBCFA}(k)) \subseteq \mathscr{L}(2\mathrm{DFA}(k+1))$$

which has been shown in several papers, for example, [42,46,51,61]. An immediate consequence is that all of the aforementioned automata models characterize deterministic logspace:

$$\bigcup_{k\geq 1} \mathscr{L}(2\mathrm{DFA}(k)) = \bigcup_{k\geq 1} \mathscr{L}(2\mathrm{DPA}(k)) = \bigcup_{k\geq 1} \mathscr{L}(2\mathrm{DBCA}(k)) = \bigcup_{k\geq 1} \mathscr{L}(2\mathrm{DBCFA}(k)) = \mathsf{L}.$$

It is worth mentioning that similar inclusions and characterizations apply for the nondeterministic variants of the aforementioned automata models.

3. Descriptional complexity

Formal languages may have many representations in the world of automata, grammars and other rewriting systems, language equations, logical formulas etc. So, it is natural to investigate the succinctness of their representation by different models. The regular languages are one of the first and most intensely studied language families. It is well known that nondeterministic finite automata (1NFA(1)) can offer exponential savings in size compared with deterministic finite automata (1DFA(1)). Concerning the number of states, $2^n - 1$ is a tight bound for this type of conversion [39]. Concerning the simulation of a 2DFA(1) by a 1DFA(1) an upper bound of $(n+1)^{n+1}$ was shown in [56]. Moreover, the proof implied that any n-state 2NFA(1) can be simulated by a 1NFA(1) with at most $n2^{n^2}$ states. The well-known proof of the equivalence of two-way and one-way finite automata *via* crossing sequences reveals a bound of $O(2^{2n\log n})$ states [19]. Recently, in [29] it was noted that a straightforward elaboration on [56] shows that the cost can be brought down to even $n(n+1)^n$. However, this bound still wastes exponentially many states, since it is proven in [2] *via* an argument based on length-preserving homomorphisms that $8^n + 2$ states suffice. Recently, the problem was solved in [29] by establishing a tight bound of $\binom{2n}{n+1}$ for the simulation of two-way deterministic as well as nondeterministic finite automata by one-way nondeterministic finite automata. In the same paper tight bounds of $n(n^n - (n-1)^n)$ and $\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\binom{n}{i}\binom{n}{j}(2^i-1)^j$ are shown for two-way deterministic and two-way nondeterministic finite automata simulations by one-way deterministic finite automata. Nevertheless, some challenging problems of finite automata are still open. An important example is the question of how many states are necessary and sufficient to simulate 2NFA(1) with 2DFA(1). The problem has been raised in [53]. We refer to [8,10,21,35,53,57] for partial solutions and further reading.

All trade-offs mentioned so far are bounded by recursive functions. But, for example, there is no recursive function which bounds the savings in descriptional complexity between deterministic and unambiguous pushdown automata [62]. In [54] it is proved that the trade-off between unambiguous and nondeterministic pushdown automata is also non-recursive. A survey of the phenomenon of non-recursive trade-offs is [33]. Here we ask for the descriptional complexity of k-head finite

automata. How succinctly can a language be presented by a k-head finite automaton compared with the presentation by a nondeterministic pushdown automaton? How succinctly can it be presented by a log-space bounded Turing machine, or by a $(k+1)$-head finite automaton? It will turn out that there are no recursive functions serving as upper bounds. Thus, the trade-offs are non-recursive.

We first need some notation for descriptional complexity. For a general approach, we have to explain how to formalize the intuitive notion of a representation or description of a family of languages. A *descriptional system* is a collection of encodings of items where each item D *represents* or *describes* a formal language $L(D)$. For example, finite automata can be encoded over some fixed alphabet. The set of these encoding strings is a descriptional system that describes the regular languages. A natural property of descriptional systems is that their items are finite. In the following, we call the items *descriptors*, and identify the encodings of some language representation with the representation itself. It almost suggested itself that the underlying alphabet $\mathrm{alph}(D)$ over which D represents a language can be read off from the descriptor D.

Definition 6. A *descriptional system* E is a set of finite descriptors, such that each descriptor $D \in E$ describes a formal language $L(D)$, and the underlying alphabet $\mathrm{alph}(D)$ over which D represents a language can be read off from D. The *family of languages represented (or described)* by E is $\mathscr{L}(E) = \{L(D) \mid D \in E\}$. For every language L, the set $E(L) = \{D \in E \mid L(D) = L\}$ is the set of its descriptors in E.

Now we turn to measure the size of descriptors. Basically, a *complexity measure* for a descriptional system E is a total, recursive mapping $c : E \to \mathbb{N}$. From the viewpoint that a descriptional system is a collection of encoding strings, the length of the strings is a natural measure for the size. We denote it by length. In fact, we will use it to obtain a rough classification of different complexity measures. We distinguish between measures that (with respect to the size of the underlying alphabet) are recursively related with length and measures that are not.

Definition 7. Let E be a descriptional system with complexity measure c. If there is a recursive function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $\mathsf{length}(D) \leq g(c(D), |\mathrm{alph}(D)|)$, for all $D \in E$, then c is said to be an *s-measure*. The s-measure c is an *sn-measure*, if for any alphabet Σ, the set of descriptors in E describing languages over Σ is recursively enumerable in the order of increasing size.

Whenever we consider the relative succinctness of two descriptional systems E_1 and E_2, we assume the intersection $\mathscr{L}(E_1) \cap \mathscr{L}(E_2)$ to be not empty.

Definition 8. Let E_1 be a descriptional system with complexity measure c_1, and E_2 be a descriptional system with complexity measure c_2. A total function $f : \mathbb{N} \to \mathbb{N}$ is said to be an *upper bound* for the increase in complexity when changing from a descriptor in E_1 to an equivalent descriptor in E_2, if for all $D_1 \in E_1$ with $L(D_1) \in \mathscr{L}(E_2)$ there exists a $D_2 \in E_2(L(D_1))$ such that $c_2(D_2) \leq f(c_1(D_1))$. If there is no recursive upper bound, the *trade-off is said to be non-recursive*.

In connection with multi-head finite automata the question of determining the trade-offs between the levels of the head hierarchies arises immediately. Recently, the problem whether these trade-offs are non-recursive has been solved for two-way devices in the affirmative [28] (cf. also [32]):

Theorem 9. *Let $k \geq 1$. Then the trade-off between deterministic (nondeterministic) two-way $(k+1)$-head finite automata and deterministic (nondeterministic) two-way k-head finite automata is non-recursive. Moreover, the results hold also for automata accepting unary languages.*

But how to prove non-recursive trade-offs? Roughly speaking, most of the proofs appearing in the literature basically rely on one of two different schemes. One of these techniques is due to Hartmanis [15]. In [16] he developed a generalization and proved that, if the set of descriptors of one of the descriptional systems that have no equivalent descriptor in the other descriptional system is not recursively enumerable, then the trade-off between the systems is non-recursive. Next, we present a slightly generalized and unified form of this technique [32,33].

Theorem 10. *Let E_1 and E_2 be two descriptional systems for recursive languages such that any descriptor D in E_1 and E_2 can effectively be converted into a Turing machine that decides $L(D)$, and let c_1 be a* measure *for E_1 and c_2 be an* sn-measure *for E_2. If there exists a descriptional system E_3 and a property P that is not semi-decidable for descriptors from E_3, such that, given an arbitrary $D_3 \in E_3$,* (i) *there exists an effective procedure to construct a descriptor D_1 in E_1,* (ii) *D_1 has an equivalent descriptor in E_2 if and only if D_3 does not have property P, then the trade-off between S_1 and S_2 is non-recursive.*

In order to apply Theorem 10 one needs a descriptional system E_3 with appropriate problems that are not even semi-decidable. An immediate descriptional system is the set of Turing machines for which only trivial problems are decidable and a lot of problems are not semi-decidable. When Theorem 10 is applied, one has to be a little bit careful about the negation of property P. For example, finiteness is not semi-decidable for Turing machines. Not finite means infinite, which is also not semi-decidable for Turing machines. On the other hand, emptiness is not semi-decidable whereas its negation is.

In [13] complex Turing machine computations have been encoded into small grammars. These encodings and variants thereof are of tangible advantage for our purposes. Basically, we consider *valid computations of Turing machines*. Roughly speaking, these are histories of accepting Turing machine computations. It suffices to consider deterministic Turing machines with one single tape and one single read–write head. Without loss of generality and for technical reasons, we assume that the Turing machines can halt only after an odd number of moves, accept by halting, make at least three moves, and cannot

print a blank. A valid computation is a string built from a sequence of configurations passed through during an accepting computation. Let S be the *state set* of some Turing machine M, where s_0 is the *initial state*, $T \cap S = \emptyset$ is the *tape alphabet* containing the *blank symbol*, $A \subset T$ is the *input alphabet*, and $F \subseteq S$ is the set of *accepting states*. Then a *configuration* of M can be written as a word of the form T^*ST^* such that $t_1t_2 \dots t_i s t_{i+1} \dots t_n$ is used to express that M is in state s, scanning tape symbol t_{i+1}, and $t_1, t_2, \dots,$ to t_n is the support of the tape inscription.

For the purpose of the following, *valid computations* VALC(M) are now defined to be the set of words $\$w_1\$w_2\$ \dots \$w_{2n}\$$, where $\$ \notin T \cup S$, $w_i \in T^*ST^*$ are configurations of M, w_1 is an initial configuration of the form s_0A^*, w_{2n} is an accepting configuration of the form T^*FT^*, and w_{i+1} is the successor configuration of w_i, for $1 \leq i < 2n$. The set of *invalid computations* INVALC(M) is the complement of VALC(M) with respect to the alphabet $\{\$\} \cup T \cup S$.

Now the results on the trade-offs between the levels of the head hierarchy for two-way automata are complemented. Non-recursive trade-offs between the levels of the head hierarchies of deterministic as well as nondeterministic one-way devices have been shown. Moreover, non-recursive trade-offs appear between nondeterministic two-head and deterministic k-head automata. The following results of the present section are from [33]. The next example separates one-head automata from two-head automata since the former are, trivially, equivalent to finite automata.

Example 11. Let M be a Turing machine. Then a 1DFA(2) M' can be constructed that accepts VALC(M). One task of M' is to verify the correct form of the input, that is, whether it is of the form $\$s_0A^*\$T^*ST^*\$ \dots \$T^*ST^*\$T^*FT^*\$$. This task means to verify whether the input belongs to a regular set and can be done in parallel to the second task.

The second task is to verify for each two adjacent subwords whether the second one represents the successor configuration of the first one. We show the construction for $w_i\$w_{i+1}$. Starting with the first head on the first symbol of w_i and the second head on the first symbol of w_{i+1}, automaton M' compares the subwords symbolwise by moving the heads to the right. Turing machine M has three possibilities to move its head. So, $w_i = t_1t_2 \dots t_i s t_{i+1} \dots t_n$ goes to $t_1t_2 \dots t_i s' t'_{i+1} \dots t_n$, to $t_1t_2 \dots s' t_i t'_{i+1} \dots t_n$, or to $t_1t_2 \dots t_i t'_{i+1} s' \dots t_n$. Each of the three possibilities can be detected by M'. Furthermore, M' can verify whether the differences between w_i and w_{i+1} are due to a possible application of the transition function of M. Finally, the first head is moved on the first symbol of w_{i+1}, and the second head is moved on the first symbol of w_{i+2} to start the verification of w_{i+1} and w_{i+2}. □

An immediate consequence is non-recursive trade-offs between 1DFA(k), for $k \geq 2$, and descriptional systems for the context-free languages like, for example, context-free grammars or pushdown automata. To give a flavor of how to apply the technique of Theorem 10 we include the proof of the next theorem.

Theorem 12. *Let $k \geq 2$. The trade-off between 1DFA(k) and nondeterministic pushdown automata is non-recursive.*

Proof. In order to apply Theorem 10, let E_3 be the set of Turing machines. For every $M \in E_3$, we can construct a 1DFA(k) accepting VALC(M). In [13] it was shown that VALC(M) is context free if and only if $L(M)$ is finite. Since infiniteness is not semi-decidable for Turing machines, all conditions of Theorem 10 are satisfied and the assertion follows. □

Corollary 13. *Let $k \geq 2$. The trade-off between 1DFA(k) and 1NFA(1) is non-recursive.*

The next non-recursive trade-offs compare k-head and $(k+1)$-head deterministic and nondeterministic finite automata. They are obtained by applying Theorem 10. Proofs of the results may be found in [33].

In the following we exploit the basic hierarchy theorem shown in [64], and recall that the language

$$L_n = \{w_1\$w_2\$ \dots \$w_{2n} \mid w_i \in \{a, b\}^* \text{ and } w_i = w_{2n+1-i}, \text{ for } 1 \leq i \leq n\}$$

is accepted by some deterministic or nondeterministic one-way k-head finite automaton if and only if $n \leq \binom{k}{2}$. Now the language is extended in order to meet our purposes. Basically, the idea is to keep the structure of the language but to build the subwords w_i over an alphabet of pairs, that is, $w_i = \begin{smallmatrix} u_1 & u_2 & \dots & u_m \\ v_1 & v_2 & \dots & v_m \end{smallmatrix}$, where the u_j and v_j are symbols such that the upper parts of the subwords are words over $\{a, b\}$ as in L_n. The lower parts are valid computations of some given Turing machine M. Let $W_M = \{\begin{smallmatrix} u \\ v \end{smallmatrix} \mid u \in \{a, b\}^*, v \in \text{VALC}(M), |u| = |v|\}$. Then $L_{n,M}$ is defined to be

$$\{w_1\$w_2\$ \dots \$w_{2n} \mid w_i \in W_M \text{ and } w_i = w_{2n+1-i}, \text{ for } 1 \leq i \leq n\}.$$

Lemma 14. *Let $k \geq 2$ and M be some Turing machine. Then a 1DFA($k+1$) can be constructed that accepts $L_{\binom{k}{2}+1,M}$.*

It is a straightforward adaptation of the hierarchy result of [64] to prove that the language $L_{\binom{k}{2}+1,M}$ is not accepted by any 1NFA(k) if $L(M)$ is infinite.

Lemma 15. *Let $k \geq 2$ and M be some Turing machine accepting an infinite language. Then $L_{\binom{k}{2}+1,M}$ is not accepted by any 1NFA(k).*

Now an application of Theorem 10 yields the non-recursive trade-offs between any two levels of the deterministic or nondeterministic head hierarchy by using the results of Lemmas 14 and 15 on the languages $L_{\binom{k}{2}+1,M}$.

Theorem 16. *Let $k \geq 1$. The trade-offs between 1DFA($k+1$) and 1DFA(k), between 1NFA($k+1$) and 1NFA(k), and between 1DFA($k+1$) and 1NFA(k) are non-recursive.*

The next question asks for the trade-offs between nondeterministic and deterministic automata. Clearly, the trade-off between 1NFA(1) and 1DFA(1) is recursive. But following an idea in [64] which separates nondeterminism from determinism, non-recursive trade-offs on every level $k \geq 2$ of the hierarchy are shown in [33]. Again, the theorem is proved by applying Theorem 10.

Theorem 17. *Let $k \geq 2$. Then the trade-off between 1NFA(2) and 1DFA(k) and, thus, between 1NFA(k) and 1DFA(k) is non-recursive.*

Another consequence of the fact that the set of valid computations is accepted by k-head finite automata is that many of their properties are not even semi-decidable. We can transfer the results from Turing machines.

Theorem 18. *Let $k \geq 2$. Then the problems of emptiness, finiteness, infiniteness, universality, inclusion, equivalence, regularity, and context freeness are not semi-decidable for $\mathscr{L}$(1DFA(k)), $\mathscr{L}$(1NFA(k)), $\mathscr{L}$(2DFA(k)), and $\mathscr{L}$(2NFA(k)).*

In general, a family $\mathscr{L}$ of languages possesses a *pumping lemma in the narrow sense* if for each $L \in \mathscr{L}$ there exists a constant $n \geq 1$ computable from L such that each $z \in L$ with $|z| > n$ admits a factorization $z = uvw$, where $|v| \geq 1$ and $u'v^iw' \in L$, for infinitely many $i \geq 0$. The prefix u' and the suffix w' depend on u, w, and i.

Theorem 19. *Let $k \geq 2$. Then any language family whose word problem is semi-decidable and that effectively contains the language families $\mathscr{L}$(1DFA(k)), $\mathscr{L}$(1NFA(k)), $\mathscr{L}$(2DFA(k)), or $\mathscr{L}$(2NFA(k)) does not possess a pumping lemma (in the narrow sense).*

Proof. Let L be a language from the family in question and assume that there is a pumping lemma. Clearly, L is infinite if and only if it contains some w with $|w| > n$. So, we can semi-decide infiniteness by first computing n and then verifying for all words longer than n whether they belong to L. If at least for one word the answer is in the affirmative, then by pumping infinitely many words belong to L. This is a contradiction to the fact that infiniteness is not semi-decidable as is shown in Theorem 18. □

Finally, note that no minimization algorithm for the aforementioned devices exists, since otherwise emptiness becomes decidable, which contradicts Theorem 18.

Theorem 20. *There is no minimization algorithm converting some 1DFA(k), 1NFA(k), 2DFA(k), or 2NFA(k), for $k \geq 2$, to an equivalent automaton of the same type with a minimal number of states.*

Proof. For a given input alphabet A, we consider a minimal automaton accepting the empty language. It has one state which is non-accepting. Assume that there is a minimization algorithm. Then we can minimize an arbitrary automaton under consideration and check whether the result has one state that is rejecting. In this way emptiness becomes decidable, which is a contradiction to Theorem 18. □

4. Alternative and intermediate computational models

In the previous section we have seen that multi-head finite automata are a very powerful model. Their capacities have consequences on their descriptional and computational complexity, that is, the existence of non-recursive trade-offs and the undecidability of many decidability problems. This is at least disconcerting from an applied perspective, where we are much more interested in recursive trade-offs and problems that are decidable. So, the question arises what are the reasons for non-recursive trade-offs, and under which assumptions undecidable questions become decidable. Here we focus on three different alternative and intermediate computational models, namely (1) multi-head automata accepting bounded languages, (2) data-independent or oblivious multi-head finite automata, and (3) parallel communicating finite automata. We continue with multi-head finite automata accepting bounded languages.

4.1. Multi-head finite automata accepting bounded languages

If we impose the structural restriction of "boundedness," that is, all languages considered have only words of the form $a_1^*a_2^* \dots a_n^*$, for distinct letters $a_1, a_2, \dots, a_n$, then for context-free grammars and pushdown automata it is known that the trade-offs become recursive and decidability questions become decidable [36]. In this section, we summarize the corresponding results for multi-head finite automata accepting bounded languages. They have interesting connections to semilinear sets. Let us first recall the necessary definitions and notations.

Definition 21. Let $A = \{a_1, a_2, \dots, a_n\}$. A language $L \subseteq A^*$ is said to be *letter bounded* (or *bounded*) if $L \subseteq a_1^*a_2^* \dots a_n^*$.

Next we define semilinear sets.

Definition 22. A subset $P \subseteq \mathbb{N}^n$ is said to be a *linear set* if there exist $\alpha_0, \alpha_1, \dots, \alpha_m$ in $\mathbb{N}^n$ such that

$$P = \left\{ \beta \mid \beta = \alpha_0 + \sum_{i=1}^{m} a_i\alpha_i, \text{ where } a_i \geq 0, \text{ for } 1 \leq i \leq m \right\}.$$

A subset of $\mathbb{N}^n$ is said to be *semilinear* if it is the finite union of linear sets.

In order to relate bounded languages and semilinear sets we introduce the following notation. For an alphabet $A = \{a_1, a_2, \ldots, a_n\}$ the *Parikh mapping* $\Psi : \Sigma^* \to \mathbb{N}^n$ is defined by $\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$, where $|w|_{a_i}$ denotes the number of occurrences of a_i in the word w. We say that a language is *semilinear* if its Parikh image is a semilinear set.

In [44] a fundamental result concerning the distribution of symbols in the words of a context-free language has been shown. It says that for any context-free language L, the Parikh image $\Psi(L) = \{\Psi(w) \mid w \in L\}$ is semilinear. Semilinear sets have many appealing properties. For example, they are closed under union, intersection, and complementation [11]. Furthermore, bounded semilinear languages have nice decidability properties, since the questions of emptiness, universality, finiteness, infiniteness, inclusion, and equivalence are decidable (see [11]). The main connection between bounded languages and multi-head finite automata is the following result [22,59].

Lemma 23. *Let $k \geq 1$ and L be a bounded language accepted by a 2NFA(k) which performs a constant number of head reversals. Then $\Psi(L)$ is a semilinear set.*

The above characterization implies immediately that there are no one-way multi-head finite automata which accept the non-semilinear languages $L_1 = \{a^{2^n} \mid n \geq 1\}$ or $L_2 = \{a^n b^{i \cdot n} \mid i, n \geq 1\}$. Additionally, if two-way multi-head finite automata accept these languages, then the number of head reversals on the input tape must be unbounded. The situation changes for unbounded languages. There are languages accepted by deterministic one-way multi-head finite automata, whose Parikh map is not semilinear. For example, the language $L_3 = \{aba^2ba^3b \ldots a^nb \mid n \geq 1\}$ is accepted by some 1DFA(2) by checking that the size of adjacent a-blocks is always increasing by one. On the other hand, it is easily observed that the Parikh image $\Psi(L_3)$ is not semilinear. Due to the relation to semilinear sets and their positive decidability results we obtain the borderline between decidability and undecidability as follows.

Theorem 24. *Let $k \geq 2$. The problems of emptiness, universality, finiteness, infiniteness, inclusion, and equivalence are undecidable for the language families $\mathscr{L}$(1DFA(k)), $\mathscr{L}$(1NFA(k)), $\mathscr{L}$(2DFA(k)), and $\mathscr{L}$(2NFA(k)), if the automata are allowed to perform at most a constant number of head reversals. The problems become decidable for bounded languages.*

Proof. The undecidability results follow from Theorem 18. For the decidability results in the bounded case we use Lemma 23 and the fact that the above questions are decidable for semilinear sets. □

We next summarize results concerning decidability questions on models which are generalized or restricted versions of multi-head finite automata. Let us start with nondeterministic two-way multi-head pushdown automata [22] which are nondeterministic two-way multi-head finite automata augmented by a pushdown store. In fact, the characterization in Lemma 23 has been shown in [22] for the stronger model of nondeterministic two-way multi-head pushdown automata.

Lemma 25. *Let $k \geq 1$ and L be a bounded language accepted by an nondeterministic two-way k-head pushdown automaton which performs a constant number of head reversals. Then $\Psi(L)$ is a semilinear set.*

Thus, the positive decidability results obviously hold also for such automata and we obtain the following corollary.

Corollary 26. *Let $k \geq 1$. The problems of emptiness, universality, finiteness, infiniteness, inclusion, and equivalence are decidable for nondeterministic two-way k-head pushdown automata accepting a bounded language, if the automata are allowed to perform at most a constant number of head reversals.*

As is the case for multi-head finite automata, we lose positive decidability and semilinearity if we drop the condition of boundedness. We may consider again the language L_3 which is clearly accepted by a one-way two-head pushdown automaton as well.

In the general model of multi-head finite automata the moves of the heads are only depending on the input, and a head cannot feel the presence of another head at the same position at the same time. If this property is added to each head, we come to the model of one-way multi-head finite automata having *sensing heads* (see [22]). This model is stronger since non-semilinear unary languages can be accepted whereas with non-sensing heads only semilinear languages are accepted in the unary case (see Lemma 23). For example, the non-semilinear language $\{a^{n^2} \mid n \geq 1\}$ can be accepted by a deterministic one-way finite automaton having three sensing heads. On the other hand, it is shown in [22] that every bounded language which is accepted by some nondeterministic one-way finite automaton having two sensing heads can be also accepted by some nondeterministic one-way pushdown automaton with two heads, and thus is semilinear due to Lemma 25. This borderline on the computational capacity between two and three sensing heads carries over to the border between decidability and undecidability as follows.

Theorem 27. 1. *The problems of emptiness, universality, finiteness, infiniteness, inclusion, and equivalence are undecidable for deterministic one-way finite automata with at least three sensing heads.*
2. *The aforementioned problems become decidable for nondeterministic one-way finite automata with two sensing heads that accept a bounded language.*

Proof. The undecidability results are shown in [46]. The decidability results follow from the characterization by nondeterministic one-way pushdown automata with two heads [22], Lemma 25, and the fact that the questions are decidable for semilinear sets. □

The computational capacity of two-way finite automata with sensing heads and bounds on the number of input reversals is investigated in [20]. We next consider a restricted version of one-way multi-head finite automata which allows only one designated head to read and distinguish input symbols whereas the remaining heads cannot distinguish input symbols, that is, the remaining heads get only the information whether they are reading an input symbol or the right endmarker. These automata are called *partially blind* multi-head finite automata [27]. In this model the decidability results are much better, since it can be shown that every language, not necessarily bounded, which is accepted by a partially blind nondeterministic one-way multi-head finite automaton has a semilinear Parikh map. In detail, we have the following theorem from [27].

Theorem 28. 1. *The problems of emptiness, finiteness, and infiniteness are decidable for partially blind nondeterministic one-way multi-head finite automata. Universality is undecidable and, thus, the problems of inclusion and equivalence are undecidable as well.*
2. *The language family accepted by partially blind deterministic one-way multi-head finite automata is closed under complementation. Thus, the problems of universality, inclusion, and equivalence are decidable.*
3. *The problems of emptiness, universality, finiteness, infiniteness, inclusion, and equivalence are decidable for partially blind nondeterministic one-way multi-head finite automata that accept a bounded language.*

Partially blind versions of one-way multi-head pushdown automata have been considered in [22,26]. The results are similar to the results for partially blind multi-head finite automata. Finally, we consider the related model of two-way one-head multi-counter machines which has been introduced in [23]. Such a machine can be described by a two-way finite automaton augmented by some fixed number of counters which can store any natural number. In [23] many results concerning the computational capacity of such machines are shown. Here, we want to summarize the results concerning their decidability problems. We obtain similar decidability results when both the number of head reversals as well as the number of counter reversals is bounded by some constant. For example, for nondeterministic two-way machines the questions of emptiness, finiteness, and infiniteness are decidable. For deterministic two-way machines the questions of universality, inclusion, and equivalence are decidable, whereas the questions become undecidable for nondeterministic machines. If we restrict ourselves to bounded languages, then the latter questions are decidable also in the nondeterministic case. If the number of head reversals or the number of counter reversals is not bounded by some constant, then the decidability questions become undecidable.

Let us finally mention some open problems in connection with bounded languages. (1) We know from Section 3 that comparing unrestricted multi-head finite automata concerning their descriptional complexity leads to non-recursive trade-offs. Even the restriction to unary languages yields non-recursive trade-offs due to Theorem 9. In this section, we have seen that the restriction to bounded languages and a finite number of head reversals leads to decidable questions and recursive trade-offs are expected. It would be interesting to know precise bounds, that is, whether the trade-off can be bounded by a polynomial, exponential, or super-exponential function. (2) Additionally, it is worth determining the trade-offs between the other variants and models discussed in this section. Do there exist conversion algorithms and, if so, how precisely can upper and lower bounds be determined? The decidability of equivalence for certain multi-head finite automata immediately leads to a naive, brute force minimization algorithm. It would be very interesting to determine the computational complexity of the minimization problem for those models with decidable equivalence problem. Does an efficient minimization algorithm exist or is it possible to show that the problem is computationally hard?

4.2. Data-independent multi-head finite automata

The notion of oblivious Turing programs was introduced in [45]. This concept was used in [48,55] to simulate Turing machines by logical networks. Moreover, obliviousness was studied in the context of parallel random access machines (PRAMs), in order to obtain characterizations of polynomial size, poly-logarithmic depth unbounded and bounded fan-in circuit classes by variants of PRAMs with oblivious or non-oblivious read and write structures [34]. Recently, obliviousness was investigated in relation with computational models from classical automata theory as, for example, multi-head finite automata in [18].

Definition 29. A k-head finite automaton M is *data-independent* or *oblivious* if the position of every *input-head* i after step t in the computation on input w is a function $f_M(|w|, i, t)$ that only depends on i, t, and $|w|$.

We denote deterministic (nondeterministic) one-way and two-way data-independent k-head finite automata by 1DiDFA(k) and 2DiDFA(k) (1DiNFA(k) and 2DiNFA(k)). The corresponding language classes are similarly denoted as in the previous sections.

Although we have defined data independence only for k-head finite automata, the same definition obviously applies to machines having a finite control, several input heads, and an extra storage medium like, for example, a pushdown store or a stack. In [47] it was shown that data independence is no restriction for multi-head one-counter automata, multi-head non-erasing stack automata, and multi-head stack automata. The same is obviously true for one-head finite automata.

Interestingly, one can show that for data-independent k-head finite automata determinism is as powerful as nondeterminism. To this end, observe that a data-independent automaton with k heads has, on inputs of length n, only one possible input-head trajectory during a computation. So, the only nondeterminism left in the k-head nondeterministic finite automata is the way in which the next state is chosen. Therefore, a powerset construction shows that even data-independent determinism may simulate data-independent nondeterminism. Thus, we have the following theorem from [17,18].

Theorem 30. *Let $k \geq 1$. Then*

1. $\mathscr{L}(1DiDFA(k)) = \mathscr{L}(1DiNFA(k))$ *and*
2. $\mathscr{L}(2DiDFA(k)) = \mathscr{L}(2DiNFA(k))$.

But why are data-independent multi-head finite automata that interesting? Similarly as ordinary multi-head finite automata characterize logarithmic space bounded Turing machine computations [14], data-independent multi-head finite automata capture the parallel complexity class NC^1. Here, NC^1 is the set of problems accepted by log-space uniform circuit families of logarithmic depth, polynomial size, with AND and OR gates of bounded fan-in. We have $\mathsf{NC}^1 \subseteq \mathsf{L}$. For the characterization the following observation is useful: Every deterministic multi-head finite automaton that works on unary inputs is already data-independent by definition. Therefore,

$$\mathsf{L}^{\mathsf{u}} \subseteq \bigcup_{k\geq 1} \mathscr{L}(2\text{DiDFA}(k))$$

where L^{u} denotes the set of all unary languages from L. However, because of the trajectory argument, we do not know whether the inclusion $\mathsf{NL}^{\mathsf{u}} \subseteq \bigcup_{k\geq 1} 2\text{DiNFA}(k)$ holds. A positive answer would imply $\mathsf{NL}^{\mathsf{u}} \subseteq \mathsf{L}^{\mathsf{u}}$ by Theorem 30. This, in turn, would lead to a positive answer of the *linear bounded automaton (LBA) problem* by translational methods [31]. In [18] the following theorem was shown.

Theorem 31. $\mathsf{NC}^1 = \bigcup_{k\geq 1} \mathscr{L}(2DiDFA(k))$.

Since there exists a log-space complete language in $\mathscr{L}(1\text{DFA}(2))$, we immediately obtain

$$\mathsf{NC}^1 = \mathsf{L} \quad \text{if and only if} \quad \mathscr{L}(1\text{DFA}(2)) \subseteq \bigcup_{k\geq 1} \mathscr{L}(2\text{DiDFA}(k)).$$

Similarly,

$$\mathsf{NC}^1 = \mathsf{NL} \quad \text{if and only if} \quad \mathscr{L}(1\text{NFA}(2)) \subseteq \bigcup_{k\geq 1} \mathscr{L}(2\text{DiDFA}(k)).$$

What is known about the head hierarchies induced by data-independent multi-head finite automata? For two-way data-independent multi-head finite automata one can profit from known results [40]. Since for every $k \geq 1$ there are unary languages that may serve as witnesses for the inclusions $\mathscr{L}(2\text{DFA}(k)) \subset \mathscr{L}(2\text{DFA}(k+1))$ and $\mathscr{L}(2\text{NFA}(k)) \subset \mathscr{L}(2\text{NFA}(k+1))$, we obtain the following corollary.

Corollary 32. *Let $k \geq 1$. Then $\mathscr{L}(2DiDFA(k)) \subset \mathscr{L}(2DiDFA(k+1))$.*

The remaining inclusions $\mathscr{L}(2\text{DiDFA}(k)) \subseteq \mathscr{L}(2\text{DFA}(k))$, for $k \geq 2$, for two-way multi-head finite automata are related to the question of whether $\mathsf{NC}^1 = \mathsf{L}$. This parallels the issue of whether the inclusion $\mathscr{L}(2\text{DFA}(k)) \subseteq \mathscr{L}(2\text{NFA}(k))$ is proper for $k \geq 2$. For the relationship between data-independent and data-dependent multi-head finite automata, we find that $\mathscr{L}(2\text{DiDFA}(k)) = \mathscr{L}(2\text{DFA}(k))$, for some $k \geq 2$, implies $\mathsf{NC}^1 = \mathsf{L}$.

Moreover, it was shown in [17] that the head hierarchy for one-way data-independent multi-head finite automata is strict. Obviously, $\text{REG} = \mathscr{L}(1\text{DiDFA}(1))$ and, hence, $\text{REG} \subset \mathscr{L}(2\text{DiDFA}(2))$. By adapting the head hierarchy result of [64] to data-independent automata, in [17] it is shown, too, that $\frac{k\cdot(k+1)}{2} + 4$ heads are sufficient to accept the witness language not being accepted with k heads.

Theorem 33. *Let $k \geq 1$. Then $\mathscr{L}(1DiDFA(k)) \subset \mathscr{L}(1DiDFA(\frac{k\cdot(k+1)}{2} + 4))$.*

Proof. We modify the language L_n from [64] as follows: Define

$$L'_n = \{w_1\$w_2\$\ldots\$w_{2n} \mid w_i \in \{a,b\}^m \text{ and } w_i = w_{2n+1-i}, \text{ for } 1 \leq i \leq n\}.$$

By similar arguments as in [64] one observes that L'_n is accepted by some deterministic one-way k-head finite automaton if and only if $n \leq \binom{k}{2}$—the essential difference between words from L_n and L'_n is that the w_i subwords are of the same length in the latter language. Now set $n = \binom{k}{2} + 1$.

The one-way $\frac{k\cdot(k+1)}{2} + 4$-head data-independent deterministic finite automaton which accepts L'_n works as follows: One head traverses $w_{\frac{k\cdot(k+1)}{2}+2}$, $w_{\frac{k\cdot(k+1)}{2}+3}$, …, $w_{k\cdot(k+1)+2}$, and $\frac{k\cdot(k+1)}{2} + 1$ heads scan the appropriate words on the left half of the input. To start the computation, another head is needed for appropriately positioning these heads on the \$ symbols to start the comparison of word pairs. This is done by sending all these heads with appropriate speeds to the right—this is nothing other than the technique of "sending signals with different speeds" from cellular automata theory. Here the input heads play the role of signals. Then, the comparison of the words is done. First, $w_{\frac{k\cdot(k+1)}{2}+1}$ is checked against $w_{\frac{k\cdot(k+1)}{2}+2}$, which requires one additional head. Afterwards, the head which has scanned the word $w_{\frac{k\cdot(k+1)}{2}+1}$ on the left half of the input is free to do the necessary repositioning for the comparison of $w_{\frac{k\cdot(k+1)}{2}+3}$ with $w_{\frac{k\cdot(k+1)}{2}}$. This goes on until the last word $w_{k\cdot(k+1)+2}$ is checked against w_1. Since by the choice of n the language L'_n is not accepted by any one-way deterministic or nondeterministic finite automaton with k heads, the claim follows. □

Thus, we have an infinite proper hierarchy with respect to the number of heads, but the bound obtained in Theorem 33 is not very good, especially for small values of k. In fact, a separation for the first four levels was obtained in [17], using the language

$$L_{F,n} = \{a^{i\cdot F(2)}\$a^{i\cdot F(3)}\$\ldots\$a^{i\cdot F(k+1)} \mid i \geq 1\},$$

where $F(j)$ is the jth *Fibonacci number*. It follows that $L_{F,n} \in \mathscr{L}(1\text{DiDFA}(k))$, if and only if $n \leq \frac{k\cdot(k-1)}{2} + 1$.

Theorem 34. $\mathscr{L}(1DiDFA(1)) \subset \mathscr{L}(1DiDFA(2)) \subset \mathscr{L}(1DiDFA(3)) \subset \mathscr{L}(1DiDFA(4))$.

Whether the one-way hierarchy for data-independent multi-head finite automata is strict in the sense that $k+1$ heads are better than k, is an open problem. Concerning the remaining open inclusions the following is known.

It is well known that the copy language $\{ww \mid w \in \{a,b\}^+\}$ cannot be accepted by any 1NFA(k) with $k \geq 1$, which is shown by a simple counting argument. On the other hand it is readily verified that the language belongs to 2DiDFA(2). Consequently, we obtain the following separation result from [17,18].

Theorem 35. *Let $k \geq 2$. Then $\mathscr{L}(1DiDFA(k)) \subset \mathscr{L}(2DiDFA(k))$.*

The copy language used in the previous proof shows even more.

Corollary 36. $\bigcup_{k\geq 1}\mathscr{L}(1DiDFA(k)) \subset \bigcup_{k\geq 1}\mathscr{L}(2DiDFA(k)) = \mathsf{NC}^1$.

We close this subsection by mentioning some open problems for further research: (1) Determine the bounds of the conversions of one-head data-independent finite automata into one-head data-dependent deterministic, nondeterministic, and alternating finite automata and *vice versa*. (2) Consider decidability and complexity questions such as equivalence, non-emptiness, etc., for k-head data-independent finite automata. Finally, the most interesting point for research might be (3) the one-way k-head hierarchy for data-independent finite automata. Is it a strict hierarchy, in the sense that $k+1$ heads are better than k?

4.3. Parallel communicating finite automata

Multi-head finite automata are in some sense the simplest model of cooperating sequential automata. They can be seen as a model with one finite state control, and the cooperation between the finite state control and the single components consists of reading the input and positioning the heads. Multi-processor automata [5] are in a way restricted multi-head finite automata. The relation between both classes is investigated in [9]. Systems of different finite automata communicating by appropriate protocols are described in [4,30]. Here, we will focus on parallel communicating finite automata systems which were introduced in [37]. Proofs of the results presented in this subsection may be found in [3].

A parallel communicating finite automata system of degree k is a device of k finite automata working in parallel with each other on a common one-way read-only input tape and being synchronized according to a global clock. The k automata communicate on request by states, that is, when some automaton enters a distinguished query state q_i, it is set to the current state of automaton A_i. Concerning the next state of the sender A_i, we distinguish two modes. In the *non-returning* mode the sender remains in its current state whereas in *returning* mode the sender is set to its initial state. Moreover, we distinguish whether all automata are allowed to request communications, or whether there is just one master allowed to request communications. The latter types are called *centralized*. Whenever the transition function of (at least) one of the single automata is undefined the whole systems halts. Whether the input is accepted or rejected depends on the states of the automata having halting transitions. The input is accepted if at least one of them is in an accepting state.

Definition 37. A *nondeterministic parallel communicating finite automata system of degree k* (PCFA(k)) is a construct $\mathcal{A} = \langle \Sigma, A_1, A_2, \ldots, A_k, Q, \triangleleft\rangle$, where Σ is the set of *input symbols*, each $A_i = \langle S_i, \Sigma, \delta_i, s_{0,i}, F_i\rangle$, for $1 \leq i \leq k$, is a *nondeterministic finite automaton* with state set S_i, initial state $s_{0,i} \in S_i$, set of accepting states $F_i \subseteq S_i$, and transition function $\delta_i : S_i \times (\Sigma \cup \{\lambda, \triangleleft\}) \to 2^{S_i}$, $Q = \{q_1, q_2, \ldots, q_k\} \subseteq \bigcup_{1\leq i\leq k} S_i$ is the set of *query states*, and $\triangleleft \notin \Sigma$ is the *end-of-input symbol*.

The automata $A_1, A_2, \ldots, A_k$ are called *components* of the system $\mathcal{A}$. A *configuration* $(s_1, x_1, s_2, x_2, \ldots, s_k, x_k)$ of $\mathcal{A}$ represents the current states s_i as well as the still unread parts x_i of the tape inscription of all components $1 \leq i \leq k$. System $\mathcal{A}$ starts with all of its components scanning the first square of the tape in their initial states. For input word $w \in \Sigma^*$, the initial configuration is $(s_{0,1}, w\triangleleft, s_{0,2}, w\triangleleft, \ldots, s_{0,k}, w\triangleleft)$. Basically, a computation of $\mathcal{A}$ is a sequence of configurations beginning with an initial configuration and ending with a halting configuration. Each step can consist of two phases. In a first phase, all components are in non-query states and perform an ordinary (non-communicating) step independently. The second phase is the communication phase during which components in query states receive the requested states as long as the sender is not in a query state itself. This process is repeated until all requests are resolved, if possible. If the requests are cyclic, no successor configuration exists. As mentioned above, we distinguish *non-returning* communication, that is, the sender remains in its current state, and *returning* communication, that is, the sender is reset to its initial state.

For the first phase, we define the successor configuration relation $\vdash$ by

$$(s_1, a_1y_1, s_2, a_2y_2, \ldots, s_k, a_ky_k) \vdash (p_1, z_1, p_2, z_2, \ldots, p_k, z_k),$$

if $Q \cap \{s_1, s_2, \ldots, s_k\} = \emptyset$, $a_i \in \Sigma \cup \{\lambda, \triangleleft\}$, $p_i \in \delta_i(s_i, a_i)$, and $z_i = \triangleleft$ for $a_i = \triangleleft$ and $z_i = y_i$ otherwise, for $1 \leq i \leq k$. For non-returning communication in the second phase, we set

$$(s_1, x_1, s_2, x_2, \ldots, s_k, x_k) \vdash (p_1, x_1, p_2, x_2, \ldots, p_k, x_k),$$

if, for all $1 \leq i \leq k$ such that $s_i = q_j$ and $s_j \notin Q$, we have $p_i = s_j$, and $p_r = s_r$ for all the other r, for $1 \leq r \leq k$. Alternatively, for returning communication in the second phase, we set

$$(s_1, x_1, s_2, x_2, \ldots, s_k, x_k) \vdash (p_1, x_1, p_2, x_2, \ldots, p_k, x_k),$$

if, for all $1 \leq i \leq k$ such that $s_i = q_j$ and $s_j \notin Q$, we have $p_i = s_j$, $p_j = s_{0,j}$, and $p_r = s_r$ for all the other r with $1 \leq r \leq k$.

A computation *halts* when the successor configuration is not defined for the current situation. In particular, this may happen when cyclic communication requests appear, or when the transition function of one component is not defined. We regard the transition function as undefined whenever it maps to the empty set. The language $L(\mathcal{A})$ accepted by a PCFA(k) $\mathcal{A}$ is precisely the set of words w such that there is some computation beginning with $w\triangleleft$ on the input tape and halting with at least one component having an undefined transition function and being in an accepting state. Let $\vdash^*$ denote the reflexive and transitive closure of $\vdash$ and set

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid (s_{0,1}, w\triangleleft, s_{0,2}, w\triangleleft, \ldots, s_{0,k}, w\triangleleft) \vdash^* (p_1, a_1y_1, p_2, a_2y_2, \ldots, p_k, a_ky_k)$$
$$\text{such that } p_i \in F_i \text{ and } \delta_i(p_i, a_i) \text{ is undefined, for some } 1 \leq i \leq k\}.$$

If all components A_i are deterministic finite automata, that is, for all $s \in S_i$ the transition function $\delta_i(s, a)$ maps to a set of at most one state and is undefined for all $a \in \Sigma$, whenever $\delta_i(s, \lambda)$ is defined, then the whole system is called *deterministic*, and we add the prefix D to denote it. The absence or presence of an R in the type of the system denotes whether it works in *non-returning* or *returning* mode, respectively. Finally, if there is just one component, say A_1, that is allowed to query for states, that is, $S_i \cap Q = \emptyset$, for $2 \leq i \leq k$, then the system is said to be *centralized*. We denote centralized systems by a C. Whenever the degree is missing we mean systems of arbitrary degree. The corresponding language classes are similarly denoted as in the previous sections.

In order to clarify our notation we give an example.

Example 38. We consider the language $\{w\$w \mid w \in \{a, b\}^+\}$ and show that it can be accepted by a DCPCFA with two components. In [3] it is additionally shown that this language can be also accepted by a DRCPCFA with two components. Thus, all types of systems of parallel communicating finite automata accept more than regular languages.

The rough idea of the construction of a DCPCFA is that in every time step the master component queries the non-master component, and the non-master component reads an input symbol. When the non-master component has read the separating symbol \$, which is notified to the master with the help of primed states, then the master component starts to compare its input symbol with the information from the non-master component. If all symbols up to \$ match, the input is accepted and in all other cases rejected. The precise construction is given through the following transition functions.

$$\begin{array}{lll} \delta_1(s_{0,1}, \lambda) = q_2 & \delta_1(s_1, \lambda) = q_2 & \delta_1(s_\$, \lambda) = q_2 \\ \delta_1(s'_a, a) = q_2 & \delta_1(s'_b, b) = q_2 & \delta_1(s_\triangleleft, \$) = \mathit{accept} \end{array}$$

$$\begin{array}{lll} \delta_2(s_{0,2}, a) = s_1 & \delta_2(s_{0,2}, b) = s_1 & \\ \delta_2(s_1, a) = s_1 & \delta_2(s_1, b) = s_1 & \delta_2(s_1, \$) = s_\$ \\ \delta_2(s_\$, a) = s'_a & \delta_2(s_\$, b) = s'_b & \delta_2(s_\triangleleft, \triangleleft) = s_\triangleleft \\ \delta_2(s'_a, a) = s'_a & \delta_2(s'_a, b) = s'_b & \delta_2(s'_a, \triangleleft) = s_\triangleleft \\ \delta_2(s'_b, a) = s'_a & \delta_2(s'_b, b) = s'_b & \delta_2(s'_b, \triangleleft) = s_\triangleleft. \end{array}$$

This completes the description of the transition functions. □

For nondeterministic non-centralized devices it is shown in [7] that returning parallel communicating finite automata systems are neither weaker nor stronger than non-returning ones. In the same paper the question is raised whether the same equivalence is true in the deterministic case. In [3] the question was answered in the affirmative. To this end, a general method to send information tokens cyclically through the returning components is introduced. Basically, for the so-called *cycling-token method* an information token is a finite record of data that can be read or written by the components. So, it can be represented by states. The precise structure of the token depends on the application. The main problem to cope with is to break the synchronization at the beginning. Otherwise, when some component A_{i+1} requests the state of A_i and, thus, A_i reenters its initial state, then A_i will request the state of A_{i-1} and so on. But these cascading communication requests would destroy necessary information. Details of the method can be found in [3].

The next lemma has been shown by applying the cycling-token method.

Lemma 39. *Let $k \geq 1$. Then $\mathscr{L}(DPCFA(k)) \subseteq \mathscr{L}(DRPCFA(k))$.*

One of the fundamental results obtained in [37] is the characterization of the computational power of (unrestricted) parallel communicating finite automata systems by multi-head finite automata, that is, $\mathscr{L}(\text{DPCFA}(k)) = \mathscr{L}(\text{1DFA}(k))$. In [3] it is additionally shown in what way a 1DFA(k) can simulate a given DRPCFA(k). Thus, the following characterization is obtained:

Theorem 40. *Let $k \geq 1$. Then the families $\mathscr{L}(DRPCFA(k))$, $\mathscr{L}(DPCFA(k))$, and $\mathscr{L}(1DFA(k))$ are equal.*

Comparing deterministic centralized systems with non-centralized systems the surprising result is known that the returning mode is not weaker than the non-returning mode. Let us consider the language

$$L_{\mathrm{rc}} = \{uc^x v\$uv \mid u, v \in \{a, b\}^*, x \geq 0\}.$$

Then we find the following situation:

Theorem 41. *The language L_{rc} belongs to the family $\mathscr{L}(DRCPCFA)$ (and thus to $\mathscr{L}(DRPCFA) = \mathscr{L}(DPCFA)$), but not to $\mathscr{L}(DCPCFA)$.*

As an immediate corollary we have the following strict inclusion.

Corollary 42. $\mathscr{L}(DCPCFA) \subset \mathscr{L}(DPCFA) = \mathscr{L}(DRPCFA)$.

In order to show that nondeterministic centralized systems are strictly more powerful than their deterministic variants consider the language

$$L_{\mathrm{mi}} = \{ww^R \mid w \in \{a, b, c\}^+\}.$$

It has been shown that its complement belongs to $\mathscr{L}(\mathrm{CPCFA})$, but does not belong to $\mathscr{L}(\mathrm{DPCFA})$.

Lemma 43. *The complement of the language L_{mi} belongs to the family $\mathscr{L}(CPCFA)$, but does not belong to $\mathscr{L}(DPCFA)$.*

Thus, we derive:

Corollary 44. $\mathscr{L}(DCPCFA) \subset \mathscr{L}(CPCFA)$ *and* $\mathscr{L}(DPCFA) \subset \mathscr{L}(PCFA)$.

Finally, we present results that compare the classes under consideration with some well-known language families.

Lemma 45. *The family $\mathscr{L}(PCFA)$ is strictly included in* NL*, hence, in the family of deterministic context-sensitive languages.*

Next we consider the relation to variants of context-free languages in detail.

Lemma 46. *All language classes accepted by parallel communicating finite automata systems are incomparable to the class of (deterministic) (linear) context-free languages.*

A similar situation has been shown for Church–Rosser languages which have been defined in [38] *via* finite, confluent, and length-reducing Thue systems. Church–Rosser languages are a language family that lies properly between the deterministic context-free languages and the context-sensitive languages but is incomparable to the context-free languages [6]. Church–Rosser languages are of particular interest, since they can be parsed rapidly in linear time and contain non-semilinear as well as inherently unambiguous languages.

Lemma 47. *All language classes accepted by parallel communicating finite automata systems are incomparable with the class of Church–Rosser languages.*

We close this section with some open problems: (1) The main open question concerning parallel communicating finite automata is to clarify the power of non-centralized *versus* centralized systems. Corollary 42 is a partial solution saying that in the deterministic case centralized, non-returning systems are weaker than non-centralized systems both in returning and non-returning mode. It would be very interesting to know whether this result also holds for the returning mode. In the nondeterministic case this question is also an open problem both in returning and non-returning modes. (2) Concerning the descriptional complexity non-recursive trade-offs are expected between those systems whose corresponding language classes are properly included. (3) Finally, it is clear that only semilinear languages are accepted when bounded languages are considered. How precisely can upper and lower bounds between the several variants of parallel communicating finite automata be determined?

References

[1] P. Berman, A. Lingas, On the complexity of regular languages in terms of finite automata, Technical Report 304, Polish Academy of Sciences, 1977.
[2] J.-C. Birget, State-complexity of finite-state devices, state compressibility and incompressibility, Math. Systems Theory 26 (1993) 237–269.
[3] H. Bordihn, M. Kutrib, A. Malcher, On the computational capacity of parallel communicating finite automata, in: Developments in Language Theory, DLT 2008, in: LNCS, vol. 5257, Springer, 2008.
[4] D. Brand, P. Zafiropulo, On communicating finite-state machines, J. ACM 30 (1983) 323–342.
[5] A. Buda, Multiprocessor automata, Inform. Process. Lett. 25 (1987) 257–261.
[6] G. Buntrock, F. Otto, Growing context-sensitive languages and Church-Rosser languages, Inform. Comput. 141 (1998) 1–36.
[7] A. Choudhary, K. Krithivasan, V. Mitrana, Returning and non-returning parallel communicating finite automata are equivalent, RAIRO Inform. Théor. 41 (2007) 137–145.
[8] M. Chrobak, Finite automata and unary languages, Theoret. Comput. Sci. 47 (1986) 149–158.
[9] P. Ďuriš, T. Jurdziński, M. Kutyłowski, K. Loryś, Power of cooperation and multihead finite systems, in: International Colloquium on Automata, Languages and Programming, in: LNCS, vol. 1443, Springer, 1998.
[10] V. Geffert, C. Mereghetti, G. Pighizzini, Converting two-way nondeterministic unary automata into simpler automata, Theoret. Comput. Sci. 295 (2003) 189–203.
[11] S. Ginsburg, The Mathematical Theory of Context-Free Languages, McGraw Hill, New York, 1966.

[12] J. Goldstine, M. Kappes, C.M.R. Kintala, H. Leung, A. Malcher, D. Wotschke, Descriptional complexity of machines with limited resources, J. UCS 8 (2002) 193–234.
[13] J. Hartmanis, Context-free languages and Turing machine computations, Proc. Symposia Appl. Math. 19 (1967) 42–51.
[14] J. Hartmanis, On non-determinancy in simple computing devices, Acta Inform. 1 (1972) 336–344.
[15] J. Hartmanis, On the succinctness of different representations of languages, SIAM J. Comput. 9 (1980) 114–120.
[16] J. Hartmanis, On Gödel speed-up and succinctness of language representations, Theoret. Comput. Sci. 26 (1983) 335–342.
[17] M. Holzer, Data-independent *versus* data-dependent computations on multi-head automata, Doctoral Thesis, Universität Tübingen, 1998.
[18] M. Holzer, Multi-head finite automata: data-independent versus data-dependent computations, Theoret. Comput. Sci. 286 (2002) 97–116.
[19] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Language, and Computation, Addison-Wesley, Reading, Massachusetts, 1979.
[20] J. Hromkovič, Fooling a two-way nondeterministic multihead automaton with reversal number restriction, Acta Inform. 22 (1985) 589–594.
[21] J. Hromkovič, G. Schnitger, Nondeterminism versus determinism for two-way finite automata: Generalizations of Sipser's separation, in: International Colloquium on Automata, Languages and Programming, ICALP 2003, in: LNCS, vol. 2719, Springer, 2003.
[22] O.H. Ibarra, A note on semilinear sets and bounded-reversal multihead pushdown automata, Inform. Process. Lett. 3 (1974) 25–28.
[23] O.H. Ibarra, Reversal-bounded multicounter machines and their decision problems, J. ACM 25 (1978) 116–133.
[24] O.H. Ibarra, J. Karhumäki, A. Okhotin, On stateless multihead automata: hierarchies and the emptiness problem, TUCS Technical Report 848, Turku Centre for Computer Science, 2007.
[25] O.H. Ibarra, C.E. Kim, On 3-head versus 2-head finite automata, Acta Inform. 4 (1974) 193–200.
[26] O.H. Ibarra, C.E. Kim, A useful device for showing the solvability of some decision problems, J. Comput. System Sci. 13 (1976) 153–160.
[27] O.H. Ibarra, B. Ravikumar, On partially blind multihead finite automata, Theoret. Comput. Sci. 356 (2006) 190–199.
[28] C. Kapoutsis, Non-recursive trade-offs for two-way machines, Int. J. Found. Comput. Sci. 16 (2005) 943–956.
[29] C. Kapoutsis, Removing bidirectionality from nondeterministic finite automata, in: Mathematical Foundations of Computer Science, MFCS 2005, in: LNCS, vol. 3618, Springer, 2005.
[30] R. Klemm, Systems of communicating finite state machines as a distributed alternative to finite state machines, Ph.D. Thesis, Pennsylvania State University, 1996.
[31] S.-Y. Kuroda, Classes of languages and linear-bounded automata, Inform. Control 7 (1964) 207–223.
[32] M. Kutrib, On the descriptional power of heads, counters, and pebbles, Theoret. Comput. Sci. 330 (2005) 311–324.
[33] M. Kutrib, The phenomenon of non-recursive trade-offs, Int. J. Found. Comput. Sci. 16 (2005) 957–973.
[34] K.-J. Lange, R. Niedermeier, Data-independences of parallel random access machines, in: Foundations of Software Technology and Theoretical Computer Science, FSTTCS 1993, in: LNCS, vol. 761, 1993.
[35] H. Leung, Tight lower bounds on the size of sweeping automata, J. Comput. System Sci. 63 (2001) 384–393.
[36] A. Malcher, G. Pighizzini, Descriptional complexity of bounded context-free languages, in: Developments in Language Theory, DLT 2007, in: LNCS, vol. 4588, Springer, 2007.
[37] C. Martín-Vide, A. Mateescu, V. Mitrana, Parallel finite automata systems communicating by states, Int. J. Found. Comput. Sci. 13 (2002) 733–749.
[38] R. McNaughton, P. Narendran, F. Otto, Church-Rosser Thue systems and formal languages, J. ACM 35 (1988) 324–344.
[39] A.R. Meyer, M.J. Fischer, Economy of description by automata, grammars, and formal systems, in: IEEE Symposium on Switching and Automata Theory, SWAT 1971, IEEE Press, 1971.
[40] B. Monien, Two-way multihead automata over a one-letter alphabet, RAIRO Inform. Théor. 14 (1980) 67–82.
[41] F.R. Moore, On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata, IEEE Trans. Comput. 20 (1971) 1211–1214.
[42] K. Morita, K. Sugata, H. Umeo, Computation complexity of *n*-bounded counter automaton and multidimensional rebound automaton, Syst. Comput. Controls 8 (1977) 80–87. Translated from [43].
[43] K. Morita, K. Sugata, H. Umeo, Computation complexity of n-bounded counter automaton and multidimensional rebound automaton, Denshi Tsushin Gakkai Ronbunshi 60-D (1977) 283–290.
[44] R.J. Parikh, On context-free languages, J. ACM 13 (1966) 570–581.
[45] M.S. Paterson, M.J. Fischer, A.R. Meyer, An improved overlap argument for on-line multiplication, in: Complexity of Computation, in: SIAM-AMS Proceedings, vol. 7, AMS, New Jersey, 1974.
[46] H. Petersen, Automata with sensing heads, in: Proceedings of the Third Israel Symposium on the Theory of Computing and Systems, IEEE Press, 1995.
[47] H. Petersen, The head hierarchy for oblivious finite automata with polynomial advice collapses, in: Mathematical Foundations of Computer Science, MFCS 1998, in: LNCS, vol. 1450, Springer, 1998.
[48] N. Pippenger, M.J. Fischer, Relations among complexity measures, J. ACM 26 (1979) 361–381.
[49] M.O. Rabin, D. Scott, Finite automata and their decision problems, IBM J. Res. Dev. 3 (1959) 114–125.
[50] M.O. Rabin, D. Scott, Finite automata and their decision problems, in: Sequential Machines — Selected Papers, Addison-Wesley, 1964, pp. 63–91.
[51] R.W. Ritchie, F.N. Springsteel, Language recognition by marking automata, Inform. Control 20 (1972) 313–330.
[52] A.L. Rosenberg, On multi-head finite automata, IBM J. Res. Dev. 10 (1966) 388–394.
[53] W.J. Sakoda, M. Sipser, Nondeterminism and the size of two way finite automata, in: Symposium on Theory of Computing, STOC 1978, ACM, ACM Press, New York, 1978.
[54] E.M. Schmidt, T.G. Szymanski, Succinctness of descriptions of unambiguous context-free languages, SIAM J. Comput. 6 (1977) 547–553.
[55] C.-P. Schnorr, The network complexity and the Turing machine complexity of finite functions, Acta Inform. 7 (1976) 95–107.
[56] J.C. Shepherdson, The reduction of two-way automata to one-way automata, IBM J. Res. Dev. 3 (1959) 198–200.
[57] M. Sipser, Lower bounds on the size of sweeping automata, J. Comput. System Sci. 21 (1980) 195–202.
[58] R.E. Stearns, A regularity test for pushdown machines, Inform. Control 11 (1967) 323–340.
[59] I.H. Sudborough, Bounded-reversal multihead finite automata languages, Inform. Control 25 (1974) 317–328.
[60] I.H. Sudborough, On tape-bounded complexity classes and multihead finite automata, J. Comput. System Sci. 10 (1975) 62–76.
[61] K. Sugata, H. Umeo, K. Morita, The language accepted by a rebound automaton and its computing ability, Electron. Commun. Japan 60-A (1977) 11–18.
[62] L.G. Valiant, A note on the succinctness of descriptions of deterministic languages, Inform. Control 32 (1976) 139–145.
[63] K. Wagner, G. Wechsung, Computational Complexity, Reidel, Dordrecht, 1986.
[64] A.C. Yao, R.L. Rivest, $k+1$ heads are better than k, J. ACM 25 (1978) 337–340.