Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 330 (2005) 311–324
Theoretical Computer Science
www.elsevier.com/locate/tcs

On the descriptional power of heads, counters, and pebbles

Martin Kutrib

Institut für Informatik, Universität Giessen, Arndtstr. 2, D-35392 Giessen, Germany

Received 17 November 2003; received in revised form 9 April 2004; accepted 23 April 2004

Abstract

We investigate the descriptional complexity of deterministic two-way k-head finite automata (k-DHA). It is shown that between non-deterministic pushdown automata and any k-DHA, $k \geqslant 2$, there are savings in the size of description which cannot be bounded by any recursive function. The same is true for the other end of the hierarchy. Such non-recursive trade-offs are also shown between any k-DHA, $k \geqslant 1$, and $\mathsf{DSPACE}(\log) =$ multi-DHA. We also address the particular case of unary languages. In general, it is possible that non-recursive trade-offs for arbitrary languages reduce to recursive trade-offs for unary languages. Here we present huge lower bounds for the unary trade-offs between non-deterministic finite automata and any k-DHA, $k \geqslant 2$. Furthermore, several known simulation results imply the presented trade-offs for other descriptional systems, e.g., deterministic two-way finite automata with k pebbles or with k linearly bounded counters.

Keywords: Descriptional complexity; State complexity; Non-recursive trade-offs; Finite automata

1. Introduction

Formal languages can 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 non-deterministic finite automata (NFA) can offer exponential savings in size compared

E-mail address: Kutrib@informatik.uni-giessen.de (M. Kutrib).

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.04.013

with deterministic finite automata (DFA). Concerning the number of states, 2^n is a tight bound for the NFA to DFA conversion [13]. Further asymptotic bounds are $O(n^n)$ for the two-way DFA to one-way DFA conversion [13], $2^{O(n^2)}$ for the two-way NFA to one-way DFA conversion [23], $O(\sqrt{2^n})$ for the two-way DFA to one-way NFA conversion [1], and $O(2^{3n})$ for the two-way NFA to one-way NFA conversion [1]. The latter reference is a valuable source for further simulation results.

All trade-offs mentioned with respect to the number of states 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 [30]. In [25] it is proved that the trade-off between unambiguous and non-deterministic pushdown automata is also non-recursive. Recent results involving the parallel model of cellular automata can be found in [10]. In particular, non-recursive trade-offs are shown between DFA and real-time one-way cellular automata (real-time OCA), between pushdown automata and real-time OCA, and between real-time OCA and real-time two-way cellular automata.

A comprehensive survey of descriptional complexity of machines with limited resources in [3], which is a valuable source for further results and references.

Nevertheless, some challenging problems of finite automata are open. An important example is the question of how many states are sufficient and necessary to simulate two-way NFA with two-way DFA. The problem has been raised in [23] and partially solved in [7,9,28].

When certain problems are difficult to resolve in general, a natural question concerns simpler versions. To this regard, promising research has been done for unary languages. It turned out that this particular case is essentially different from the general case. The problem of evaluating the costs of unary automata simulations has been raised in [28]. In [2] it has been shown that the unary NFA to DFA conversion takes $e^{\Theta(\sqrt{n\ln(n)})}$ states, the NFA to two-way DFA conversion has been solved with a bound of $O(n^2)$ states, and the costs of unary two-way to one-way DFA conversion reduces to $e^{\Theta(\sqrt{n\ln(n)})}$. Several more results can be found in [11,12]. Furthermore, in [10] it is shown for real-time OCA that non-recursive trade-offs for arbitrary languages reduce to recursive trade-offs for unary languages.

Here we investigate the descriptional complexity of deterministic two-way k-head finite automata (k-DHA). In particular, we consider the trade-offs between non-deterministic pushdown automata and k-DHA, for any $k \geqslant 2$, and the trade-offs between any k-DHA, $k \geqslant 1$, and the deterministic log-space bounded Turing machines, whose languages are exactly the languages accepted by the union of all k-DHA. All these trade-offs are shown to be non-recursive. For unary languages it is not known whether the trade-offs are recursive or not. Here we present huge lower bounds between non-deterministic finite automata and any k-DHA, $k \geqslant 2$. Furthermore, these lower bounds increase with the number of heads in a nice way. Provided minimality can be shown, these bounds can also serve as lower bounds between k-DHA and $(k+1)$-DHA.

In the next section, we define the basic notions and present a preliminary example. Section 3 is devoted to the study of the mentioned non-recursive trade-offs. Unary languages and the huge lower bounds are considered in Section 4. Finally, in Section 5 the results are adapted to other types of acceptors, e.g., deterministic two-way finite automata with k pebbles or with k linearly bounded counters. Some concerned and related open questions are discussed.

2. Basic notions

We denote the positive integers $\{1, 2, ...\}$ by $\mathbb{N}$ and the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$. For the length of a word w we write $|w|$. We use $\subseteq$ for inclusions and $\subset$ if the inclusions are strict. By $\log : \mathbb{N} \to \mathbb{N}$ we denote the function $\max\{1, \lfloor \log_2 \rfloor\}$.

Let $k \in \mathbb{N}$ 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 in between two endmarkers. The k heads of the automaton can move freely on the tape but not beyond the endmarkers. In Example 2, a two-head finite automaton is sketched that accepts the non-context-free language $\{ww \mid w \in \{a, b\}^+\}$. A formal definition is:

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

1. S *is the finite set of* internal states,
2. A *is the set of* input symbols,
3. $k \in \mathbb{N}$ *is the* number of heads,
4. $\triangleright \notin A$ *and* $\triangleleft \notin A$ *are the* left and right endmarkers,
5. $s_0 \in S$ *is the* initial state,
6. $F \subseteq S$ *is the set of* accepting states,
7. $\delta : S \times (A \cup \{\triangleright, \triangleleft\})^k \to S \times \{-1, 0, 1\}^k$ *is the* partial transition function *which satisfies*: *Whenever* $\delta(s, (a_1, \ldots, a_k))$ *is defined, then* $d_i \in \{0, 1\}$ *if* $a_i = \triangleright$, *and* $d_i \in \{-1, 0\}$ *if* $a_i = \triangleleft$, $1 \leqslant i \leqslant k$.

Let $\mathcal{A} = \langle S, A, k, \delta, \triangleright, \triangleleft, s_0, F\rangle$ be a k-DHA. A *configuration* of $\mathcal{A}$ at some time $t \in \mathbb{N}_0$ is a triple $c_t = (w, s, p)$, where for some $n \in \mathbb{N}$ the word $w = a_1 \cdots a_n \in A^n$ is the input, $s \in S$ is the current state, and $p = (p_1, \ldots, p_k) \in \{0, \ldots, n+1\}^k$ gives the current head positions. If a position p_i is 0, then head i is scanning the symbol $\triangleright$, if it is $n+1$, then the head is scanning the symbol $\triangleleft$, otherwise it is scanning a_{p_i}. The initial configuration for input w is set to $(w, s_0, (0, \ldots, 0))$. During its course of computation, $\mathcal{A}$ runs through a sequence of configurations. One step from a configuration to its successor configuration is denoted by $\vdash$. Let $w = a_1 \cdots a_n$ be the input, $a_0 = \triangleright$, and $a_{n+1} = \triangleleft$, then we set $(w, s, (p_1, \ldots, p_k)) \vdash (w, s', (p_1 + d_1, \ldots, p_k + d_k))$ if and only if $\delta(s, (a_{p_1}, \ldots, a_{p_k})) = (s', (d_1, \ldots, d_k))$. 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.

We say a k-DHA *halts in a configuration* c, if the transition function is not defined for c. If necessary, by adding some new states we can always modify a given k-DHA such that it halts in distinguished states with all heads on the right endmarker. This observation allows together with the deterministic behavior to apply the technique developed in [27]. So, we can force every k-DHA to halt on every input. This immediately implies the closure of $\mathcal{L}(k\text{-DHA})$ under complement and intersection.

The language accepted by k-DHA $\mathcal{A}$ is

$$L(\mathcal{A}) = \{w \in A^* \mid (w, s_0, (0, \ldots, 0)) \vdash^* (w, s, (p_1, \ldots, p_k)), s \in F, \text{ and } \mathcal{A} \text{ halts in } (w, s, (p_1, \ldots, p_k))\}.$$

The family of all languages accepted by k-DHA is denoted by $\mathcal{L}(k\text{-DHA})$.

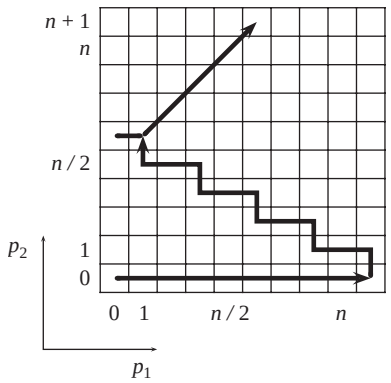

Fig. 1. Trace of the head positions of $\mathcal{A}$.

Example 2. The language $\{ww \mid w \in \{a, b\}^+\}$ is accepted by some 2-DHA $\mathcal{A}$ as follows (cf. Fig. 1).

At first $\mathcal{A}$ moves its first head to position $n+1$. Then it moves the second head to position one. The two head positions can be considered as pair from $\{0, \ldots, n+1\}^2$. So, continuing at position $(p_1, p_2) = (n+1, 1)$, automaton $\mathcal{A}$ moves its first head to n and then to $n-1$. Then its second head is moved to 2. From this position $(n-1, 2)$ the movement repeats. After another three time steps the heads are at position $(n-3, 3)$. After i repetitions the heads are located at $(n+1-2(i-1), i)$. If n is even, then after $i = (n/2)+1$ repetitions the heads are at position $(1, (n/2)+1)$. This situation is recognized by $\mathcal{A}$ during the next step since the first head is moved onto the left endmarker. This next step is the first left move of a repetition. If otherwise n is odd, then the first head is moved onto the left endmarker during the second left move of a repetition. This situation can also be recognized and leads to rejection.

Considering the head positions 1 and $(n/2)+1$, now it is easy for $\mathcal{A}$ to compare the first half of the input with the second half symbolwise, thus, to verify ww by moving the heads to positions $(2, (n/2)+2), \ldots, ((n/2), n)$, and finally to $((n/2)+1, n+1)$, which is recognized by the second head that scans the right endmarker.

3. Non-recursive Trade-offs

On one hand, the 1-head automata accept exactly the regular languages. On the other end of complexity, it is known that the languages accepted by finite automata with an arbitrary number of heads are exactly the languages accepted by deterministic Turing machines with a read-only input tape and a log-space bounded read–write working tape [4,24,26]: $\mathcal{L}(\text{DHA}) = \bigcup_{k\in\mathbb{N}} \mathcal{L}(k\text{-DHA}) = \mathsf{DSPACE}(\log)$. In between it is known that $k+1$ heads are better than k heads [14,15]. So, the computational capacity increases with the number of heads, and for $k \geqslant 2$ all families $\mathcal{L}(k\text{-DHA})$ are proper supersets of the regular languages. Since $\mathsf{DSPACE}(\log)$ is properly contained in $\mathsf{DSPACE}(n)$, each

family $\mathcal{L}(k\text{-DHA})$ is a proper subset of the context-sensitive languages. From Example 2 follows that a non-context-free language belongs to each family $\mathcal{L}(k\text{-DHA})$, $k \geqslant 2$. The general question whether the context-free languages are a subset of some family $\mathcal{L}(k\text{-DHA})$ or even of $\mathcal{L}(\text{DHA})$ is open. The question relates to famous open problems in computational complexity theory as follows. If the answer is positive, then the equality $\mathsf{DSPACE}(\log) = \mathsf{NSPACE}(\log)$ follows, since there is a context-free language which is complete for $\mathsf{NSPACE}(\log)$. If the answer is negative, then there exists a context-free language not belonging to $\mathsf{DSPACE}(\log)$. Since the context-free languages are a subset of P, the complexity classes $\mathsf{DSPACE}(\log)$ and P would be separated.

Apart from computational complexity, natural questions ask for the descriptional capacity of k-DHA. How succinctly can a language be presented by a k-DHA compared with the presentation by a non-deterministic pushdown automaton? How succinctly can it be presented by a log-space bounded Turing machine compared with the presentation by a k-DHA? This section is devoted to these questions. In particular, it is studied whether there exist upper bounds for the increase in complexity when changing from a minimal description of a language by non-deterministic pushdown automata to an equivalent minimal description by k-DHA, and when changing from a minimal description by k-DHA to an equivalent minimal description by a log-space bounded Turing machine. It will turn out that there are no recursive functions serving as upper bounds, the trade-offs are said to be *non-recursive*. In fact, by this result non-recursive trade-offs are obtained for any reasonable complexity measure (e.g., size of description). A non-recursive trade-off will exceed any difference caused by applying two reasonable complexity measures.

Concerning the definitions of descriptional complexity, we follow the presentation in [3]. A *descriptional system* D is a decidable set of finite descriptors for languages, such that each descriptor can effectively be converted into a Turing machine that decides the described language L, if L is recursive, or semidecides L, if L is recursively enumerable. For every L, let

$$\mathrm{D}(L) = \{\mathcal{D} \mid \mathcal{D} \in \mathrm{D} \text{ and } \mathcal{D} \text{ describes } L\}.$$

A total, recursive function $C : \mathrm{D} \to \mathbb{N}_0$ is a *complexity (size) measure* for D, if for any alphabet A the set of descriptors in D describing languages over A (1) is recursively enumerable in order of increasing complexity, and (2) does not contain infinitely many descriptors of the same size.

Let D_1 and D_2 be two descriptional systems, each with descriptors for all languages in some family $\mathcal{L}$, and C be a complexity measure for D_1 and D_2. A function $f : \mathbb{N}_0 \to \mathbb{N}_0$, $f \geqslant id$, is said to be an *upper bound* for the increase in complexity when changing from a minimal description in D_1 to an equivalent minimal description in D_2, if for all $L \in \mathcal{L}$:

$$\min\{C(\mathcal{D}) \mid \mathcal{D} \in \mathrm{D}_2(L)\} \leqslant f(\min\{C(\mathcal{D}) \mid \mathcal{D} \in \mathrm{D}_1(L)\}.$$

If there is no recursive upper bound, then the trade-off is *non-recursive*. The definition of lower bounds is given in Section 4.

A powerful technique for proving non-recursive trade-offs has been developed in [5]. Its idea is generalized in the following theorem.

Theorem 3. *Let* D_1 *and* D_2 *be two descriptional systems for recursive languages. If there exists a descriptional system* D_3 *such that, given an arbitrary* $\mathcal{A} \in \mathrm{D}_3$,

1. *there exists an effective procedure to construct a descriptor in* D_1 *for some language* $L_{\mathcal{A}}$,
2. $L_{\mathcal{A}}$ *has a descriptor in* D_2 *if and only if* $L(\mathcal{A})$ *has not a property* P, *and*
3. *property* P *is not semidecidable for languages with descriptors in* D_3,

then the trade-off between D_1 *and* D_2 *is non-recursive.*

Proof. Assume the trade-off with respect to a complexity measure C is bounded by some recursive function f. Let L be a language with a descriptor $\mathcal{A}_1 \in \mathrm{D}_1$. If L has a descriptor $\mathcal{A}_2$ in D_2, then there exists an $\mathcal{A}_2$ such that $C(\mathcal{A}_2) \leqslant f(C(\mathcal{A}_1))$. Since f and C are recursive, the value $f(C(\mathcal{A}_1))$ can be computed. We can recursively enumerate the finite number of descriptors in D_2 whose size is at most $f(C(\mathcal{A}_1))$. All these descriptors can effectively be converted into Turing machines that decide for any input whether it belongs to the described languages. The same holds for the descriptor $\mathcal{A}_1$. By comparing the enumerated descriptors and $\mathcal{A}_1$ on successive inputs over the alphabet of L, we can detect when none of the descriptors is equivalent to $\mathcal{A}_1$. As a result, a Turing machine is constructed that halts if and only if L has no descriptor in D_2. So, the set

$$R = \{\mathcal{A} \mid \mathcal{A} \in \mathrm{D}_1 \text{ and } L(\mathcal{A}) \text{ has no descriptor in } \mathrm{D}_2\}$$

is recursively enumerable.

Now the theorem follows due to the following contradiction. Assuming that the trade-off is bounded by a recursive function, we show that there exists an effective procedure that semidecides property P for $L(\mathcal{A})$ with a descriptor $\mathcal{A} \in \mathrm{D}_3$. Since there exists an effective procedure to construct the descriptor $\mathcal{A}_1$ in D_1 for the language $L_{\mathcal{A}}$, there exists an effective procedure to semidecide whether $\mathcal{A}_1$ belongs to the set R. If the answer is yes, then $L_{\mathcal{A}}$ has no descriptor in D_2, which implies that $L(\mathcal{A})$ has the property P. □

The goal of the following is to apply this theorem. In order to satisfy the conditions, we have to find the descriptional system D_3 and the languages $L_{\mathcal{A}}$. To this end, we will use at some point the set of *valid computations of Turing machines* [6]. It suffices to consider deterministic Turing machines with one single tape and one single read–write head. Basically, a valid computation is a string built from a sequence of configurations which are passed through during an accepting computation. Let S be the state set of some Turing machine $\mathcal{M}$, where s_0 is the initial state, $T \cap S = \emptyset$ is the tape alphabet containing the blank symbol $\sqcup$, $A \subset T$ is the input alphabet, and $F \subseteq S$ is the set of accepting states. Then a configuration of $\mathcal{M}$ can be written as a word of the form T^*ST^* such that $t_1 \cdots t_i s t_{i+1} \cdots t_n$ is used to express that $\mathcal{M}$ is in state s, scanning tape symbol t_{i+1}, and t_1 to t_n is the support of the tape inscription.

For the purpose of the following, valid computations $\mathrm{VALC}(\mathcal{M})$ are now defined to be the set of words of the form $w_1\$w_2\$\cdots\$w_n$, where $\$ \notin T \cup S$, $w_i \in T^*ST^*$ are configurations of $\mathcal{M}$, w_1 is an initial configuration of the form s_0A^*, w_n is an accepting configuration of the form T^*FT^*, and w_{i+1} is the successor configuration of w_i, $1 \leqslant i \leqslant n$. The set of *invalid*

computations INVALC($\mathcal{M}$) is the complement of VALC($\mathcal{M}$) (with respect to the alphabet $\{\$\} \cup T \cup S$).

Theorem 4. *Let $\mathcal{M}$ be a Turing machine. Then a* 2-DHA $\mathcal{A}$ *can be constructed that accepts* VALC($\mathcal{M}$).

Proof. The first task of $\mathcal{A}$ is to scan the input and to verify its structure. In particular, it has to verify whether it is of the form $w_1\$\cdots\$w_n$, whether each w_i is of the form T^*ST^*, whether w_1 is of the form s_0A^*, and whether w_n is of the form T^*FT^*.

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 $\mathcal{A}$ compares the subwords symbolwise by moving the heads to the right. Turing machine $\mathcal{M}$ has three possibilities to move its head. So, $w_i = t_1\cdots t_i s t_{i+1}\cdots t_n$ goes to $t_1\cdots t_i s' t'_{i+1}\cdots t_n$, to $t_1\cdots s' t_i t'_{i+1}\cdots t_n$, or to $t_1\cdots t_i t'_{i+1} s'\cdots t_n$. Each of the three possibilities can be detected by $\mathcal{A}$. Furthermore, $\mathcal{A}$ can verify whether the differences between w_i and w_{i+1} are due to a possible application of the transition function of $\mathcal{M}$. 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 of the previous theorems are non-recursive trade-offs between the languages accepted by any k-DHA, $k \geqslant 2$, and the context-free languages.

Theorem 5. *Let $k \geqslant 2$ be a natural number. The trade-off between* k-DHA *and nondeterministic pushdown automata is non-recursive.*

Proof. In order to apply Theorem 3, let D_3 be the set of Turing machines. For every $\mathcal{M} \in D_3$, define $L_{\mathcal{M}}$ to be VALC($\mathcal{M}$). So, $L_{\mathcal{M}}$ belongs to $\mathcal{L}(k\text{-DHA})$. Furthermore, it is not semidecidable for Turing machines whether they accept infinite languages [6,10]. Since in [6] it is shown that VALC($\mathcal{M}$) is context free if and only if $L(\mathcal{M})$ is finite, all conditions of Theorem 3 are satisfied and the assertion follows. □

Corollary 6. *Let $k \geqslant 2$ be a natural number. The trade-off between* k-DHA *and* 1-DHA *is non-recursive.*

Another consequence of the fact that the set of valid computations is accepted by k-DHA, $k \geqslant 2$, is that many properties of $\mathcal{L}(k\text{-DHA})$ are not even semidecidable. We can transfer the results from Turing machines to k-DHA. See, e.g., [10] where this has been done for real-time one-way cellular automata.

Lemma 7. *Let $k \geqslant 2$ be a natural number. It is not semidecidable for arbitrary* k-DHA $\mathcal{A}$, $\mathcal{A}'$ *whether $L(\mathcal{A})$ is context free, $L(\mathcal{A})$ is regular, $L(\mathcal{A})$ is finite, $L(\mathcal{A})$ is infinite, $L(\mathcal{A})$ is empty, $L(\mathcal{A}) = L(\mathcal{A}')$, and $L(\mathcal{A}) \subseteq L(\mathcal{A}')$, respectively.*

The proof of the lemma relies on the fact that k-DHA can be constructed that accept VALC($\mathcal{M}$) and INVALC($\mathcal{M}$) for arbitrary Turing machines $\mathcal{M}$. If, for example, the finiteness would be semidecidable for k-DHA, it would be for Turing machines, too. Furthermore, we can exploit the facts that VALC($\mathcal{M}$) is context free if and only if $L(\mathcal{M})$ is finite, and INVALC($\mathcal{M}$) is regular if and only if $L(\mathcal{M})$ is finite.

A further consequence is that there exists no minimization algorithm for k-DHA, $k \geqslant 2$. Now we turn to the non-recursive trade-offs at the upper end of the hierarchy in question.

Theorem 8. *Let $k \geqslant 2$ be a natural number. The trade-off between log-space bounded Turing machines and k-DHA is non-recursive.*

Proof. In order to apply Theorem 3, let D_3 be the set of k-DHA.

From [4,24,26] the proper inclusion $\mathcal{L}(k\text{-DHA}) \subset \mathsf{DSPACE}(\log)$ follows. Let L_1 be a fixed language from $\mathsf{DSPACE}(\log) \setminus \mathcal{L}(k\text{-DHA})$ which is defined over some alphabet A. There exists an acceptor $\mathcal{M}_1$ for L_1. For every k-DHA $\mathcal{A}$ we define

$$L_{\mathcal{A}} = \{w \in A^+ \mid w \in L_1 \vee \exists u \in L(\mathcal{A}) : \ |u| \leqslant \log(|w|)\}.$$

The property P which is not semidecidable for languages in $\mathcal{L}(k\text{-DHA})$ is emptiness.

At first we show $L_{\mathcal{A}} \in \mathcal{L}(k\text{-DHA})$ if and only if $L(\mathcal{A})$ is not empty.
If $L(\mathcal{A}) = \emptyset$, then $L_{\mathcal{A}} = L_1$. This implies $L_{\mathcal{A}} \notin \mathcal{L}(k\text{-DHA})$.
If $L(\mathcal{A}) \neq \emptyset$, then there exists a shortest $u_0 \in L(\mathcal{A})$. This implies that $L_{\mathcal{A}}$ is the union of the regular language $\{w \in A^+ \mid |u_0| \leqslant \log(|w|)\}$ and the finite language $\{w \in L_1 \mid \log(|w|) < |u_0|\}$. Therefore, $L_{\mathcal{A}}$ is regular and belongs to $\mathcal{L}(k\text{-DHA})$.

It remains to be shown that there exists an effective procedure to construct a log-space bounded Turing machine $\mathcal{M}$ that accepts $L_{\mathcal{A}}$. With input w Turing machine $\mathcal{M}$ can successively generate all words u with lengths less than or equal to $\log(|w|)$. For each u it simulates the k-DHA $\mathcal{A}$ (which can be assumed to halt on every input without loss of generality). If some u is accepted by $\mathcal{A}$, then $\mathcal{M}$ accepts w. If none of the u is accepted by $\mathcal{A}$, then $\mathcal{M}$ simulates the acceptor $\mathcal{M}_1$ for L_1 in order to determine whether to accept the input w.

Altogether, we can now apply Theorem 3 and conclude that a recursive trade-off would imply the semidecidability of emptiness for k-DHA. □

4. Lower bounds for unary languages

After investigating some upper bounds in the previous section, now we turn to lower bounds. Since this is sensible only for cases where no non-recursive trade-offs are known, we restrict to unary languages. On one hand, unary languages are often much simpler than general languages such that open problems are solved for the particular unary case. An important example is the open question how many states are sufficient and necessary in the worst case to simulate non-deterministic finite automata with two-way deterministic finite automata. The problem has been raised in [23] and solved for the unary case with a bound of $O(n^2)$ states in [2]. On the other hand, it may happen that non-recursive trade-offs for

general languages reduce to recursive trade-offs for unary languages, e.g., between real-time one-way cellular automata and finite automata [10].

Let D_1 and D_2 be two descriptional systems, each with descriptors for all languages in some family $\mathcal{L}$, and C be a complexity measure for D_1 and D_2. A function $f : \mathbb{N}_0 \to \mathbb{N}_0$, $f \geqslant id$, is said to be a *lower bound* for the increase in complexity when changing from a minimal description in D_1 to an equivalent minimal description in D_2, if for infinitely many $L \in \mathcal{L}$:

$$\min\{C(\mathcal{D}) \mid \mathcal{D} \in D_2(L_i)\} \geqslant f(\min\{C(\mathcal{D}) \mid \mathcal{D} \in D_1(L_i)\}).$$

The rest of the section is devoted to huge unary lower bounds for the trade-off between finite automata with k heads and non-deterministic finite automata. To this end, we present an infinite series of functions f_i, $i \in \mathbb{N}$, whose inverses F_i are used to define an infinite series of regular languages. These languages are witnesses for the lower bounds of the trade-offs.

Let $f : \mathbb{N} \to \mathbb{N}$ be a function. We write $f^{[i]}$, $i \in \mathbb{N}$, for its i-fold composition, which is defined by $f^{[0]} = id$, $f^{[m+1]} = f(f^{[m]})$, $m \geqslant 0$. If $f(n) < n$ for all $n \in \mathbb{N}$, then we define $f^*(n) = \min\{m \in \mathbb{N} \mid f^{[m]}(n) = 1\}$.

Now let $f_1 = id$, $f_2 = \log$, $f_3 = f_2^* = \log^*$, $f_4 = f_3^* = (\log^*)^*$, and in general $f_i = f_{i-1}^* = \underbrace{(((\log^*)^*)\cdots^*)}_{i-2 \text{ times}}$ for $i \geqslant 3$.

Let us mention that even f_3 grows very slowly: $\log^*(2) = 1$, $\log^*(4) = 2$, $\log^*(16) = 3$, $\log^*(65536) = 4$, and $\log^*(2^{65536}) = 5$.

For an increasing function $f : \mathbb{N} \to \mathbb{N}$ the *inverse* is defined by

$$f^{-1}(n) = \min\{m \in \mathbb{N} \mid f(m) \geqslant n\}.$$

We denote the inverses of f_i by F_i. In order to obtain the functions F_i, we observe that the inverse of $f^*(n)$ is $F^{[n]}(1)$, where F denotes f^{-1}, i.e., $f^*(F^{[n]}(1)) = n$.

The witness languages for the lower bounds are as follows. Let $k \in \mathbb{N}$ and $d \in \mathbb{N}$ be constants, then the infinite unary language

$$L_{k,d} = \{a^x \mid x \geqslant F_k(d)\}$$

is regular. Just to give a flavor of the numbers in question, let $\exp(n)$ denote the function 2^n and consider $f_2(n) = \log(n)$, $F_2(n) = \exp(n) = 2^n$, and F_3 and F_4:

$$f_3(n) = \log^*(n),$$

$$F_3(n) = \exp^{[n]}(1) = 2^{\left.2^{\cdot^{\cdot^{2}}}\right\}n},$$

$$f_4(n) = f_3^*(n) = (\log^*)^*(n),$$

$$F_4(n) = ((\log^*)^{-1})^{[n]}(1) = (\exp^{[n]}(1))^{[n]}(1) = \underbrace{2^{\left.2^{\cdot^{\cdot^{2}}}\right\}2^{\left.2^{\cdot^{\cdot^{2}}}\right\}\cdots 2^{\left.2^{\cdot^{\cdot^{2}}}\right\}n}}_{n}}.$$

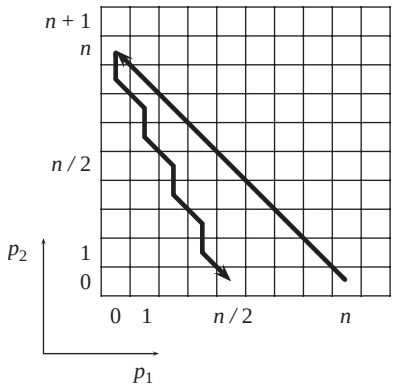

Fig. 2. Trace of the head positions during the division of n by 2.

Since we are dealing with so big numbers, in the following we use a somehow sloppy notation. For readability we omit the big O for the orders of magnitude at most places. The goal is to show the following theorem.

Theorem 9. *Let $k \geqslant 1$ be a natural number. A lower bound for the trade-off between the number of states of unary k-*DHA *and non-deterministic finite automata is given by the function F_k.*

Obviously, a non-deterministic finite automaton needs at least $F_k(d)$ states in order to accept $L_{k,d}$.

The next example shows that two heads can save at least exponentially many states, and thus 1-DHA and 2-DHA are considered as separated from a descriptional complexity point of view [3].

Example 10. We sketch the construction of a 2-DHA $\mathcal{A}$ that accepts the language $L_{2,d}$ with $O(d)$ states. Basically, $\mathcal{A}$ tests whether the logarithm of the length of the input is at least d. To this end, $\mathcal{A}$ successively tries to divide the input length d times by 2. The result of the iterated divisions is given by the position of the first head, respectively. Fig. 2 depicts the trace of the head positions during the first division.

In general, a division of some number q starts with the first head at position $p_1 = q$ and the second head at position $p_1 = 0$. (Before the first division, the first head is moved to the right end.) Then during q time steps the first head is moved to the left and the second head to the right, respectively. Now the heads are located at position $(p_1, p_2) = (0, q)$. Next, automaton $\mathcal{A}$ moves the second head to the left, and subsequently the second head again to the left and simultaneously the first head to the right. From this position $(p_1, p_2) = (1, q-2)$ the movement repeats. After another two time steps the heads are at position $(p_1, p_2) = (2, q-4)$. After $(q/2)$ repetitions the heads are at position $(p_1, p_2) = ((q/2), 0)$, what completes the division.

Clearly, for the iterated phases of head movements during one division some constant number of states are sufficient. In addition, $\mathcal{A}$ has to count the number of performed divisions up to d, which gives $O(d)$ states altogether.

In the following proof of Theorem 9 we generalize the behavior of the 2-DHA of Example 10 to more heads.

Proof of Theorem 9. It has been shown by the previous example that $L_{2,d}$ is accepted by a 2-DHA with $O(d)$ states. The constructed 2-DHA verifies in d phases of head movements whether the logarithm of the input length is at least d. The input values of the phases are given by the positions of a head. Concluding inductively, we now assume that $L_{k,d}$ is accepted by a k-DHA $\mathcal{A}$ with $O(d)$ states, whereby $\mathcal{A}$ verifies in d phases of head movements whether the function f_k applied to the input length yields at least d. The input values of the phases are given by the positions of a head.

A $(k+1)$-DHA $\mathcal{A}'$ for $L_{k+1,d}$ extends $\mathcal{A}$ as follows. At first $\mathcal{A}'$ uses k heads in order to simulate the behavior of $\mathcal{A}$, but instead of performing the phases of head movements at most d times, $\mathcal{A}'$ performs these phases as long as possible. In addition, $\mathcal{A}'$ counts the number of performed phases by the position of its $(k+1)$th head. Therefore, the $(k+1)$th head position gives the result of applying the function f_k to the input length. This completes the first phase of head movements of $\mathcal{A}'$. Now $\mathcal{A}'$ can iterate this phase by using its $(k+1)$th head position as input value of the next phase, simulating $\mathcal{A}$ with its heads 2 to $k+1$ and counting the new number of performed phases of $\mathcal{A}$ by its first head. So, the position of the first head gives the result of applying the function $f_k^{[2]}$ to the input length. Now $\mathcal{A}'$ can try to iterate its behavior d times in order to try to apply the function $f_k^{[d]}$ to the input length. Since $f_{k+1} = f_k^*$, this implies that $\mathcal{A}'$ verifies in d phases of head movements whether the function f_{k+1} applied to the input length is at least d. Moreover, the input values of the phases of $\mathcal{A}'$ are given by the positions of a head, respectively.

For the phases of head movement some constant number of states is sufficient, which depends on k only. Since $\mathcal{A}'$ has to count up to d, we obtain $O(d)$ states. □

From the construction we obtain also upper bounds for the number of states that are necessary for a k'-DHA $\mathcal{A}$ to accept the languages $L_{d,k}$, for $k > k' > 1$. We let $\mathcal{A}$ do clever counting. This means, that $\mathcal{A}$ applies the function $f_{k'-1}$ successively to the input length, and counts the number of application by states. So, $\mathcal{A}$ needs to be able to count up to $f_{k'-1}^*(F_k(d)) = f_{k'}(F_k(d))$, for $k' \geqslant 3$. For $k' = 2$ we obtain immediately $\log(F_k(d)) = f_2(F_k(d))$ states.

5. Other types of acceptors and open questions

We have shown the results of the previous sections in terms of k-head finite automata. Due to a wide range of relations between several types of finite automata, the non-recursive trade-offs in general and the huge lower bounds in the unary case are also true for other types. In fact, we obtain non-recursive trade-offs between a descriptional system D and nondeterministic pushdown automata, if the computational capacity of D is not weaker

than of 2-DHA. For adaption of the non-recursive trade-offs between DSPACE(log) and k-DHA it suffices that the computational capacity of D is not stronger than of k-DHA, for some k. Here we use known results about deterministic two-way finite automata with k pebbles (k-DPA), with k linearly bounded counters (k-DBCA), and with k linearly bounded counters with full-test (k-DFCA). In order to adapt our results, we present the hierarchy

$$\mathcal{L}(k\text{-DHA}) \subseteq \mathcal{L}(k\text{-DPA}) \subseteq \mathcal{L}(k\text{-DBCA}) \subseteq \mathcal{L}(k\text{-DFCA}) \subseteq \mathcal{L}((k+1)\text{-DHA})$$

which has been shown in several famous papers, e.g., [16,18,21,29]. A consequence is

$$\mathcal{L}(\text{DHA}) = \mathcal{L}(\text{DPA}) = \mathcal{L}(\text{DBCA}) = \mathcal{L}(\text{DFCA}) = \text{DSPACE}(\log).$$

Therefore, the non-recursive trade-offs are adapted to the devices in question. The lower bounds in the unary case are adapted by the observations

$$\mathcal{L}(k\text{-DFCA}) = \mathcal{L}((k+1)\text{-DHA}) \subseteq \mathcal{L}((k+1)\text{-DPA}),$$

which are easily verified for unary languages, and that the construction for acceptors of the languages $L_{k,d}$ also works for k-DBCA.

5.1. Some open questions

We have done a few steps towards the exploration of descriptional complexity of deterministic two-way finite automata with additional resources as heads, pebbles or bounded counters. There are still many open problems in that field. Finally, we briefly discuss some of them in terms of k-head finite automata.

- What is the general trade-off between $(k+1)$-DHA and k-DHA?

To apply the technique used so far, one could try to use a language from $\mathcal{L}((k+1)\text{-DHA}) \setminus \mathcal{L}(k\text{-DHA})$ and to bound the accepted words from it by the lengths of $\text{VALC}(\mathcal{M})$. This could be done, e.g., by concatenating the words. Obviously, the resulting language can be accepted by some $(k+1)$-DHA, but it is open to show that it can be accepted if and only if $\mathcal{M}$ does not have some desired property P. Nevertheless, we conjecture that the trade-offs are non-recursive.

- Can we derive unary lower bounds for the trade-off between $(k+1)$-DHA and k-DHA from the presented lower bounds?

If the acceptors for the languages $L_{k,d}$ could be shown to be minimal, then the answer to the question is yes. In that case we would obtain $f_{k_2}(F_{k_1}) = F_{k_1-k_2+2}$ as lower bound between k_1-DHA and k_2-DHA, for $k_1 > k_2 \geqslant 3$.

- What are the upper bounds for the unary case?
  Are they non-recursive as in the general case?
  Are they recursive as is the situation for real-time OCA?
- The next questions concern an in some sense intermediate model. So-called k-dimensional rebound automata (k-DRA) [8,16,19,20,22,29] are finite automata whose input tape is a d-dimensional hypercube. But the input is provided one-dimensional only, all the other tape cells are blank.

For the unary case it holds $\mathcal{L}(k\text{-DHA}) = \mathcal{L}(k\text{-DRA})$. For arbitrary languages $\mathcal{L}(k\text{-DHA}) \supset \mathcal{L}(k\text{-DRA})$ is known. Furthermore, $\mathcal{L}(\text{DRA}) \subset \mathsf{DSPACE}(\log)$. So, natural questions are for the trade-offs between rebound automata and DFA etc. Do we obtain non-recursive trade-offs?

References

[1] J.-C. Birget, State-complexity of finite-state devices, state compressibility and incompressibility, Math. Systems Theory 26 (1993) 237–269.
[2] M. Chrobak, Finite automata and unary languages, Theoret. Comput. Sci. 47 (1986) 149–158.
[3] J. Goldstine, M. Kappes, C.M.R. Kintala, H. Leung, A. Malcher, D. Wotschke, Descriptional complexity of machines with limited resources, J. Univ. Comput. Sci. 8 (2002) 193–234.
[4] J. Hartmanis, On non-determinancy in simple computing devices, Acta Inform. 1 (1972) 336–344.
[5] J. Hartmanis, On the succinctness of different representations of languages, SIAM J. Comput. 9 (1980) 114–120.
[6] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Language, and Computation, Addison-Wesley, Reading, MA, 1979.
[7] J. Hromkovic, G. Schnitger, Nondeterminism versus determinism for two-way finite automata: generalizations of Sipser's separation, Internat. Coll. Automata, Languages and Programming (ICALP 2003), Lecture Notes in Computer Science, vol. 2719, Springer, Berlin, 2003, pp. 439–451.
[8] K. Inoue, I. Takanami, H. Taniguchi, A note on rebound automata, Inform. Sci. 26 (1982) 87–93.
[9] H. Leung, Tight lower bounds on the size of sweeping automata, J. Comput. System Sci. 63 (2001) 384–393.
[10] A. Malcher, Descriptional complexity of cellular automata and decidability questions, J. Autom. Lang. Combin. 7 (2002) 549–560.
[11] C. Mereghetti, G. Pighizzini, Two-way automata simulations and unary languages, J. Autom. Lang. Combin. 5 (2000) 287–300.
[12] C. Mereghetti, G. Pighizzini, Optimal simulations between unary automata, SIAM J. Comput. 30 (2001) 1976–1992.
[13] A.R. Meyer, M.J. Fischer, Economy of description by automata, grammars, and formal systems, in: IEEE Symp. on Switching and Automata Theory, 1971, pp. 188–191.
[14] B. Monien, Transformational methods and their application to complexity problems, Acta Inform. 6 (1976) 95–108.
[15] B. Monien, Corrigenda: transformational methods and their application to complexity problems, Acta Inform. 8 (1977) 383–384.
[16] K. Morita, K. Sugata, H. Umeo, Computation complexity of n-bounded counter automaton and multidimensional rebound automaton, Systems Comput. Controls 8 (1977) 80–87.
[18] H. Petersen, Automata with sensing heads, in: Israel Symposium on the Theory of Computing and Systems, 1995, pp. 150–157.
[19] H. Petersen, Fooling rebound automata, Mathematical Foundations of Computer Science (MFCS 1999), Lecture Notes in Computer Science, vol. 1672, Springer, Berlin, 1999, pp. 241–250.
[20] H. Petersen, Separation results for rebound automata, Mathematical Foundations of Computer Science (MFCS 2000), Lecture Notes in Computer Science, vol. 1893, Springer, Berlin, 2000, pp. 589–598.
[21] R.W. Ritchie, F.N. Springsteel, Language recognition by marking automata, Inform. Control 20 (1972) 313–330.
[22] M. Sakamoto, K. Inoue, I. Takanami, A two-way non-deterministic one-counter language not accepted by nondeterminstic rebound automata, Trans. IEICE E-73 (1990) 879–881.
[23] W. J. Sakoda, M. Sipser, Nondeterminism and the size of two way finite automata, in: ACM Symp. on Theory of Computing (STOC 1978), 1978, pp. 275–286.
[24] W.J. Savitch, A note on multihead automata and context-sensitive languages, Acta Inform. 2 (1973) 249–252.
[25] E.M. Schmidt, T.G. Szymanski, Succinctness of descriptions of unambiguous context-free languages, SIAM J. Comput. 6 (1977) 547–553.

[26] J.I. Seiferas, Relating refined space complexity classes, J. Comput. System Sci. 14 (1977) 100–129.
[27] M. Sipser, Halting space-bounded computations, Theoret. Comput. Sci. 10 (1980) 335–338.
[28] M. Sipser, Lower bounds on the size of sweeping automata, J. Comput. System Sci. 21 (1980) 195–202.
[29] K. Sugata, H. Umeo, K. Morita, The language accepted by a rebound automaton and its computing ability, Electron. Comm. Japan 60-A (1977) 11–18.
[30] L.G. Valiant, A note on the succinctness of descriptions of deterministic languages, Inform. Control 32 (1976) 139–145.