Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Multi-head Automata

Bachelor Thesis

2013

Boris Vida

COMENIUS UNIVERSITY, BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

MULTI-HEAD AUTOMATA

BACHELOR THESIS

Study programme:	Computer Science
Study field:	2508 Computer Science, Informatics
Department:	Department of Computer Science
Supervisor:	Prof. RNDr. Pavol Ďuriš, CSc.

Bratislava, 2013
Boris Vida

20934890

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Boris Vida
Study programme:	Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	9.2.1. Computer Science, Informatics
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Multi-head automata

Aim: The main aim is to elaborate a review on computational power of various types of multi-head automata focused on data-independent ones.
The second aim is to compare computational power of some kinds of automata mentioned above and/or to elaborate alternative proofs for some known results concerning such types of automata.

Supervisor:	prof. RNDr. Pavol Ďuriš, CSc.
Department:	FMFI.KI - Department of Computer Science
Head of department:	doc. RNDr. Daniel Olejár, PhD.
Assigned:	17.10.2012
Approved:	24.10.2012

doc. RNDr. Daniel Olejár, PhD.
Guarantor of Study Programme

....

Student Supervisor

20934890

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Boris Vida
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	9.2.1. informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Mnohohlavové automaty

Cieľ: Hlavným cieľom je vypracovať prehľad výsledkov o výpočtovej sile rôznych typov mnohohlavových automatov so zameraním na tzv. data-independent automaty.
Druhým cieľom je porovnať výpočtovú silu niektorých z vyššie uvedených typov automatov, prípadne vypracovať alternatívlne dôkazy pre niektoré známe výsledky týkajúce sa takýchto typov automatov.

Vedúci:	prof. RNDr. Pavol Ďuriš, CSc.
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	doc. RNDr. Daniel Olejár, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 17.10.2012

Dátum schválenia: 24.10.2012

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

....
študent

....
vedúci práce

Acknowledgement

I would like to express gratitude to my supervisor Prof. RNDr. Pavol Ďuriš, CSc. for his help and advices by writing of this thesis. I would also like to thank Mgr. Marek Zeman, who has initiated my interest in the field of complexity of formal languages.

Abstrakt

V práci sa zaoberáme všeobecnými mnohohlavovými konečnými automatmi a predkladáme stručné zrhnutie doterajších výsledkov v oblasti ich výpočtovej zložitosti, s prihliadnutím na nedeterminizmus, dvojsmerný pohybu a počet hláv. Okrem toho poukazujeme na ekvivalenciu tohto modelu s logaritmicky pamät'ovo ohraničenými Turingovými strojmi a prinášame vzt'ah zložitosi mnohohlavových konečných automatov s inými otázkami v teórii výpočtovej zložitosti formálnych jazykov.

Podobné otázky skúmame aj vo vzt'ahu k dátovo-nezávislým mnohohlavovým konečným automatom, pričom prinášame aj dva vlastné výsledky, ktoré porovnávajú ich výpočtovú silu s inými známymi modelmi, ktorými sú všeobecné mnohohlavové konečné automaty a čiastočne slepé konečné automaty. Taktiež spomíname ekvivalenciu nimi akceptovanej triedy jazykov s triedou $\mathbf{NC}^1$. Navyše prezentujeme aj niekol'ko otvorených problémov týkajúcich sa tohto modelu.

Kľúčové slová: mnohohlavové automaty, výpočtová zložitost', jazykové triedy, dátová nezávislost'

Abstract

In our thesis we concern ourselves with general multihead finite automata and we offer a brief summary of existing results in the area of their computational complexity, with considering of non-determinism, head movement and head count. Beside this, we show the equivalence of this model with logarithmic space bounded Turing machines and we bring the relation of multihead finite automata complexity to another question in the theory of computational complexity of formal languages.

We examine similar questions about data-independent multihead finite automata, whereby we bring two own results, that compare their computational power to another well-known models, namely general multihead finite automata and partially blind finite automata. We also mention the equivalence of the language family accepted by this model to the class $\mathbf{NC}^1$. Moreover, we present few open problems regarding this model, too.

Keywords: multihead automata, computational complexity, language classes, data-independence

Contents

Introduction

The field of complexity of formal languages is one of the most interesting, but also most difficult parts of computer science. On the one hand, it needs strong theoretical and mathematical foundations, because it deals with most abstract computational models, strict formal definitions of algorithms and many non-trivial concepts from combinatorics and discrete mathematics.

On the other hand, there are far-reaching consequences of the theory of complexity in the "real world" applications of computer science. One can surely agree, that it is not insignificant, how long an algorithm e. g. for searching of string in a text file runs, and how much memory it needs. Although this example is oversimplified and any "practical-oriented" programmer could come up with an effective algorithm for this purpose, the theory of complexity gives us useful tools for production of evidence of effectivity and for optimization of computational patterns.

Because of aforementioned (and other) reasons, this field was very "lucrative" for research and many results were accomplished. But despite all effort, there are still few (and quite a bit) open problems, that resist to all attempts to find a solution. We believe, that P - NP problem, so the question, if language classes accepted by deterministic and non-deterministic polynomial time-bounded Turing machines are equivalent, is well-known to the reader.

Similar open problems can be found in parallel computing. Probably the most important, still unresolved question is the relation between NC and P class. Informally, it is the question, if every problem, that can be solved effectively (in polynomial time) on a sequential model, such as Turing machine, can be solved effectively (that is, in poly-logarithmic time with polynomial number of processors) on a parallel model, e. g. Boolean circuit. The formal definition of NC is given in chapter 2.

In this thesis we would like to introduce some aspects and questions of complexity theory related to multihead automata, with emphasis on data-independent ones. Our main aim is to provide a brief review on computational power of these models. We will present several known results, that has been achieved so far. Part of these results will be provided with own alternative proofs.

The second ambition is to bring out some open problems in this field, with their relation to other well known unresolved questions. We will try to present existing progress in the search for answers and with a bit of luck, bring few own results in this area.

The first chapter gives definitions and basic results about multihead automata in general. We examine the language class accepted by this model, relations to other language classes and computational models. We would also like to inspect complexity classes belonging to multihead automata with special restrictions (determinism, number of heads, etc.).

The topic of the second chapter are data-independent multihead automata. After necessary definitions and basic properties, we will present own research regarding this model.

We assume, that the reader is acknowledged with basic concepts of formal languages. If this is not the case, we recommend to obtain this understanding from [1].

We believe, that this thesis will provide the reader with an interesting insight in the topic of multihead automata.

Chapter 1

Multi-head finite automata

1.1 Definitions

Finite state machines (or finite automata) are probably the first and simplest computational model to meet in the formal languages theory. It is the most commonly used computational formalization of regular languages $\mathcal{R}$ and one can easily come up with a constructional proof of its equivalence with type 3 grammar, i. e. grammar with just one non-terminal symbol at the end of each sentential form. A natural extension of finite state machine is adding more reading heads. This is exactly the model called multihead finite automaton. More formally:

Definition 1. A *multihead finite automaton* is a sextuplet $A = (Q, \Sigma \cup \{¢, \$\}, k, \delta, s_0, F)$, where

1. Q is the finite set of states,
2. Σ is the input alphabet; $¢, \$ \notin \Sigma$ is left, respectively right endmarker
3. k is number of heads,
4. δ is a partial transition function $\delta : Q \times (\Sigma \cup \{¢, \$\})^k \to 2^{Q \times \{-1,0,1\}^k}$, where -1 on $(i+1)$-th position means, that i-th head will move one field left, 0 means no movement a 1 stands for one step right. If the input symbol on i-th position is ¢ (resp. \$), i-th head cannot move left (right).
5. $s_0 \in \Sigma$ is initial state,
6. $F \subseteq Q$ is set of accepting states.

For given k, such automaton is called k-head. If $\delta : Q \times (\Sigma \cup \{¢, \$\}) \to (Q \times \{-1, 0, 1\}^k)$ (i. e. for each combination of state and input symbol, there is only one - or none - possible resulting state and head movement), automaton is called deterministic. If the second element of all right sides of δ function is 0 or 1, machine is called one-way.

We abbreviate two-way non-deterministic k-head automaton as $2NFA(k)$. Similarly, we write $2DFA(k)$, $1NFA(k)$ and $1DFA(k)$.

Under *configuration* of 2NFA(k) we understand a (k+2)-plet $C = (w, q, z_1, ..., z_k)$, where $w \in \Sigma^*$ is string written on the input tape, $q \in Q$ is current internal state and $\forall i \in \{1, ..., k\} : z_i$ denotes position of the i-th head, $z_i \in \{0, ..., |w| + 1\}$ ($z_i = 0$ means, that i-th head is on ¢, |w|+1 stands for \$). The starting (or initial) configuration is $(w, s_0, 1, ..., 1)$.

Computational step $\vdash$ is a relation on configurations defined as follows: $(w, q, z_1, ..., z_k) \vdash (w, p, z_1 + d_1, ..., z_k + d_k) \leftrightarrow (p, d_1, ..., d_k) \in \delta(q, a_1, ..., a_k)$ and $\forall i \in \{1, ..., k\}$, symbol on z_i-th position of w is a_i.

The *language* accepted by 2NFA(k) A is set $L(A) = \{w \in \Sigma^* | (w, s_0, 1, ..., 1) \vdash^* (w, q, z_1, ..., z_k), q \in F$ and A halts on $(w, q, z_1, ..., z_k)\}$.

1.2 Language family accepted by multihead finite automata

The first question, that will interest us, is to define language class accepted by multihead finite automata. Following theorem shows the equivalence with another natural model.

Definition 2. By **L** we denote the family of languages accepted by deterministic logarithmic space bounded Turing machines.
By **NL** we denote the family of languages accepted by non-deterministic logarithmic space bounded Turing machines.

Theorem 1. $\mathbf{L} = \bigcup_{k \in \mathbb{N}} \mathcal{L}(2DFA(k))$ and $\mathbf{NL} = \bigcup_{k \in \mathbb{N}} \mathcal{L}(2NFA(k))$

Proof. We will prove only the first part of theorem, i. e. the deterministic case. The reader could surely transform our proof for the latter. Also, we will leave out some technical details, that are not important for the idea of proof.

1. $\mathbf{L} \supseteq \bigcup_{k \in \mathbb{N}} \mathcal{L}(2DFA(k))$

In this part, we will use multitape Turing machines. In [2] it was shown, that multitape machines can be transformed into one-tape with quadratic increase of time complexity, but the same computational power and space complexity.

Let $(w, q, z_1, ..., z_k) \vdash (w, p, z_1 + d_1, ..., z_k + d_k)$ be a computational step of $2NFA(k)$ A. Then $2k$-tape Turing machine M will simulate this step as follows:

(a) The position of i-th head is written in binary on $2i$-th and $(2i - 1)$-th tape of M.

(b) For each head of A, M reads corresponding field of input tape: First, the input tape head of M relocates to the left border of input tape. Then, input tape head travels right, for each step the number on $2i$-th tape is decremented by one. Input tape head stops, when the value written on $2i$-th tape is null. Then it reads symbol on input tape and saves it in the internal state of M (there is a finite number of combinations of k input symbols, so the state set will be finite).

(c) When the input tape for all of k heads is complete, M changes its internal state to p and starts a procedure for changing the positions of heads - M increments the value on $(2i - 1)$-th tape by d_i and copies its value on $2i$-th.

M accepts, if $(w, p, z_1 + d_1, ..., z_k + d_k)$ is an accepting configuration of A.

2. $\mathbf{L} \subseteq \bigcup_{k\in\mathbb{N}} \mathcal{L}(2DFA(k))$

Proof of this inclusion uses principle similar to simulation of Turing machines on a two counter machine - the maximal number written on m fields is k^m, where $k = |\Gamma|$, Γ is the tape alphabet of the Turing machine. Like in the first part of our proof, we will omit exact technical details.

Let $M = (Q, \Sigma, \Gamma, \delta, s_0, F)$ be a Turing machine with one input tape and one working tape (since M is logarithmic bounded, it can use only $\log n$ fields, where n is the length of the input string). We will simulate M on a multihead finite automaton A.

First, we describe the representation of the configuration of M on multihead automaton A: We consider base-l numeral system, which numerals are letters of Γ. Let the number written in this numeral system on the part of working tape from the left endmarker to working tape head be α. Hence α can not be bigger than $l^{\log n}$ (represented in our Γ-numeral system), position of each of k heads of A (in fact, we use just around a half, that is k_1 heads of A) can be seen as a coefficient in polynomial - if we denote the position of i-th head on the input tape of A as r_i, then the value of polynomial $\sum_{j=0}^{k_1} r_j * l^j = \alpha$.

In a similar way, we can remember the number written from input head to the right endmarker β (for some technical reasons, we will look at this number as it was reversed, i. e. with least significant literal on the left), using remaining k_2 heads. The symbol under the input head, as well as current state of M, can be written in the internal state of A.

The computational step of M is simulated as follows:

(a) A reads the symbol under the input tape head of M. This symbol is, as mentioned above, written in the internal state of A.

(b) A changes corresponding ordinate in its internal state, according to its current state, symbol under the input head and δ-function of M.

(c) Now, the input tape head of M can move to the left, to the right, or it can stand still.

- If it stands still, A just changes proper coordinate of its internal state.
- If it moves right, A starts procedure for incrementing the value of α (see above) by moving its heads. In fact, it has to multiplicate α by k and add the value of letter read in this step.

  Next, A has to estimate the symbol under the input head of M in its new position. Now, A starts a similar procedure for division of β. Modulo of this division will be the new symbol to read, while its integral part will be the new number β.

  We believe, that the reader could come up with exact description of this two procedures (having in mind, that k is a constant).
- If the input tape head of M moves left, the process is very similar, besides we have to divide α and multiplicate and add to β.

(d) We have memorized the state of M and symbol under the input tape head in the internal state of A, as well as values on the left and right part of the input tape, hence the computational step is complete and we may continue.

(e) If the state of M, memorized in internal state of A is accepting, automaton A accepts.

Thus, we have shown, that logarithmic space bounded Turing machines are equivalent (in terms of computational power) to multihead finite automata. □

Whether **L** = **NL**, or not, is an open problem. In fact, one of reasons to examine the relationship shown in previous theorem was to separate classes **L** and **NL**.

Although we can not separate deterministic and non-deterministic multihead finite automata for now (and that means, we can not separate **L** and **NL**), some other, more specific relations between this problem and power of non-determinism in multihead finite automata were shown. In [3], following theorem was introduced:

Theorem 2. **L** = **NL** if and only if $\mathcal{L}(2NFA(3)) \subseteq \bigcup_{k\in\mathbb{N}} \mathcal{L}(2DFA(k))$.

This result was further enhanced in [4] to work with one-way two-head non-deterministic multihead finite automata:

Theorem 3. **L** = **NL** if and only if $\mathcal{L}(1NFA(2)) \subseteq \bigcup_{k\in\mathbb{N}} \mathcal{L}(2DFA(k))$.

As we will see further, these results are interesting and non-trivial, because one-way, just like two-way automata provide a strict hierarchy of computational power regarding to number of heads (that means, $\mathcal{L}(2NFA(3)) \subsetneq \bigcup_{k\in\mathbb{N}} \mathcal{L}(2NFA(k))$).

1.3 Relations of subclasses

In this part of our thesis, we would like to present some known results, that provide class hierarchies of multihead finite automata in respect of non-determinism, number of heads and one-way versus two-way movement.

1.3.1 Number of heads

Natural question, when we are handling with multihead finite automata, is, whether number of input heads affects the computational power. And if so, is this sequence infinite, or collapses to some constant number?

It is easy to see, that for one-way deterministic finite automata, two heads are better than one (seeing that $\mathcal{L}(1DFA(1)) = \mathcal{R}$ and non-regular language $L = \{a^n b^n\}$ can be accepted by very simple $1DFA(2)$).

By well-known powerset construction one can show the equivalence of $1DFA(1)$ and $1NFA(1)$. Furthermore, in [5] was presented, that two-way deterministic finite automaton is no stronger than one-way deterministic one and similar result was achieved for non-deterministic automata too.

Using these assumptions, it is clear, that two heads are more than one for every finite automaton, regardless of use of non-determinism or two way movement.

Can be similar result accomplished for any $k \geq 1$? This question was resolved by Yao and Rivest in [6] with following theorem:

Theorem 4. The language $L_b = \{w_1 * w_2 * ... * w_{2b} | (w_i \in \{0,1\}^*) \wedge (w_i = w_{2b+1-i})\, for\ 1 \leq i \leq 2b\}$ is recognizable by a $1NFA(k)$ if and only if $b \leq \binom{k}{2}$.

Proof. We present just the outline of the proof. The implication from right to left was proved by Rosenberg in [7] by full mathematical induction. The correctness of basis of the induction (e. g. for $k = 2$) is easy to see. Now, we want to show, that a word $w_1 * w_2 * ... * w_{2b} \in L_b$.

Let us assume, that $b \leq \binom{k}{2}$. While first head traverses words w_{2b-k+2} to w_{2b}, remaining $k-1$ heads are comparing these substrings to w_{k-1}, w_{k-2}, ..., w_1, respectively. When the comparison is finished, our $k-1$ heads can be positioned at beginning of w_k (first head is now at the right endmarker, so we can not use it anymore) and used to detect, if $w_k * w_{k+1} * ... * w_{2b-k+1} \in L_{b-k+1}$. Taking into account, that $b \leq \binom{k}{2}$, we can see, that $b-k+1 \leq \frac{k(k-1)}{2} - k + 1 = \frac{k^2}{2} - \frac{3}{2}k + 1 \leq \binom{k-1}{2}$, ergo L_{b-k+1} can be accepted with $k-1$ heads by induction hypothesis.

The proof of the other implication can be achieved by a contradiction. We define a new language $L_b^n = \{w_1 * w_2 * ... * w_{2b} | (w_i \in \{0,1\}^n) \wedge (w_i = w_{2b+1-i})\, for\ 1 \leq i \leq 2b\}$. Note, that $L_b^n \subseteq L_b$, substrings w_i have some fixed length n. Let A be a $1NFA(k)$, that accepts language L_b^n (for some large constant n). Yao and Rivest have shown, that A has then to accept some word $w \notin L_b$.

Now, the main notion of the proof is, that for $w = w_1 * w_2 * ... * w_{2b} \in L_b^n$ and $b > \binom{k}{2}$, there always exists such an index i, that subwords w_i and w_{2b-i+q} are never scanned simultaneously. If a pair of heads reads w_{i_0} and w_{2b-i_0+1} at the same time, it can not read any such pair for $i_1 \neq i_0$ (because the automaton is one-way). Thus, we can cover only $\binom{k}{2}$ of substring pairs and since $b > \binom{k}{2}$, such an index i exists.

The proof follows with showing, that if $x = x_1 * x_2 * ... * x_i * ... * x_{2b-i+1} * ... * x_{2b}$ and $y = y_1 * y_2 * ... * y_i * ... * y_{2b-i+1} * ... * y_{2b}$ are two distinct words in L_b^n with similar heads movement, then A accepts word $x_1 * x_2 * ... * x_i * ... * x_{2b-i} * y_{2b-i+1} * x_{2b-i+2} * ... * x_{2b} \notin L_b$. The contradiction now follows. □

Seeing that the procedure for accepting L_b is deterministic and proof of the second part holds for deterministic, as for non-deterministic automata, we may state following corollary:

Corollary 4.1. Let $k \geq 1$. Then

1. $\mathcal{L}(1DFA(k)) \subsetneq \mathcal{L}(1DFA(k+1))$
2. $\mathcal{L}(1NFA(k)) \subsetneq \mathcal{L}(1NFA(k+1))$

The same question for two-way automata was resolved few years later in [8], where following theorem was presented:

Theorem 5.

1. $\mathcal{L}(2DFA(k)) \subsetneq \mathcal{L}(2DFA(k+1))$
2. $\mathcal{L}(2NFA(k)) \subsetneq \mathcal{L}(2NFA(k+1))$

The proof of this theorem uses the characterization of logarithmic space shown in section 1.2, but it's precise form goes beyond the content of our thesis. We note, that the witness languages are unary (over a one-letter alphabet) - this fact will be interesting in chapter 2.

1.3.2 Determinism and non-determinism

As we have mentioned before, by one-headed finite automata non-determinism is no stronger than determinism. But in case of multihead finite automata, standard powerset construction can not be used, because the non-determinism concerns not only changes of state, but also the heads movement. Does this mean, that non-deterministic multihead finite automata have more computational power? The answer for one-way case was also brought by Yao and Rivest in [6]. We will show two of their theorems, but first, we induce a helpful lemma:

Lemma 6. $\mathcal{L}(1DFA(k))$ is closed under complementation.

Proof. The proof of this lemma is the same as for one-head deterministic automata, i. e. since the computation of automaton A on a word w is deterministic (ergo just one) and it ends, when A halts, we just invert the set of accepting states. □

Now, first theorem about the relation of non-deterministic and deterministic automata, with proof:

Theorem 7. For every $k \geq 1$, there is a language M_k recognized by $1NFA(2)$ but by no $1DFA(k)$.

Proof. Let M_k be the complement of L_b from theorem 4, where $b = \binom{k}{2}$. By this theorem and lemma 6, it is not recognized by any $1DFA(k)$. Automaton $1NFA(2)$, that accepts M_k, can guess the matched pair w_i, w_{2b-i+1}, which is not equal and then verify its inequality. □

The second result is more general, but once again, we provide just the main idea of the proof:

Theorem 8. The language $L = \{w_1 * w_2 * ... * w_{2b} | (\forall i, 1 \leq i \leq 2b)((w_i \in \{0,1\}^* \# \{0,1\}^*) \wedge [(\exists j,k)(w_j = x\#y \wedge w_k = x\#z \wedge y \neq z)] \mathit{for\ any}\ \ b \geq 1\}$ can be recognized by $1NFA(2)$ but by no $1DFA(k)$, for any k.

The language can be seen as a dictionary with $2b$ entries in form of *key#value*. Word w is in language L, if and only if there are two words with the same key, but different value.

Proof. The technique of the proof is very similar to proof of Theorem 4. We create a subset $\bar{L} \subseteq L$, such that the key of each w_i is binary representation of $\min(i, 2b - i + 1)$. Now, it is possible to show, that if an $1DFA(k)$ automaton A accepts every word in $\bar{L}$, then it accepts some word $w \notin L$. □

So, as we could see, for one-way case the following corollary can be derived:

Corollary 8.1. For every $k \geq 2$, $\mathcal{L}(1DFA(k)) \subsetneq \mathcal{L}(1NFA(k))$

And what is the situation by two-way multihead finite automata? In section 1.2 we have seen, that the relation of $\mathcal{L}(2DFA(k))$ and $\mathcal{L}(2NFA(k))$, thus the question, if $\mathbf{L} = \mathbf{NL}$, is still open.

1.3.3 One-way and two-way movement

The last question, with that we will deal in this chapter, is, whether two-way automata are computationally stronger than one-way. Once again, we have already mentioned, that for one-head automata, two way movement does not give any advantage (the proof of this claim uses the idea of transitional sequences [5]).

In multihead finite automata, the situation is different. It is easy to see, that the mirror language $L = \{w | w \in \{a,b\}^* \wedge w = w^R\}$ is not recognized by any $1NFA(k)$. However, it is accepted by trivial $2DFA(2)$ - the first head reads a word w from left to right, while second head reads it from right to left.

From this simple train of thought, following corollary can be derived:

Theorem 9. Let $k \geq 2$. Then

1. $\mathcal{L}(1DFA(k)) \subsetneq \mathcal{L}(2DFA(k))$
2. $\mathcal{L}(1NFA(k)) \subsetneq \mathcal{L}(2NFA(k))$

With this theorem, we end our little insight in the world of multihead finite automata in general. The next chapter concerns with a special class of this model, namely data-independent multihead finite automata.

Chapter 2

Data-independence

2.1 Definitions

In this chapter, we will occupy with the notion of data-independence, in relation to multihead automata. This model was first introduced in [9], although the concept of data-independence (also called obliviousness) was examined before, e. g. on Turing machines. In this part of our thesis, we would like to digest current knowledge and present two own results concerning this area.

Definition 3. Let A be a multihead finite automaton. If the position of the i-th input head after t steps of computation on a word w depends only on i, t and $|w|$, A is called a *data-independent (oblivious) multihead finite automaton*.

We abbreviate a two-way non-deterministic k-head data-independent finite automaton as $2DiNFA(k)$. The language class accepted by this automaton is denoted as $\mathcal{L}(2DiNFA(k))$. In a similar way, we will use abbreviations $1DiNFA(k)$ (for one-way non-deterministic automata), $1DiDFA(k)$ (one-way deterministic), $2DiDFA(k)$ (two-way deterministic) etc.

The configuration, computational step and language accepted by a data-independent multihead finite automaton are defined in the same way, as for the data-dependent ones.

As one can see, data-independence is a semantic restriction, which bounds the movement of input heads of an automaton - the head movement has to be identical for all words of the same length (that means, for words not belonging to accepted language, too).

Having this in mind, it is easy to see, that every deterministic multihead finite automaton over a one letter alphabet is data-independent, since there is only one word for each given length. Ergo, we can state the following lemma:

Lemma 10. Let L be any unary language in **L**. Then $L \in \bigcup_{k\in\mathbb{N}} \mathcal{L}(2DiDFA(k))$

However, we do not know, if this is true also for the non-deterministic case, because a non-deterministic multihead finite automaton can have more various computations with different head movement on a single input. Thus, it does not fulfill the definition of data-independence. The relation between classes unary-**NL** and $\mathcal{L}(2DiNFA(k))$ is still an open problem.

2.2 Language class accepted by data-independent multihead finite automata

As we have seen, general multihead finite automata characterize a very well known language class **L** (resp. **NL**). Is the case similar with data-independent automata, too? The answer is, yes. Data-independent multihead finite automata accept a class, that is quite important in word of parallel computing, namely $\mathbf{NC}^1$, the class of problems, that can be solved efficient on parallel computing devices (such as Boolean circuits, alternating Turing machines etc.)

Definition 4. By $\mathbf{NC}^1$ we denote the language class accepted by uniform Boolean circuits with polynomial size and logarithmic depth.

The proof of the equivalence of $\mathbf{NC}^1$ and $\bigcup_{k\in\mathbb{N}} \mathcal{L}(2DiNFA(k))$ was presented in [9], using another model, namely log-space uniform leveled branching program families of constant width and polynomial depth. As shown by Barrington in [10], this model is equivalent to aforementioned Boolean circuits.

Definition 5. A *branching program* is a finite directed acyclic graph with one source node and few sink nodes. Non-sink nodes are called testing nodes. A sink node can be accepting or rejecting. Each of test nodes is labelled with one input variable x_i and has two output edges marked 0 and 1. An input $x \in \{0, 1\}^*$ is accepted iff the computation starting in source node and then traversing test nodes (following output edge with the value of x_i) ends in an accepting sink node.

A *level* is a set of nodes with equal distance from the source node.

Width of the branching program is maximal number of nodes in one level, its *depth* is maximal number of levels.

The *family* of branching programs is a sequence $\{B_n\}$, where $\forall i$, B_i is a branching program, that operates on inputs of length i.

The family of branching programs $\{B_n\}$ is called *log-space uniform*, if there is a log-space bounded Touring machines, that can compute the code of the branching program B_i from input 1^i for all i.

Theorem 11. $\mathbf{NC}^1 = \bigcup_{k\in\mathbb{N}} \mathcal{L}(2DiNFA(k))$

Proof. The proof of this theorem ([9]) is constructional and we present just outline of the simulation of a data-independent multihead finite automaton on a branching program. Let A be a $2DiNFA(k)$. Since A is data-independent, the positions of input heads of A depends only on length of the input, ergo the levels of the branching program B can be seen as computational steps of A. Now, we know, which fields of the input tape will be read in step t, so in B, we test corresponding input nodes in level t. The width of this program is constant, because there is only constant number of input combinations. Since A has polynomial number of configurations, we can reduce the length of computation to polynomial time (if the computation runs longer, it will be circular). For these reasons, B has only polynomial depth. □

The natural question comes out, if the restriction of data-independence has any effect on the computational power. And if so, can we compensate it by adding more input heads, non-determinism, etc.? We will occupy with this question later, when we become familiar with basic concepts of data-independent multihead finite automata.

2.3 Relation of subclasses

Just like in the first chapter, we will occupy with class hierarchies and explore, if also for data-independent multihead finite automata, non-determinism is stronger than determinism, two-way movement than one-way and if more heads brings greater computational power.

2.3.1 Determinism and non-determinism

We have seen, that for general multihead finite automata, there is a language, that is accepted by a two-head one-way non-deterministic automaton, but by no k-head one-way deterministic automaton. Moreover, we have stated, that this question for two-way setting is still open and has important place in the theory of complexity.

The problem for determinism was, that it could not simulate the possibility of differences in head movement. However, data-independent multihead finite automata, even the non-deterministic ones, have the head movement fixed for all words of the same length - that means, that also for different computations on the same input w, the head movement has to be identical.

Now, it is easy to see, that the following theorem holds:

Theorem 12. For every $k \in \mathbb{N}$:

1. $\mathcal{L}(1DiDFA(k)) = \mathcal{L}(1DiNFA(k))$
2. $\mathcal{L}(2DiDFA(k)) = \mathcal{L}(2DiNFA(k))$

Proof. Since the only power of non-determinism is in the choice of the next state, classic powerset construction, well known from one-head finite automata, can be used for one-way, as for two-way setting. □

So, the situation with data-independent multihead finite automata is (in this concern) simpler and in our latter sections, we will not distinguish between deterministic and non-deterministic automata. Also, in proofs of other theorems, we will consider deterministic automata only, unless otherwise stated.

2.3.2 One-way and two-way movement

We have seen, that non-determinism does not give any more computational power for data-independent multihead finite automata, which was quite unexpected, since the situation is different than with data-dependent ones. However, such a surprising result does not awaits us, when occupying with one-way versus two-way head movement. In [9], following relations were shown:

Theorem 13. Let $k \geq 2$. Then $\mathcal{L}(1DiDFA(k)) \subsetneq \mathcal{L}(2DiDFA(k))$.

Moreover, $\bigcup_{k\in\mathbb{N}} \mathcal{L}(1DiDFA(k)) \subsetneq \bigcup_{k\in\mathbb{N}} \mathcal{L}(2DiNFA(k))$.

Proof. To prove this inclusions, we will once again use the mirror language $L = \{w | w \in \{a, b\}^* \wedge w = w^R\}$. As stated before, this language cannot be accepted by any one-way multihead automaton, but is easy to construct a appropriate two-way data-independent automaton with two heads - the first head travels to the right endmarker and then, the comparison starts - the first head traverses the word from right to left, while the second goes from left to right and the automaton compares the equality of corresponding input symbols. □

The same relation obviously holds also for non-deterministic data-independent automata, since they are equivalent with deterministic ones.

2.3.3 Number of heads

Now we would like to inspect the hierarchies of data-independent multihead finite automata in relation to number of input heads. We have seen by data-dependent automata, that $k + 1$ heads do give us more computational power than k heads. Moreover, in two-way deterministic setting, the witness language was unary, as stated by Theorem 5. Since every multihead deterministic automaton is data-independent by Lemma 10, we can derive following result:

Theorem 14. For every $k \in \mathbb{N}$:

1. $\mathcal{L}(2DiDFA(k)) \subsetneq \mathcal{L}(2DiDFA(k + 1))$
2. $\mathcal{L}(2DiNFA(k)) \subsetneq \mathcal{L}(2DiNFA(k + 1))$

And what is the situation like with one-way automata? The automaton for language used in [6], which shows the head hierarchy of one-way multihead finite automata (deterministic as well as non-deterministic), is not data-independent, so it can not be used for our proof.

It turned out, that this question is not easy to solve and the problem of head hierarchy of one-way data-independent multihead automata is probably the most important open problem in this area. Although, some partial solutions have been found.

It is easy to see, that the language class accepted by one-head data-independent automata is exactly the class of regular languages, since every one-head deterministic one-way finite automaton is data-independent. Clearly, there is an one-way two-head data-independent finite automaton accepting non-regular language $L = \{0^n 1 0^n | n \geq 0\}$, therefore $\mathcal{L}(1DiDFA(1)) \subsetneq \mathcal{L}(1DiDFA(2))$.

Next few levels were separated in [11], where following theorem was proved:

Theorem 15. $\mathcal{L}(1DiDFA(2)) \subsetneq \mathcal{L}(1DiDFA(3)) \subsetneq \mathcal{L}(1DiDFA(4))$.

The breakthrough occurred in [13], where following bound was estimated:

Theorem 16. Let $k \geq 1$. Then $\mathcal{L}(1DiDFA(k)) \subsetneq \mathcal{L}(1DiDFA(\frac{k.(k+1)}{2} + 4))$.

Proof. The witness language is a modification of language from Theorem 4, so that the subwords w_i have fixed length. Formally, let $L_b^n = \{w_1 * w_2 * ... * w_{2b} | (w_i \in \{0,1\}^n) \wedge (w_i = w_{2b+1-i}) \, for \; 1 \leq i \leq 2b\}$. This modification was also used in proof of Theorem 4, where it was also shown, that L_b^n can be accepted by an one-way k-head finite automaton if and only if $b \leq \frac{k.(k+1)}{2}$.

Therefore, there is no one-way k-head data-independent automaton, which accepts language L_b^n for $b = \binom{k}{2}+1$. Now, we would like to show the working of one-way $(\frac{k.(k+1)}{2}+4)$-head data-independent automaton A for this language.

Similarly to proof of Theorem 4, first head traverses the second half of input, i. e. words $w_{\frac{k.(k+1)}{2}+2}$, $w_{\frac{k.(k+1)}{2}+2}$, etc. to $w_{k.(k+1)+2}$. Next $\frac{k.(k+1)}{2} + 1$ heads read corresponding substrings w_1 to $w_{\frac{k.(k+1)}{2}+1}$ in the first half on the input. Since A is data-independent, the positioning of heads to start of the substrings cannot be done by looking for symbols *, so we will need one additional "positional" head - then, the positioning of heads will be achieved by sending them with different speeds and stopping, when the "positional" head finds the right endmarker (we believe, that this technique is well known to the reader).

Now, the comparison of the corresponding substrings starts. First, $w_{\frac{k.(k+1)}{2}+1}$ is compared to $w_{\frac{k.(k+1)}{2}+2}$. For correct position of the input head on the second half, one additional head is required. When the comparison is finished, the head from $w_{\frac{k.(k+1)}{2}+1}$ is used to reposition the second head on the next separator. Then, words $w_{\frac{k.(k+1)}{2}}$ and $w_{\frac{k.(k+1)}{2}+3}$ are compared and so on.

Thus, we have shown, that for $b = \binom{k}{2} + 1$, language $L_b^n \in \mathcal{L}(1DiDFA(\frac{k.(k+1)}{2} + 4)) - \mathcal{L}(1DiDFA(k))$ and therefore our theorem is proven. □

This result was later improved in [12], where the bound was enhanced from circa-quadratic to approximately linear growth of input heads. To be specific, following result was introduced, in our thesis, we present it without proof:

Theorem 17. Let $k \geq 1$. Then $\mathcal{L}(1DiDFA(\sqrt{2}.k)) \subsetneq \mathcal{L}(1DiDFA(2k+2))$.

So, we have inspected the hierarchies of subclasses of data-independent multihead finite automata and we also believe, that the reader was given a basic insight in this model. In the next section, we would like to present two own results in this area.

2.4 Relation to other models

Now its time to resolve the question, if the restriction of data-independence has any influence on computational power. Although the natural intuition is, that data-dependent multihead automata should be computationally stronger, this assumption has not been confirmed by any formal proof.

Actually this question is closely related to a very important complexity theory question - the relation of classes $\mathbf{NC}^1$ and $\mathbf{L}$. Profiting from aforementioned simulations of $\mathbf{L}$ and $\mathbf{NC}^1$ on data-dependent and data-independent multihead automata, respectively, it is easy to see, that $\mathbf{NC}^1 \subseteq \mathbf{L}$. If the inclusion is proper, is an open problem, but since in [14] it was shown, that there is a $\mathbf{L}$-complete language in $\mathcal{L}(1DFA(2))$ and a $\mathbf{NL}$-complete language in $\mathcal{L}(1NFA(2))$, Holzer in [9] has presented following:

Theorem 18.

1. $\mathbf{NC}^1 = \mathbf{L}$ if and only if $\mathcal{L}(1DFA(2)) \subseteq \mathcal{L}(2DiDFA(k))$
2. $\mathbf{NC}^1 = \mathbf{NL}$ if and only if $\mathcal{L}(1NFA(2)) \subseteq \mathcal{L}(2DiDFA(k))$

Although this question was not solved yet in general, for one-way setting, it turned out, that data-dependent multihead automata really have more computational power than data-independent ones, even when restricted to determinism and two heads.

Theorem 19. $\mathcal{L}(1DFA(2)) - \bigcup_{k\in\mathbb{N}} \mathcal{L}(1DiDFA(k)) \neq \emptyset$.

Proof. As a witness language, we use $L = \{w\#^+w\#^+ | w \in \{a,b\}^+\}$.

It is easy to see, that this language can be accepted by an $1DFA(2)$ - the first head looks for the second occurrence of the string w, while the second waits at the beginning of the input. Then, the words are compared.

The remaining task is to show, that L cannot be accepted by $1DiDFA(k)$ for any k. To accomplish this, we use similar technique as in the proof of Theorem 4. Ergo, we assume, that there is a $1DiDFA(k)$ for L and this assumption will lead to a contradiction.

Let $A = (Q, \Sigma \cup \{¢, \$\}, k, \delta, q_0, F)$ be a k-head deterministic data-independent finite automaton, which accepts L. Let p be an integer sufficiently large in relation to $|Q|$ (later we will see, why is this important). Let $h = 4.k^2 + 1$. Now, let L_h be a subset of L, in which the length of substrings w is $h.p$ and total number of separators # is a multiple of $2.h.p$, whereby they are separated in multiples of p. Formally, $L_h = \{w\#^{i.p}w\#^{j.p} | w \in \{a, b\}^{h.p} \wedge i + j = 2.h\}$. Since A accepts L, it has to accepted every word in L_h.

Now, we divide the input x in blocks of p characters. This way, we can write the input x as $x_1x_2...x_h\#^{i.p}x'_1x'_2...x'_h\#^{j.p} \cong u_1u_2...u_{4.h}$. We state, that for every input from L_h, there always is a pair (x_s, x'_s), which is never scanned simultaneously during the computation. We argue as follows:

We say, that a couple of blocks (u_a, u_b) is *covered* by a pair of heads, if there is a point of the computation, in which one of the heads reads the block u_a and at the same time, the second head scans the block u_b.

We want to determine the number of couples (u_a, u_b), that can be covered by one pair of input heads. The input x consists of p blocks. We denote the sequence of covered block pairs by $(i_1, j_1), (i_2, j_2), ..., (i_l, j_l)$. Since for every m it holds, that $i_m \leq i_{m+1} \wedge j_m \leq j_{m+1} \wedge \neg(i_m = i_{m+1} \wedge j_m = j_{m+1})$, the number of couples (u_a, u_b) covered by one pair of heads is $8.h - 1$. We have $\binom{k}{2}$ pairs of heads and therefore, k-head finite automaton can in one computation cover less than $k^2.8.h$ couples (u_a, u_b).

Now, the question is, how much possible relative positions can a pair (x_s, x'_s) have. Since the number of blocks of # is $i + j$, for fixed x_t, there are $i + j = 2.h$ possible positions of x'_t. For h different values of s, we have $h.2.h$ possible relative positions of a couple (x_s, x'_s). As we have stated before, $h = 4.k^2 + 1$, so the claim, that there always is a pair (x_s, x'_s) never scanned simultaneously is proven right. Since A is data-independent, the couple is the same in every word with the same structure (under structure, we understand integers i, j).

Now, we define the subclasses of words from L_h according to the internal state of automaton A, when the scanning of subword x'_s is started. Since the language L_h is "dense" (there are 2^{ph} words of length $4.p.h$), there surely are two words $y = y_1...y_h\#^{i.p}y'_1...y'_h\#^{j.p}$, $z = z_1...z_h\#^{i.p}z'_1...z'_h\#^{j.p}$, which belong to the same class (this can be proven using Dirichlet's principle). This means that automaton A cannot know, if it scans input y or z. Since they are both accepted by A, we may construct an input $y_1...y_{s-1}z_sy_{s+1}...y_h\#^{i.p}y'_1...y'_{s-1}y'_sy'_{s+1}...y'_h$, which is also accepted by A, although $v \notin L$.

We have completed the contradiction and therefore, L cannot be accepted by $1DiDFA(k)$ for any k. □

So, the relation for one-way multihead finite automata is can be summed up as follows:

Corollary 19.1. $\bigcup_{k\in\mathbb{N}} \mathcal{L}(1DiDFA(k)) \subsetneq \bigcup_{k\in\mathbb{N}} \mathcal{L}(1DFA(k))$.

Finally, we would like to compare data-independent multihead finite automata to one more model, which is a partially blind finite automaton. We bring just one result for one-way two-head case to give a quick insight in this relation. We denote the family of languages accepted by a one-way deterministic finite automaton with one blind and one normal input head by $\mathcal{L}(1DPBFA(2))$.

Theorem 20. $\mathcal{L}(1DPBFA(2)) - \mathcal{L}(1DiDFA(2)) \neq \emptyset$.

Proof. We prove this theorem using language $L = \{a^ib^jc^i | i, j \geq 0\}$. Note, that this language can also be accepted by a finite counter automaton (a pushdown automaton with unary working alphabet) and therefore, it is context-free.

The partially blind finite automaton works as follows: while the normal head reads symbols a, the blind one moves with double speed. Then, when the sighted head finds first symbol b, the blind head changes its speed to normal (i. e. the speed of sighted head). The blind head should find the right endmarker at the same time, when the normal head arrives at first symbol c. Then, the normal head finishes checking the structure of the input word.

Now, we want to show, that L cannot be accepted by any two-head one-way data-independent deterministic finite automaton. Once again, we prove this by contradiction.

Let A be $1DiDFA(2)$, which accepts language L. First, we would like to analyze the head movement on a word of length n, for n sufficiently large in relation to $|Q|$ (Q is the state set of automaton A). To accomplish this, let us look at a computation on input a^n. Since A is data-independent, the head movement on all other inputs of length n will be the same.

Since A is deterministic, A on a^n enters a loop P during the first $|Q|$ steps of the computation, and A remains in that loop till one of the heads reaches the end of the input. There are, in facts, just two possibilities.

1. Just one of the heads moves in the loop P. Now, replace the input a^n with the word $a^i b^j c^i$, where $2.i + j = n$.

   As we have stated before, after maximum $|Q|$ steps of computation, automaton A works in a loop P, while the second heads stands still. The movement of the second head starts, when the first one has reached the end of the input.

   But, such a computation can be simulated by a proper one-head two-way automaton B, which works as follows: B simulates the "moving head" of A (so the one, that moves in cycle P). Since the second head of A does not move in this part of the computation, the simulation is correct. When the "moving head" reaches the right endmarker, B stops the simulation of the loop P and its reading head travels to the left end of the input string. Then, the simulation of the second head of A is done (with corresponding state changes). Once again, only one head of A moves in this part of the computation, so also this simulation is proper.

   As we have stated in Subsection 1.3.3, one-head two-way automata recognize exactly the class $\mathcal{R}$. Hovewer, the language L clearly is not regular, and therefore it cannot be accepted by automaton B and consequently, nor it can be accepted by automaton A with such a head movement pattern.

2. The second possibility is, that both heads are moving in the loop P. Again, what does it mean on an input word $a^i b^j c^i$, where $2.i + j = n$?

   Consider the input $a^i b^j c^i$ with $i = |Q|$ and $j = |Q|^2$. Now, we claim, that there is a point t of the computation, where both heads are scanning symbols b.

   By h we denote the "slower" head, by g the "faster" one. Let t be the first step of computation, when g reads the rightmost character of the prefix $a^i b^j$ (i. e. $t \geq i + j = |Q| + |Q|^2$) and h also reads the prefix $a^i b^j$. We state, that also h reads the part of the input with symbols b, since both heads move in the cycle P of maximum length $|Q|$ and therefore, both heads have to make at least 1 step right every $|Q|$ steps of computation (every loop iteration). We remind, that the head movemnt on input words a^n and $a^ib^jc^i$ is identical, so it has to be the same in first t steps of computation, too.

Now, for $k = 1, 2, ..., |Q| + 1$, let s_k denote the internal state of automaton A after t steps of computation on the input $a^{i-k}b^{j+2k}c^{i-k}$. Clearly, according to Dirichlet's principle, ther are two unequal indexes m, l, for which $s_m = s_l$. This means, that after t steps of computation, the internal state of A (and also the head positions, since A is data-independent) will be the same for input strings $w_1 = a^{i-m}b^{j+2m}c^{i-m}$ and $w_2 = a^{i-l}b^{j+2l}c^{i-n}$. Hovewer, this also means, that automaton A will be in the state s_m on the input $w_3 = a^{i-m}b^{j+m+l}c^{i-l}$ - and also the head position will be the same. Finally, the computation will continue in the same way as on input w_2, ergo it will be accepting. Nevertheless, $w_3 \notin L$ and the contradiction follows.

As we have seen, L cannot be accepted by any $1DiDFA(2)$, but is accepted by a quite simple $1DPBFA(2)$. $\square$

We believe, that this two theorems have shown a bit about the restriction of data-independence on multihead finite automata.

Conclusion

In our thesis, we were concerned with multihead finite automata, with emphasis on the subclass of data-independent multihead finite automata. Our main aim was to present a brief summary of their computational power and the computational complexity in relation to determinism, one-way or two-way movement and number of input heads.

The first chapter digests known results about multihead finite automata in general, while the main result, namely the equivalence with logarithmic space bounded Turing machines, was provided with our own proof. We have also presented few other theorems from various authors, some of them were provided with outlines of their proofs. We believe, that this chapter has given some basic insight in the computational power of multihead finite automata.

Also in the first chapter, there were mentioned few open problems concerning the computational complexity of multihead finite automata, probably the most important one was the relation of classes **L** and **NL**. Similar question in this area, which has not been resolved so far, is the separation of classes **L** and **P**, respectively **NL** and **P**.

The second chapter presents current state of the research in the area of data-independent multihead finite automata. Again, we have investigated determinism, head movement and number of heads from the complexity point of view. Moreover, we have completed our intention to compare the computational power of one-way deterministic multihead automata with one-way data-independent multihead automata, whereby we have brought an own result concerning this problem. Also, we have examined the relation of partially blind automata with our model, presenting our own result, too.

Nevertheless, data-independent multihead finite automata still provide a wide variety of open problems, primarily the question, if they really are computationally weaker than general multihead finite automata in two way setting. This would resolve the question, whether $\mathbf{NC}^1 = \mathbf{L}$ or not. Also, there still is an unresolved relation of one-way data-independent multihead finite automata concerning the head count.

Another problem, worth further research is the relation between partially blind finite automata and data-independent multihead finite automata in general (since our results regards only one-way setting and two heads).

We close our thesis with a constatation, that the research in the field of complexity of multihead finite automata and specifically data-independent ones, is worthwhile, since it has wide implications in theory of formal languages.

Bibliography

[1] J.E. Hopcroft and J.D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley Pub. Co., 1969.

[2] C.H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[3] J. Hartmanis. *On non-determinancy in simple computing devices*. Acta Inform. 1 (1972) 336-344.

[4] I.H. Sudborough. *On tape-bounded complexity classes and multihead finite automata*. J. Comput. System Sci. 10 (1975) 62-67.

[5] M.O. Rabin, D. Scott. *Finite Automata and Their Decision Problems*. IBM Journal of Research and Development 3 (1959) 114-125.

[6] A.C. Yao, R.L. Rivest. *k+1 heads are better than k*. J. ACM 25 (1978) 337-340.

[7] A.L. Rosenberg. *On multi-head finite automata*. IBM Journal of Research and Development 10 (1966) 388-394.

[8] B. Monien. *Two way multihead automata over a one-letter alphabet*. RAIRO Inform. Théor. 14 (1980) 67-82.

[9] M. Holzer. *Multi-head finite automata: data-independent versus data-dependent computations*. Theoret. Comput. Sci. 26 (1983) 335-342.

[10] D.A. Barrington. *Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1*. J. Comput. System Sci. 38 (1) (1989) 150-164.

[11] M. Holzer. *Data-independent versus data-dependent computations on multihead automata*. Doctoral Thesis, Universität Tübingen, 1998.

[12] P. Ďuriš. *A note on the hierarchy of one-way data-independent multi-head finite automata*. In Proceedings of Electronic Colloquium on Computational Complexity (ECCC), (2012) 92-92

[13] M. Holzer, M. Kutrib, A. Malcher. *Complexity of multi-head finite automata: Origins and directions*. Theoretical Computer Science, 412 (2011), 83-96.

[14] K. Wagner, G. Wechsung. *Computational complexity, Mathematics and its Applications (East Europeans Ser.)*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1986.