Five New Simulation Results on Turing Machines

Wiedermann, Jiří
1995

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Five New Simulation Results on Turing Machines

Jiří Wiedermann[1]

Technical report No. 631

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+422) 66414244 fax: (+422) 8585789
e-mail: wieder@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Five New Simulation Results on Turing Machines

Jiří Wiedermann[1]

Technical report No. 631

Abstract

The first result shows that any multihead Turing machine (TM) can be simulated by a cellular automaton in real time. This disproves a recent claim that multihead TMs are not scalable — i.e., that they cannot be physically realized in such a way that the duration of one machine move remains fixed regardless of the size of the machine.

The second result describes the effect of adding the second working tape to originally single tape TM. Especially, it is shown that any nondeterministic single–tape TM computation of time complexity $T(n)$ performing altogether $W(n)$ writes can be nondeterministically simulated by two tapes in time $O(\sqrt{T(n)W(n)})$. Thus, for $W(n) = o(T(n))$ a speed–up by adding the second tape is achieved.

Last three results deal with the problem of efficient simulation of RAMs by TMs. First, the complexity estimation of the earlier deterministic simulation of RAMs by TMs, designed by the author in 1983, is improved: using a new relation between uniform space and logarithmic time RAM complexity measures it is shown that a RAM of logarithmic time complexity $LT(n)$ can be simulated by a multitape Turing machine in time $O(LT^2(n)/\log LT(n))$. This presents the first known TM simulation that is subquadratic solely in terms of time comlexity of the simulated RAM. Then, a new nondeterministic simulation of any nondeterministic RAM by a multitape TM in time $O(LT(n)\log LT(n))$, is described; this simulation, although asymptotically not faster, is substantially simpler than the earlier one designed by the author. Finally, an optimal, linear time simulation of a nondeterministic RAM by a Σ_2–machine is presented.

Keywords

Computational complexity, simulations, Turing machines, RAM

1 Introduction

The present paper describes five new results related to the efficiency of the fundamental computational machine model — viz. Turing machines.

The first result deals with the problem of scalability of multihead TMs — i.e.,with the problem whether there is an efficient physical realization of such machines that could asymptotically preserve the unit–time complexity measure that is usually defined for the above machines.

The second results defines a natural class of nondeterministic, single–tape TM computations that can be speeded–up with the help of the second working tape.

The third results concernes an efficient deterministic simulation of a RAM by a TM; it presents an improvement over the best known result for the above problem, as described earlier by the present author.

The fourth result presents a new simulation technique for the case of simulating a nondeterministic RAM by a nondeterministic multitape TM. This new technique presents a substantial simplification of the author's earlier result.

The fifth result is concerned with the efficient, linear time simulation of RAMs by a still more powerful TM model — namely by a Σ_2 machine.

In a more detail, inclusive the respective motivations and related results, the above mentioned results are described in the sequel, each in a special section.

2 Scalability of Multihead Turing Machines

2.1 Introduction

A family $\mathcal{F}$ of abstract computational devices $\{F_n\}_{n\geq 1}$, where each F_n is capable to process any input of size n, is called *scalable* if there exist such a physical realization (i.e., such that obeys fundamental physical laws) of $\mathcal{F}$ that for each n the maximal duration of any computational step (measured in any real time units) of F_n does not depend on n. Roughly this notion of scalability has been introduced by Billardi and Preparata in [3], where it is claimed that besides cellular automata single head Turing machines are the only known example of scalable computational machines. Moreover, multihead Turing machines (having several heads at each tape) are presented as an example of a non-scalable computational device, since *"...communication between heads introduces a delay proportional to tape length, which precludes scalability"*.

In the following subsection we shall show that the last intuitive statement is not correct — namely we show that any multihead Turing machine can be simulated in an on–line manner by a linear cellular automaton. Since cellular automata represent a paradigmatical example of a scalable computational device, this simulation proves that multihead Turing machines are scalable as well. The scalability of multitape Turing machines, with one head per tape (that is implied also by the previous result), has been shown in [14], by showing their linear time simulation by so–called *parallel TM*.

2.2 Simulating Multihead TMs by Cellular Automata

The simulation is based on a clever encoding of a Turing machine tape in a zig–zag manner into the line of cellular automata in such a way that all Turing machine heads can remain in fact stationary — only tape segments between neighbouring heads are shifted.

The details are given in the proof of the following theorem, where w.l.o.g. we shall assume that a multihead TM to be simulated is a single–tape machine, since such a machine can simulate in a linear time any multitape multihead TM:

Theorem 2.1 *A single–tape multihead TM can be simulated by a row of cellular automata in real time and linear space.*

Proof outline. Let our multihead TM $\mathcal{M}$ have k heads, for some $k \geq 1$. Split its tape into *tape segments*, each segment consisting of tape cells laying between neighbouring heads (when the positions of two or more heads coincide, the respective segments are empty). Further, in each segment of length $\ell > 0$, behind its $\lceil \ell/2 \rceil$ cell, define a so—called *join*; if ℓ is odd add one more empty cell behind the join. Formally, define the joins also at the left and right end of the tape. Now fold the tape along the (internal) joins in a zig–zag manner. As a result, due to the definition of join positions, the individual tape heads appear one above the others.

The resulting folded tape can now be viewed as a single, non–folded k–track tape, where on each track the tape cells between the two consecutive joins are represented

This new k–track tape will now be represented with the help of a row of cellular automata. In this representation, each automaton corresponds to one cell having k "tracks" and remembers in its state the symbols written at the respective tracks, plus information whether there is a head (of $\mathcal{M}$) scanning the given track in the given cell, an whether there is a join on a given track behind the given cell.

The moves of the individual heads of $\mathcal{M}$ are simulated by the respective moves of corresponding tape segments. For instance, imagine that a particular head of $\mathcal{M}$ has to move right. In our representation on the folded tape this means that the tape contents between the two joins surrounding that particular head has to be shifted appropriately (to the left at odd tracks, and otherwise to the right). Assume that the tape contents has to be moved left. To do so, the head "issues" a signal to both sides, asking for shifting the cell contents one cell to the left; this signal is spread (and executed) until it reaches the respective joins, where it vanishes. Then the position of the respective joins are updated appropriately — i.e., in our case both joins are shifted by one position to the left.

A more thorough reflection reveals that the respective moves of all heads of $\mathcal{M}$ can be realized in parallel, by superimposing the respective signals issued by each head into a single complex signal exchanged between neighbouring cellular automata.

Note that the previous implementation of $\mathcal{M}$ requires a row of cellular automata that is unbounded from both sides. To change it into a row unbounded from only one side, it is necessary to "fold" the previous row of automata once more, this time along the position where the heads of $\mathcal{M}$ are represented, and again to "reprogramme" the automaton appropriately ...

It is clear that the resulting automaton will simulate the original machine in real time, and in linear space.

□

A similar simulation can be designed also for the case when the target machine is a single tape parallel Turing machine, whose scalability (called *physical feasibility* in the original paper) has been shown in [14].

3 On the Power of the Second Tape

3.1 Introduction

Single–tape Turing machines (TMs) are generally recognized as simple but cumbersome fundamental computational devices, since the known simulations of other fundamental machine models by single–tape TMs usually lead to at least quadratic time loss in time efficiency. Especially, it is well known that the reduction of tapes of a multitape TM down to a single tape leads also to a quadratic time simulation, and this simulation in the deterministic case cannot in general be improved [8].

Confronted with these results, one can ask what is to be expected in the opposite case — namely in the case when tapes are added to a single–tape TM. Of course, on one hand, one can immediately see that in some cases the addition of the second tape need not help in improving the time efficiency of the underlying machine (e.g., consider the case of recognizing the parity of a number, written in a usual binary notation). But, on the other hand, the example of palindrome recognition, that on a single tape requires time $\Omega(n^2)$, whereas on two tapes can be done in linear time indicates that there are cases when the introduction of the second tape can help substantially.

The above observation leads naturally to the problem of finding such class of single–tape computations that can be speeded–up by additional tape (or tapes), while otherwise preserving the computational resources of machines involved.

Surprisingly it appears that so far the problem in form as stated above has not been studied in complexity theory; only problems where the simulating machine is of a more powerful type have been studied occasionally. For instance, in [5] the problem of simulating a single–tape TM by a unit–cost RAM is addressed; in [11] the simulating machine is a probabilistic log–cost RAM, while in [10] it is an alternating Turing machine, and in [9] a Σ_2–machine. Thus, in all previous cases the computational ability of underlying machines has been increased by adding some computational resource (random access, probabilism, unbounded or bounded alternation, respectively) that is intuitively considered to be a more powerful resource than a mere additional tape.

There are, however, at least two results that show — albeit indirectly, nonconstructively — that one can save nontrivial amount of time in some cases by adding the second tape, *plus nondeterminism*, to originally *deterministic* single–tape machine [6], [9].

In the paper we shall show that there is a special, quite natural and wide class of nondeterministic single–tape Turing machine computations — realized by so–called *sporadically writing Turing machines* — that can be speeded–up appreciably by adding

the second tape. Thus, unlike in all previous cases, in this case the speed–up is achieved only by adding the second tape, without otherwise increasing the computational abilities of the underlying machine. Moreover, the speed–up is shown directly via fast simulation of the original machine.

Definition 3.1 *A Turing machine of time complexity $T(n)$ performing $W(n)$ write operations (i.e., operations that alter the contents of some tape cell), for any $W(n) \geq 1$, is called a sporadically writing machine, iff $W(n) = o(T(n))$.*

3.2 Speeding–up Sporadically Writing TMs

The following theorem states that under certain technical assumptions, any nondeterministic, single–tape sporadically writing Turing machine can be speeded–up by adding the second tape:

Theorem 3.1 *Let $\sqrt{T(n)/W(n)}$ be fully time constructible, with $T(n) \geq n^2$. Than any T(n)–time–bounded nondeterministic single–tape TM, performing W(n) writes, can be simulated by a nondeterministic two–tape TM (with one working tape actually being only a write–once tape) in time $O(\sqrt{T(n)W(n)})$.*

Proof Outline. The proof of the previous theorem is based on a space efficient, so–called *rectangular representation* of single–tape nondeterministic computation, (see e.g. [5] for a similar approach). In our proof, however, the rectangles are defined in such a way that for a simulating nondeterministic two–tape machine it is possible to guess and verify the computation of a sporadically writing single–tape Turing machine in a sublinear time.

To obtain the rectangular representation of any fixed computation of any single–tape nondeterministic machine M, it is helpful to see its computation as a sort of a diagram, that consists of individual instantaneous descriptions (ID's) of M, for subsequent time steps $t = 0, 1, \ldots$, written one above another. Besides the current contents of M's tape in each ID also the current head positions is recorded by writing down the current state of M's finite control over the position just scanned by M's head. In this way also the trajectory of Ms head, on a given input, is recorded in our diagram.

Following [7], for $T(n) \geq n^2$, w.l.o.g. we can assume that M is of space complexity $O(\sqrt{T(n)})$.

Now, starting from some position j, with $0 \leq j < b(n)$, the tape of M can be partitioned into blocks of length $b(n)$ (the value of $b(n)$ to be determined later) in such a way that the sum of crossing sequences over the block boundaries will be at most $T(n)/b(n)$ (see e.g. [5] or [7] for details).

The resulting block boundaries will split the computational diagram into at most $\sqrt{T(n)}/b(n)$ vertical slots. These slots will be further split into rectangles by writing the horizontal line across those points in the diagrams in which a write operation occurs, and by adding horizontal edges to the top, and the bottom of each slot. Since there

are at most $W(n)$ writes, there will be at most $O(W(n) + \sqrt{T(n)}/b(n))$ rectangles in our diagram.

Each rectangle will be thus represented by its two horizontal sides of length $b(n)$, giving the contents of the corresponding block at the respective time steps, and by the two vertical sides of length ℓ_1 and ℓ_2, respectively, giving the respective crossing sequences in chronological order for each side separately.

Thus, the size of each rectangle representation is of order $O(b(n) + \ell_1 + \ell_2)$, what in a total gives $O(W(n)b(n) + T(n)/b(n))$ for all rectangles.

Now, the idea of simulation is first to guess the above rectangular representation, and then to verify whether the guess was correct — i.e., whether all rectangles 'fit' together (so–called *global correctness*) and, whether each rectangle represents a valid piece of M's computation, with write operation occurring only at the horizontal rectangle boundaries (so–called *local correctness*).

Suppose that for a given computation the rectangular representation is guessed and written down on the M's tape in the following order: slot by slot, from left to right, and within each slot, rectangle by rectangle, in chronological order.

Then the global correctness can be verified easily with the help of the second tape by first verifying that all vertical borders fit and then verifying that all horizontal boundaries fit, in time linear in the total length of rectangular representation of M's computation.

The local verification of each rectangle can be done in time proportional to $O(b(n)+\ell_1+\ell_2)$, since — due to the 'non–write' condition maintained inside any block — within each rectangle the head can move only in finitely many ways that depend only on the state of the finite control of M at the moment of head entering into the given block. All possible ways for head movement within a block can be extracted in a linear time from the knowledge of corresponding crossing sequence by performing a fixed number of sweeps over this sequence.

Hence, the verification of all rectangles can be done in time proportional to their representation size.

To obtain the best performance of our simulation algorithm, it is enough to choose $b(n) = \sqrt{T(n)/W(n)}$.

Note also that by the previous simulation the number of writes is being increased to approximatively $\sqrt{T(n)W(n)}$.

□

To appreciate the quality of the result consider the case when a multitape Turing machine M_1 of time complexity $T(n)$ is to be simulated by a single–tape machine M_2; when the simulation is performed in a standard 'textbook' way, M_2 uses $O(T(n))$ writes with quadratic loss in time efficiency. Since the resulting machine M_2 is a sporadically writing single–tape machine, according to the previous theorem it can be simulated further by a two–tape nondeterministic machine M_3 in time $O(T^{3/2}(n))$. Thus, the transition from M_1 to M_3 has lead to time loss by factor $O(\sqrt{T(n)})$, whereas one knows that a multitape machine M_1 can be simulated directly by a nondeterministic two tape machine in linear time. This, on one hand, shows that for this particular case

our simulation can be improved but, on the other hand, leaves open the possibility that in some other cases the simulation is optimal.

Thus the optimality of the above simulation remains to be seen.

It is not clear whether a similar theorem could be proved also for the case of two–dimensional Turing machines; the case of multitape machines is open as well.

4 Deterministic Simulation of RAMs by TMs

4.1 Introduction

Efficient simulations of RAMs by TMs, stating polynomial time–equivalence of the respective models, belong among the basic results in complexity theory. A standard "textbook" result along these lines originates in the paper by Cook and Reckhow [4] where RAMs have been introduced for the first time. It states that a RAM (without multiplication or division) of *logarithmic time complexity* $LT(n)$ can be simulated by a multitape TM in time $O(LT^2(n))$. The question, whether the previous simulation can be improved remained open (see e.g. [2], or [1]).

Later on, in [13] a simulation of time complexity $O(LT^2(n)/\log US(n))$ has been designed, where $US(n)$ denotes *uniform space complexity* of the RAM computation at hand, i.e., the number of different RAM registers used during the computation. This result thus implies a subquadratic simulation, but it is not completely satisfactory, since the efficiency of TM simulation is not expressed solely in terms of time complexity of the original RAM computation.

In the sequel we shall prove a new relation between uniform space complexity, and logarithmic time complexity, of RAM computations, that leads to the further improvement in efficiency estimation of the previous simulation. Namely, a " pure subquadratic" time estimation of simulation of form $O(LT^2(n)/\log LT(n))$ can be achieved.

4.2 Relating Uniform Space to Logarithmic Time

From the definition of the respective RAM complexity measures it follows trivially that $US(n) \leq LT(n)$. However, it appears that also a less trivial dependency holds between the respective measures:

Lemma 4.1 *For any RAM computation it holds* $US(n) = O(LT(n)/\log LT(n))$.

Proof outline. Assume the opposite — i.e., that for an arbitrary $c_1 > 0$, $US(n) > c_1 LT(n)/\log LT(n)$, for sufficiently large n. Logarithmic cost of any RAM computation is certainly no less than the cost of mere accesses to all $US(n)$ registers used in the computation, and thus $LT(n) \geq \sum_{i=1}^{US(n)} \lceil \log(i+1) \rceil \geq c_2 US(n) \log US(n)$, for a suitable constant $c_2 > 0$. From our assumption it follows now from the previous estimate that $LT(n) > c_1 c_2 LT(n)/\log LT(n)(\log LT(n) - \log\log LT(n)) \geq c_1 c_2 c_3 LT(n)$, for a suitable constant c_3, with $0 < c_3 < 1$, and large n. Choosing c_1 such that $c_1 c_2 c_3 \geq 1$ leads to the conclusion that for sufficiently large n, $LT(n) > LT(n)$ — a contradiction. □

4.3 Subquadratic Simulation

The previous relation between uniform space and logarithmic time presents a key argument in the complexity analysis of the subsequent deterministic simulation of a RAM by a TM. Namely, by making use of it it is possible to improve the complexity estimation of the earlier simulation described by the author in [13]. Next we shall state the new improved result and outline the proof:

Theorem 4.1 *Any deterministic RAM computation of logarithmic time complexity $LT(n)$ can be simulated by a deterministic multitape TM in time $O(LT^2(n)/\log LT(n))$.*

Proof outline. At the heart of the simulation there is an efficient representation, and management, of the RAM registers on the TM tape. The registers used during the computation are represented on the tape as pairs $\#\#register\_number\#register\_contents\#\#$; moreover, these pairs are continuously kept in a sorted order, from left to right, in a non–increasing manner, according to length of their description.

The RAM instructions are realized in an on–line manner as dictated by the RAM program and current value of program counter. To realize an instruction requiring access to certain registers, its arguments are found by scanning the register tape from its right end, and transferred to a special working tape. Then the instruction is executed (i.e., the respective operation is evaluated), and the result, residing in some register, is inserted back into the register tape, into the proper place. For the details (e.g., allocating new registers, indirect addressing, organizing and maintaining the data on TM tapes, etc., see the original simulation in [13]). But even from this brief description it can be seen that the time complexity of realization of a single RAM instruction is proportional to the distance a TM head has to travel on register tape in order to find the necessary arguments. Due to the ordering of pairs on the register tape, and to the fact that tape is being traversed from its right end, this distance is bounded by the logarithmic cost of the instruction at hand, multiplied by the the total number of registers represented on the tape (i.e., by $US(n)$).

The total cost estimate of the entire simulation is then obtained by summing the costs of all instruction realizations performed during the simulations and clearly is of order $O(LT(n).US(n))$. Using the estimate on uniform space from the previous lemma, the last expression can be transformed into the one as claimed in the theorem.

□

As far as the optimality of the previous simulation is concerned, it has been shown in [13] that it is optimal for the on–line case (executing the next instruction only after executing the previous instruction), and the "integer" model (i.e., for the case that the addresses of registers ore treated as "atomic", physically indivisible, units).

5 Nondeterministic Simulation of RAMs by TMs

5.1 Introduction

Investigating nondeterministic simulations between RAMs and TMs is a less frequent task; the only result along these lines seems to be the result by the present author

in [13]. In this paper quite cumbersome recursive simulation algorithm, based od the divide and conquer approach, has been designed. In what follows we shall describe a new simulation algorithm, that, although asymptotically of the same efficiency as the original one, is easy to explain and easy to analyze.

5.2 A Superlinear Nondeterministic Simulation

As it is to be expected, a nondeterministic simulation of a RAM by a multitape TM makes heavy use of nondeterminism in order to guess the sequence of instructions realized by RAM; whether the guessed sequence represents indeed a valid RAM computation is subsequently verified with the help of sorting.

Theorem 5.1 *A nondeterministic RAM computation of logarithmic time complexity $NLT(n)$ can be simulated by a nondeterministic two–tape Turing machine of time complexity $O(NLT(n) \log NLT(n))$.*

Proof outline The Turing machine first guesses, and writes down, on one of its tapes, the "history" of RAM computation, as dictated by the RAM program, in the following way:

- the history is a sequence of "RAM instruction snapshots", in chronological order;
- each snapshot is identified by the instruction code, and contains a double list of the respective instruction operands;
- the first list specifies the values of operands as they enter into the instruction, while the second list specifies the values as they are left after the instruction is executed;
- an operand value is given by the pair $\#\#register\_number\#register\_contents\#\#$, similarly as in the case of deterministic simulation.

All the previous data items are separated by suitable separators that enable the TM to identify the individual items. Note already now that only the instruction code, and the operand values within the first list, must be guessed, since the values in the the second list are uniquely determined by the previous data.

Next the TM has to verify the consistency of its guesses — i.e., whether the values of operands entering into each instruction are the same as those left in the respective registers by some earlier instructions that have altered the same registers for the last time (so–called *register input–output condition*). This is done by sorting the pairs, representing the operands in the respective parts of instruction snapshots, according to register addresses, in a stable manner. In this way registers with the same address will be brought together and the condition of stability will guarantee for them that in the resulting sorted sequence a correct chronology of register rewritings can be verified, simply by checking the previous register "input–output" condition for any two physically neighbouring (on the tape) registers.

The most time consuming process in the previous simulation is the sorting phase. It is known that a tape of length $NLT(n)$ can be sorted, e.g., with the help of a two–tape merge sort, using a logarithmic number of passes over the data. This leads straightforwardly to the estimation as stated in the theorem.

□

The previous simulation is optimal in the "integer" model mentioned in the close of the previous section, since (nondeterministic) sorting of $n/\log n$ integers, each of size $\log n$, requires $O(n \log n)$ time on a multitape TM [12], while, clearly, on a nondeterministic RAM the same task can be accomplished in linear time. Note also that any faster nondeterministic sorting algorithm would improve the above simulation as well.

6 Simulating RAMs by Two Alternations

From the remark in the close of the previous section it follows that a faster simulation could be achieved in the case when we would be able to sort faster on the simulating machine. This is the case of Σ_2–machines (i.e., nondeterministic TMs that are, moreover, allowed to perform a single "universal" alternation at the end of computation):

Theorem 6.1 *Any nondeterministic RAM of time complexity $NLT(n)$ can be simulated by a multitape Σ_2–machine in linear time, performing $O(\log NLT(n))$ universal branching operations.*

Proof outline. The first simulation phase is identical to that from the case of the previous nondeterministic simulation. Clearly, this task requires linear time. In the second phase, where the consistency of guessed instruction sequence has to be verified, the parallelism offered by Σ_2–machines is invoked: using $O(\log NLT(n))$ universal branchings, a sufficient number (less than $LNT(n)$) of machine copies are generated, one for each instruction used, and each copy verifies, in parallel, the consistency of guesses for the respective registers involved in the instruction at hand. On a two–tape Σ_2–machine, this verification can be done again in linear time.

□

Obviously, in some cases the last simulation cannot be improved (think for instance about the sorting problem mentioned in the close of the previous section). An interesting open question is whether by a larger number of alternations it is possible to achieve a sublinear time simulation of a RAM.

7 Conclusions

We have presented five new results related to Turing machines. All of them are in fact represented by efficient simulations, where at least at one end — either as a simulated machine, or as a simulating machine, some model of Turing machine is to be found. All the results bring new insights into the computational efficiency of the respective machines.

The first result, showing the scalability of multihead TMs, reassures the physical feasibility of yet an other model of TMs (in the sense that the model at hand can be physically realized with the help of cellular automata, with no loss in efficiency), and points, in a sense, to a certain "well definideness" of all variants of TMs that have been shown so far to be scalable.

Next result shows that by adding the second tape to originally single tape TM one can, in some cases, speed–up the computation of the original machine. It would be of interest to find further classes of such computations, or even to show that additional tape helps always, for sufficiently complex computations.

Last three results are concerned with the classical task of efficient simulation of RAMs by TMs. Turing machines with increasing computational power are considered. For the first time, a "pure subquadratic", in terms of the time complexity of the RAM machine to be simulated, deterministic simulation has been showed. Next, a new, simple and transparent nondeterministic simulation algorithm of superlinear time complexity has been designed. Finally, a "new" task of simulating RAMs by bounded alternation TMs has been studied. It has been proved that two alternations are enough to achieve a liner time simulation. Except for the latter case, the question of optimality of the former two RAM simulations presents an open problem.

Bibliography

[1] Aho, A.V. — Hopcroft, J. — Ullman, J.D.: The Design and Analysis of Computer Algorithms. *Addison–Wesley*, Reading, Mass., 1974

[2] Borodin, A.: Computational Complexity: Theory and Practice. In: *Currents in Theory of Computing*, edited by A.V. Aho, Prentice Hall, Englewood Cliffs, N.J., 1974

[3] Billardi, G. — Preparata, F.P.: Horizons of Parallel Computation. Technical Report No. CS–93–20, May 1993

[4] Cook, S.A. — Reckhow, R.A.: Time Bounded Random Access Machines. In:*JCSS*, vol. **7**, No. 4, 1973, pp. 354–375

[5] Hopcroft, J. — Paul, W. — Valiant L.: On time versus space and related problems. *Proc. IEEE FOCS* **16**, 1975, pp. 57–64

[6] Kannan, R.: Alternation and the power of nondeterminism (Extended abstract). *Proc. 15–th STOC*, 1983, pp. 344–346

[7] Li, M. — Neuféglise, H. — Torenvliet, L. — van Emde Boas, P.: On Space Efficient Simulations. *ITLI Prepublication Series*, CS–89–03, University of Amsterdam, 1989

[8] Li, M. — Vitányi, P. M. B.: Kolmogorov complexity and its applications. In: *Handbook of theoretical computer science*, Vol. A : Algorithms and complexity (J. van Leeuwen, Editor), Elsevier, Amsterdam, 1990

[9] Maass, W. — Schorr, A.: Speed-up of Turing Machines with One Work Tape and Two–way Input Tape. *SIAM J. Comput.*, Vol. **16**, no. 1, 1987, pp. 195–202

[10] Paul, W. — Prauss, E. J. — Reischuk, R.: On Alternation. *Acta Informatica*, **14**, 1980, pp. 243–255

[11] Robson, J. M.: Fast probabilistic RAM simulation of single tape Turing machines computations. *Information and Control* **63**, 1984, pp. 67–87

[12] Stoss, H.J.: Rangiercomplexität von Permutationen. *Acta Informatica*, No. 2, 1973, pp. 80–96

[13] Wiedermann, J.: Deterministic and Nondeterministic Simulation of the RAM by the Turing Machine. In: *Proceedings of IFIP'83*, R.E.A. Mason (Ed.), Elsevier Science Publishers B. V. (North Holland), 1983

[14] Wiedermann, J.: Weak Parallel Machines: a New Class of Physically Feasible Parallel Machine Models. Proc. 17th International Symposium MFCS'92, LNCS Vol. 629, Springer Verlag, 1992, pp. 95–111