Czech Pattern Recognition Workshop 2000, Tomáš Svoboda (Ed.)
Peršlák, Czech Republic, February 2–4, 2000
Czech Pattern Recognition Society

Parallel Turing Machines on a Two-Dimensional Tape

Daniel Průša* František Mráz†

Charles University, Department of Computer Science,
Malostranské nám. 25, 118 00 PRAHA 1, Czech Republic
e-mail: mraz@ksvi.mff.cuni.cz, prusa@barbora.ms.mff.cuni.cz

Abstract *We introduce two models of parallel two-dimensional Turing machine with different possibilities of communication among control units and we compare them.*

1 Introduction

Our motivation comes from the area of picture recognition where parallel approaches are being investigated. The parallel models, that will be introduced in this paper, can be the theoretical basis for such a research. This paper is based on master thesis [3].

In [1] there are studied 2-dimensional generalizations of finite state automata. We propose a generalization of the classical Turing machine. Our models work on a two dimensional tape and are enhanced by more than one control unit (with one head each). We assume that the number of the input picture fields is large and therefore we require the number of control units to be substantially smaller than the number of the input picture elements. The number of units always corresponds to the number of rows of the input picture.

The communication between control units always plays the major role in an efficiency of parallel computations. We suggest two models of a parallel Turing machine which differ just in the way of communication between its control units.

In the first model, the control units are ordered in a ring topology and direct communication is possible between the neighbour units only. In the second model, the exchange of information among the units is possible only by writing and reading symbols on the tape and there is not any direct communication between units there.

In the following section we introduce picture languages and the parallel models. Also, we give one picture language L_P there, which is later used in the comparison of the introduced models. In Sect. 4 we show a generalization of the crossing sequences (introduced in [2]) for the case of a two dimensional tape. Using the crossing sequences technique, we show some lower bound on the time complexity of recognition of L_P on the second introduced model. This yields the final comparison between the models, which shows that the direct communication between the control units is better than communication by writing and reading on a tape.

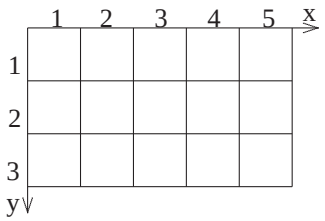

Figure 1: The system of coordinates used in picture descriptions

2 Definitions

A picture P over an alphabet B (which is the input for our models) is a two-dimensional $m \times n$ array of fields containing symbols from the alphabet B. m is the number of rows and n is the number of columns of P. In the following descriptions we will use the system of coordinates in a picture which is depicted in Fig. 1.

Speaking about a position of a specific field, we use words like up, down, right, left, first row, last row etc. with respect to this scheme. We will work with models of a two-dimensional Turing machine, which have such pictures as their input. Then we suppose, that all the fields of the two-dimensional tape, beside the fields from the rectangular area containing P, contain a special background symbol $\#$ ($\notin B$).

Let P_1 and P_2 be two pictures of size $k \times m$ and $k \times n$ respectively. The *column concatenation* $P_1 \oplus P_2$ is the picture of size $k \times (m+n)$ arising by pasting together last column of P_1 and the first column of P_2. Similarly, for two pictures P_1 and P_2 of size $m \times k$ and $n \times k$, respectively, the *row concatenation* $P_1 \ominus P_2$ is the picture of size $(m+n) \times k$ arising by pasting together last row of P_1 and the first row of P_2.

In a definition of a model of parallel computations, many attributes of the model must be specified. Such attributes are for example the starting configuration and the halting configuration, parameters connected with the parallel architecture of the model. One of the important issues is the management of conflicts which arise when at the same time two or more heads operate on the same field of the working tape. Another important attribute is the communication between the control units, which has been already mentioned. A thorough

*Supported by the Grant Agency of Charles University, Grant-No. 157/1999/A INF/MFF

†Supported by the Grant Agency of the Czech Republic, Grant-No. 201/99/0236

discussion of such attributes was done in [3].

Special attention deserves the problem how to distinguish the control units during a computation. It is necessary to have a possibility to distinguish two units so that they do not make the same activity during the computation (in the case of the identical activity the units would not be used effectively). This problem is mainly connected with the model without communication. In our models the control units are distinguished by different positions of their heads in initial configurations. Another possibility could allow starting configurations with several heads placed on the same field and their control units being in the same state. In this case, we would need a special mechanism to distinguish units, e. g. for every unit to have a flag, which indicates a successful or an unsuccessful writing.

In our models in a step an arbitrary number of heads can read the same field of the tape – thus the reading conflicts are allowed. The writing conflicts in our models are managed by a paradigm called PRIORITY, which is often used in parallel models. We suppose a linear ordering on the set of control units. If there is a writing conflict, the writing is done by the unit, which is the smallest in the ordering (has the smallest priority) among the units writing to the field.

Definition 1 The parallel Turing machine with linear number of heads with communication *(*CommTM*) is a tuple* $T = (Q, A, B, \delta, q_0, F, G)$*, where*

- Q *is a finite set of states,*
- $q_0 \in Q$ *is the initial state,*
- $F \subseteq Q$ *is a set of halting states,*
- $G \subseteq F$ *is a set of accepting states,*
- A *is a finite tape alphabet which does not contain the symbol* Δ *and contains the background symbol* $\#$*,*
- $B \subset A$ *is an input alphabet which does not contain the background symbol* $\#$*,*
- $\delta : Q^3 \times A \to Q \times (A \cup \Delta) \times M$ *is the transition function, where* $M = \{L, R, U, D, N\}$ *is a set of elements representing the movements of the head (to the left, right, up, down and no movement, resp.). There is a following restriction on this function:*
  $(\forall q_2 \in F)(\forall q_1, q_3 \in Q)(\forall a \in A)\delta(q_1, q_2, q_3, a) = (q_2, \Delta, N)$*. (The restriction says, that every control unit, which is in a halting state, does not change its state, position of its head and this head does not write any symbol.)*

Turing machine T consists of the control units with the finite set of states. Every control unit has its own head, which can move vertically and horizontally on the working tape that contains symbols from the alphabet A.

We suppose, that the set of the control units is linearly ordered – by i-th control unit we mean the i-th unit in the linear ordering. According to this ordering we define the ring topology of the control units – every unit has the right neighbour which corresponds to the following unit and similarly the left neighbour corresponds to the previous unit in the ordering. The first and the last units are exceptions. The left neighbour of the first unit is the last unit and vice versa.

The *configuration* of the machine T consists of the contents of the tape, the states of all control units and the locations of their heads. The computation on an input picture P of size $m \times n$ is being done by m control units. In the *initial configuration* all the fields of the two-dimensional tape, beside the fields from the rectangular area containing P, contain the special background symbol $\#$ ($\notin B$), i-th control unit ($1 \leq i \leq m$) has its head on i-th field of the first column of the input picture and all the control units are in the state q_0.

T works in steps. The activity of a particular head is determined by the transition function. $\delta(q_l, q, q_r, a) = (q', b, d)$ means that when the control unit is in the state q, scans a, the left neighbour unit is in the state q_l and the right neighbour unit is in the state q_r then the control unit enters the state q', writes b in the scanned field and moves in the direction d. In a parallel step, every control unit changes its state, rewrites the scanned symbol and moves its head. Writing the symbol Δ has a special meaning – no writing. When in a step there are more than one head above the same field of the tape, the writing is done by the control unit, which writes a symbol that differs from Δ and is the smallest in the ordering among units with heads over the same field. If every head writes down Δ then there is not any overwriting.

A configuration in which states of all control units are from the set F is called *final configuration*. The computation of the machine T ends, when T reaches such a configuration (for the first time). In a final configuration, if the first control unit is in an accepting state (from G), the input picture is accepted, otherwise the input picture is rejected. The halting states from $F - G$, which are not accepting, are called the rejecting states.

Definition 2 The parallel Turing machine with the linear number of heads without communication *(*NoCommTM*) is a tuple* $T = (Q, A, B, \delta, q_0, F, G)$*, where the particular components have the same meaning as for CommTM. The only difference between the formal definitions of these models is in the transition function. The transition function is of the form* $\delta : Q \times A \to Q \times (A \cup \Delta) \times M$*. The meaning of* M *and* Δ *is identical to the meaning of these symbols in the previous model (Definition 1). There is a restriction on the transition function again:*
$(\forall q \in F)(\forall a \in A)\delta(q, a) = (q, \Delta, N)$*. This restriction has the same meaning as above.*

As we have already mentioned, the CommTM and NoCommTM machines differ only in the definitions of a computation step. In a parallel step, every control unit of a NoCommTM makes a step according to the transition function δ. $\delta(q, a) = (q', b, d)$ means that when the control unit is in the state q and scans a then it enters the state q', writes b in the scanned field and moves in direction d. It means, the computation step of a particular unit does not depend on the states of other units.

We say, that the computation of a Turing machine (*CommTM*, resp. *NoCommTM*) *is bounded by the input picture area*, iff every head is moving only over the fields of the input picture area during the whole computation and eventually a head can move out of the input picture area in a computation step, but then this head must in the next step move backwards to the input picture area and the head cannot write any symbol on the field, which is out of the input picture area. The computations, which will follow, are bounded by the input picture area.

For a given machine we define the time complexity of its computation in the usual way. We should only emphasize that we will consider the complexity as a function of the input picture diameter. The diameter of a picture is defined as the upper integer part of the arithmetical average of the picture sizes. In the case of a square picture, the diameter is identical with its size.

Obviously, each NoCommTM machine M_N can be simulated by a CommTM machine M_C working in the same time as M_N. On the other side, any CommTM M_C can be simulated by a NoCommTM M_N. Thus, the models have the same recognition power, but we will show that the communication is an advantage. More exactly, we will show a CommTM M_C which cannot be simulated by a NoCommTM with a constant slowdown.

3 The language L_P

Let L_P denote the picture language containing exactly the square pictures in which all rows are palindromes of the form ww^R, where $w \in \{0,1\}^*$. We will show, how L_P can be recognized by our models.

Let us sketch the *NoCommTM* machine M_N recognizing L_P. The computation of M_N on an input picture P has three phases. In the first phase, the square form $n \times n$, where n is even, of the picture is checked. In the second phase, i-th head tests whether i-th row is a palindrome of the form ww_R for some $w \in \{0,1\}^*$. In the third phase, results from the all rows are collected by the first control unit.

1. The verification in the first phase can be done in the following way. The control unit finds out whether it is the first unit in the ordering by moving up and down (the field above the field scanned initially by the first unit contains the background symbol #). The units beside the first one stay at their initial field. The first unit climbs the diagonal from the left upper corner. The climbing should end in the right down corner of the picture on an even row. The unit remembers the result of the check and moves its head into the first column of the last row. Then it moves upward. On the way up it marks the symbols in the first column by a mark denoting the result of the verification. If the mark scanned by the i-th head ($1 < i$) in the respective row denotes 'O.K.' then the i-th control unit starts the second phase on the i-th row, otherwise seeing the mark of failure the respective unit enters a halting state. The first control unit ends the first phase in the left upper corner. If the verification failed, then the unit enters a rejecting halting state, otherwise it starts the second phase on the first row.

2. The verification of a row in the second phase is done by consecutive comparison of the 1st and n-th symbol of the row, 2nd and $(n-1)$-th symbol and so on (using some marking on the already compared symbols). During the computation, we record in states, whether we have found a pair of different symbols. We continue until all fields of the row are marked. Then the head moves into the first field of the row and writes there 1 when the row is a palindrome and writes 0 if it is not a palindrome.

3. In the third phase, the units beside the first one enter some halting state. The first control unit consequently scans the first column in the top-down direction. If it scans some symbol 0, then it enters an rejecting state, otherwise if it enters a field containing the background symbol #, then it accepts (all fields in the first row of the picture contain 1).

The verification of the square form requires time $O(n)$. The verification of a row requires for every unit the same number of computing steps (the verification continues even if a pair of different symbols is found). It implies that the heads of the i-th unit, for $i > 1$, come into the leftmost column before the head of the first unit finishes the second phase. The first unit scans consecutively all particular results until it finds a row which is not palindrome (then rejects) or it enters the field under the input picture (in that case all the units signalized palindromes, thus M_N accepts).

The verification of a row requires a quadratic (in n) time and the final evaluation of all results requires a linear time. Thus M_N recognizes the language L_P in a quadratic time.

Let us show how to recognize the language L_P by a *CommTM* M_C. It can verify the square form in the same way as M_N. The second phase – the verification of palindromes in the rows, can be made in linear time by utilizing the communication between neighbour units. The verification of a row is done by two control units which are neighbours in the topology. First of all we divide control units into two groups according to their order - odd one and even one. This division is done by a signal, which is sent by the first unit. The signal is consecutively passed towards the last unit. The signal carries one-bit information, that is always changed to the opposite value after the signal is passed to the next unit. (Let us remark that the signal passing through the topology of units is done by convenient unit states changing. The change of a state depends on states of two neighbour units and that is why it is possible to pass a signal from one unit to another.) Next, the heads of the even units move into the end of the previous row and then, together with the head of the corresponding odd unit, which is placed in the same row now, they verify this row. The verification is made by synchronized scanning the row in opposite directions. After that the next row in the picture is verified by the same pair of units. That means every pair of units verifies two rows.

Then all the particular results are sent to the first unit as a signal. The last unit sends a signal, which contains one-bit information, through the other units in topology towards the first unit. If a control unit has found that one of its rows is not a palindrome of the form ww^R then such unit sets the value

of the received signal to a value, which indicates a failure and halts. According to the final value of the signal the first control unit decides whether an input picture belongs to L_P or does not.

4 Crossing Sequences

The next definition is a generalization of the crossing sequence defined in [2] for the classical Turing machine. Our generalization is based on the fact, that we define a crossing sequence for a computation on a two-dimensional input and in addition for more than one head, in particular for a computation of a *NoCommTM* machine.

Definition 3 *Let $T = (Q, A, B, \delta, q_0, F, G)$ be a NoCommTM machine, P be an input picture of size $m \times n$, i be an integer, $1 \leq i < n$ and the computation $\mathcal{C}$ of T on P is finite. The* crossing sequence of the machine T on the horizontal position i of the input picture P *is a sequence $\mathcal{K}$ constructed in the following way:*

1. *We initialize $\mathcal{K}$ by an empty sequence. We examine each step of $\mathcal{C}$ from the first one until the last one. We start by step $j = 1$.*

2. *For $h = 1, \ldots, m$ (in this order), we examine the j-th step made by the control unit h. If the control unit h enters a state q and its head moves from the i-th column to the $i + 1$-th column or vice versa in a row r, we append the new element (h, q, r, j) to $\mathcal{K}$. The element means that the head of the unit h in parallel step j crossed the border between columns i and $i + 1$ in the row r. Otherwise, if the head does not move in this way, nothing is appended to $\mathcal{K}$.*

   If j is not the last step of $\mathcal{C}$, we increase j by one and repeat this step.

The crossing sequence on the position i of an input picture P records all the movements of all the heads through the border between i-th and $(i + 1)$-st column of P.

We can define a similar notion of a crossing sequence of T on the vertical position i of a picture P which records all the movements of heads between the i-th and $(i + 1)$-st row of P.

Let i be a number of a column. We can divide the working area into two parts – a left part, which includes every field of the picture with the x-coordinate less or equal to i and a right part, which includes the remaining fields of the picture. If we know the crossing sequence on the horizontal position i and the contents of one of the two parts in the starting configuration, we can determine, how this part will be changed during the computation without knowing the rest of the input picture. We can determine positions of the heads and states of the control units in this part, because we know for a given control unit, which moves into the second part in what row and in which state its head returns (if ever). It is important here that we consider the *NoCommTM* model, i.e. without communications. The only possibility how to get a piece of information from the second part of the working area to the first part is based on storing such information in the control unit state. This observation cannot be done for the model with communication (*CommTM*).

The previous observations are summarized in the following lemma:

Lemma 4 *Let T be a* NoCommTM *machine. Let us assume T accepts two pictures $P_1 = a_1 \oplus \ldots \oplus a_n$ and $P_2 = b_1 \oplus \ldots \oplus b_m$, where a_i, b_j are pictures consisting of one column and with the same number of the rows. Let us further assume, that there are two integers i, j, where $1 \leq i \leq n$, $1 \leq j \leq m$, and the crossing sequence of T on the horizontal position i of P_1 is identical with the crossing sequence of T on the horizontal position j of P_2, then the machine T accepts picture $P_3 = a_1 \oplus \ldots \oplus a_i \oplus b_{j+1} \oplus \ldots \oplus b_m$ too.*

Proof: Let us consider the computation of T on the picture P_3, in particular its k-th step. Note, please, that the computations of T on P_1 and P_2 can have different lengths. Let l_1 denotes the length of the computation of T on P_1.

If $k \leq l_1$, then the left part of the working area, which is determined by the i-th column of P_3, is the same as the respective part of the working area after the k-th step of the computation on P_1. Moreover, in these (parts of) configurations, the heads of the corresponding control units are located on the same positions. This holds because of the identical crossing sequence on the i-th position of P_1 and P_3 and identical content of the first i columns of P_1 and P_2.

If $k > l_1$, then the left part of the configuration of the computation on P_3 in the k-th step is the same as the left part of the ending configuration of the computation on P_1. The reason for this is that the heads do not move between i-th and $(i + 1)$-th column after finishing the computation on P_1 and the control units of all the heads which are located in the left part are in final states (thus, they cannot make any changes).

We get a similar observation for the right parts of configurations of the computations on P_3 and on P_2.

From the above observation it follows that the computation of T on P_3 is finite and in the halting configuration of the computation of T on P_3, the first control unit is in an accepting state (because this state is an ending state of the first unit in the computation on P_1 or in the computation on P_2). □

Obviously, an analogous lemma can be shown for horizontal division of a picture using crossing sequences on vertical positions.

Lemma 5 *Let $T = (Q, A, B, \delta, q_0, F, G)$ be a* NoCommTM *machine with the time complexity t, let P be an input picture of size $m \times n$. Then the sum of the lengths of all non empty crossing sequences on vertical positions $1, \ldots, n - 1$ in P is less than or equal to $m.t(\frac{m+n}{2})$.*

Proof: Every control unit contributes to one of the crossing sequences during its horizontal movement to the right or to the left with just one element. The number of head movements is bounded by the number of the computation steps. From this the statement of the lemma follows. □

5 Lower Bounds on the Time Complexity

Using the crossing sequences and lemmas we get some results about lower bounds on the time complexity of the recognition of L_P by *NoCommTM* machines. This is a generalization of a technique from [2].

Theorem 6 *Let r be a real number, $r > 1$. There does not exist any* NoCommTM *machine which recognizes the language L_P in time $O(n^2/log^r n)$.*

Proof: Let $T = (Q, A, B, \delta, q_0, F, G)$ be a *NoCommTM* machine which recognizes L_P in time $t(n)$. We will show that $t(n) \notin O(n^2/log^r n)$ for every real number $r > 1$.

First of all we make an observation. Let us consider two pictures of the same sizes from L_P which are of the forms $P_1 = W_1 \circledcirc W_2 \circledcirc W_2^R \circledcirc W_1^R$ and $P_2 = W_3 \circledcirc W_4 \circledcirc W_4^R \circledcirc W_3^R$ (P^R is the picture in which each row is the reversion of the corresponding row from P). Let us assume that the vertical crossing sequences on the positions between W_1, W_2 and between W_3, W_4 are identical, W_1 and W_3 have the same number of columns and $W_1 \neq W_3$. According to Lemma 4 T accepts $W_1 \circledcirc W_4 \circledcirc W_4^R \circledcirc W_3^R$ too, but this is not possible because the picture $W_1 \circledcirc W_4 \circledcirc W_4^R \circledcirc W_3^R$ does not belong to L_P, because of $W_1 \neq W_3$. Thus the mentioned crossing sequences are different. We denote this observation as (1).

Let n be an even positive integer. Let $\bar{L}$ denote the set of all pictures from L_P of size $n \times n$. Let $s = |Q|$. For n we define the function $p : \{1, \ldots, n\} \to N$, where $p(i)$ is the average length of the crossing sequence on the position i for pictures from $\bar{L}$.

There are $2^{\frac{n^2}{2}}$ pictures in $\bar{L}$. At least one half of these pictures (i.e. $2^{\frac{n^2}{2}-1}$) have the length of the crossing sequence on the position i less or equal to $2.p(i)$ (because $p(i)$ is the arithmetical average of all such lengths). We estimate the number of all crossing sequences with lengths less or equal to $2.p(i)$. Each j-th element from a crossing sequence is of the form (h_j, q_j, r_j, t_j). The first component of such a tuple is one of n different values, the second component is one of s different values, the third component is one of n different values and finally the fourth component is one of the $t(n)$ different values. We get that the number of all distinct crossing sequences of the length k is maximally $(s.n^2.t(n))^k$ and it means that the number of all crossing sequences of the length less or equal to $2.p(i)$, is maximally $(s.n^2.t(n))^{2.p(i)+1}$ (because $\sum_{k=1}^{m} q^k = q.\frac{q^m-1}{q-1} \leq q^{m+1}$ we take $q = s.n^2.t(n)$ and $m = 2.p(i)$).

Let i be an integer from $\{1, \ldots, \frac{n}{2}\}$. From the two previous paragraphs it follows that there are at least $\frac{2^{\frac{n^2}{2}-1}}{(s.n^2.t(n))^{2.p(i)+1}}$ pictures with an identical crossing sequence on the position i. It follows from (1) that all these pictures have identical columns $1, \ldots, i$. There are $2^{n.(\frac{n}{2}-i)}$ such a different pictures. It implies the inequality:

$$\frac{2^{\frac{n^2}{2}-1}}{(s.n^2.t(n))^{2.p(i)+1}} \leq 2^{n.(\frac{n}{2}-i)}$$

We get from this:

$$i.n - 1 \leq (2.p(i) + 1).log_2(s.n^2.t(n))$$

We sum these inequalities for $i \in \{1, \ldots, \frac{n}{2}\}$:

$$\frac{n}{4}.\left(1 + \frac{n}{2}\right).n - \frac{n}{2} \leq \left(2.\sum_{i=1}^{\frac{n}{2}} p(i) + \frac{n}{2}\right).log_2(s.n^2.t(n))$$

According to Lemma 5 we have $\sum_{i=1}^{\frac{n}{2}} p(i) \leq n.t(n)$, it implies:

$$\frac{n^2}{8} + \frac{n}{4} - \frac{1}{2} \leq \left(2.t(n) + \frac{1}{2}\right).log_2(s.n^2.t(n))$$

The time complexity of the machine T has to satisfy this inequality for every even $n > 0$. We can see that a linear time complexity cannot satisfy the inequality, because for $t(n) = O(n)$ the right part of the inequality is $O(n.log(n))$ while the left part is $\Omega(n^2)$. It implies, L_P cannot be recognized in a linear time.

We can see next that the inequality is not satisfied for a time complexity $t(n) = O(\frac{n^2}{log^r n})$, where r is a real number larger than 1, too. The right part is $O(\frac{n^2.log(n)}{log^r(n)})$ in this case. (Let us remark that for $r = 1$ the inequality is satisfied.) □

6 Conclusions

In this paper we have introduced two models of parallel Turing machines working on a 2-dimensional tape. The control units of a *NoCommTM* machine in contrast to control units of a *CommTM* machine cannot communicate. We studied how this fact influences time complexity of recognition of these models. We have presented the language L_P which can be recognized in a linear time by a *CommTM* machine and we proved that it cannot be done by any *NoCommTM* machine. We know that L_P can be recognized in a quadratic time by a *NoCommTM* machine and our theorem says that it is not possible in time $O(\frac{n^2}{log^r n})$ for every real number $r > 1$. There still remains the question if this lower bound of recognition can be improved or if there exists a *NoCommTM* machine which recognizes L_P in a better than quadratic time.

Let us return to the definitions of our models. We can see that there is a nonsymetry – the number of control units does not depend on the number of columns of an input picture. Each computation on a picture with just one row of n fields is done by one control unit, while in the case of a one-column picture with n fields, the computation is done by n control units. It is possible to modify our models to have not such a nonsymetry. E.g. in the initial configuration, the heads could be placed in the first row and in the first column of an input picture. But such a modification would not change the results presented in this paper and our definition is more simple.

References

[1] D. Giammarresi and A. Restivo. Two-dimensional languages. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words, pages 215–267. Springer-Verlag, Berlin, 1997.

[2] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, N. Reading, MA, 1980.
[3] D. Průša. Parallel automata working on a 2-dimensional inputs. Masterthesis, Charles University, Prague, 1998. In czech.