Acta Informatica 1, 336—344 (1972)
© by Springer-Verlag 1972

On Non-Determinancy in Simple Computing Devices

J. Hartmanis*

Received October 21, 1971

Summary. This paper studies one-tape Turing machines with k read-only heads which are restricted to the original input. The main result shows that if any set accepted by such a 3-head non-deterministic Turing machine can be accepted by a deterministic Turing machine with more read-only heads, then the deterministic and non-deterministic context-sensitive languages are identical. Several related results are derived and some tantalizing open problems are discussed.

Introduction and Preliminaries

This paper deals with the problem of non-determinacy in some very simple computing devices and shows that if non-determinacy can be eliminated for these devices by using a "more complicated" device of the same type, then non-determinacy can be eliminated in a wide class of computing devices, including the linearly bounded automata. Thus these results give a very simple class of computing devices, for which we cannot yet answer the question about the power of non-determinism, and for which a deeper understanding of these problems may yield a solution of the classic and embarassing problem about non-deterministic context-sensitive languages [1, 2].

The first class of computing devices considered consists of one-tape Turing machines with k read-only heads, $k = 1, 2, 3, \ldots,$ which are restricted to the original input. We assume that the input is placed between special end-markers and thus the k heads can read the input but cannot go past the end markers. For sake of brevity we will refer to these devices as *k-head automata.* The input is *accepted* if, after starting the automaton in its starting state with all heads on the left endmarker, the automaton enters an accepting state and halts. The input is *rejected* if the automaton enters a rejecting state and halts or if it does not halt. Note that the automaton can sense when two heads are on the same tape square.

Let C_D^k denote the set of all languages accepted by deterministic k-head automata and let C_N^k denote the class of all languages accepted by non-deterministic k-head automata.

The main result of this paper shows that if every non-deterministic 3-head automaton can be replaced by a deterministic automaton with more heads, that is

$$C_N^3 \subseteq U_{k=1}^{\infty} C_D^k,$$

then the deterministic and non-deterministic context-sensitive languages are the same. In other words, if the non-determinacy for all 3-head automata can be

* This research has been supported in part by National Science Foundation Grant GJ-155.

eliminated by using more heads, then any non-deterministic linearly bounded automaton (l.b.a.) can be replaced by a deterministic l.b.a. The proof of this result is achieved by a three fold reduction of this problem and contains a novel encoding of many head positions on an input into one head position on an extended input.

Surprisingly, it looks that even the simple sounding problem of determining whether there exists a set accepted by a non-deterministic 3-head automaton which cannot be accepted by any deterministic k-head automaton, appears to be very difficult.

We conclude this part of the paper by deriving a result which shows that adding more heads to an automaton increases its computational power and discussing several open problems.

Finally, we indicate the general applicability of the proof techniques used and indicate how they can be applied to the many-tape—two-tape problem for Turing machines.

Reduction to Machines with Three Heads

In this section we prove that if every non-deterministic 3-head automaton can be replaced with a deterministic automaton with (possibly) more heads, then the deterministic and non-deterministic context-sensitive languages are the same.

The proof proceeds through several reductions. The first reduction is by means of a known result [3].

Lemma. If every set accepted by a non-deterministic $L(n) = \log n$-tape bounded Turing machine can be accepted by a deterministic $L(n) = \log n$-tape bounded Turing machine, then the deterministic and non-deterministic context sensitive languages are the same.

Proof. For the sake of completeness we describe a $L(n) = \log n$-tape bounded Turing machine and outline the proof. A $L(n) = \log n$-tape bounded T.M. is a T.M. with two tapes: the input tape on which between end markers is placed the input and on which the T.M. can move a two-way, read-only head; the working tape of the T.M. is initially blank and on it the machine can move a two-way read-write head which cannot visit more than $[\log_2 n]$ tape squares for an input of length n. The accepting and rejecting of inputs for this machine is defined in the same way as for a k-head automaton. For a more detailed discussion of $\log n$-tape bounded T.M.'s see [2, 3, 4].

Assume that any non-deterministic $L(n) = \log n$-tape bounded T.M. can be replaced by a deterministic $L(n) = \log n$-tape bounded T.M. Let $A \subseteq \Sigma^*$ be a set accepted by a non-deterministic linearly bounded automaton M. We will show that then A can also be accepted by a deterministic l.b.a. To do this let $\#$ be a new symbol not in Σ and let

$$T(A) = \{w \mid w = u\#^t;\ u \in A \text{ and } |u| + t = 2^{|u|}\},$$

where $|u|$ designates the length of the sequence u. Thus we are just attaching enough markers to each sequence u in A to increase its length exponentially. Observe now that $T(A)$ can be accepted by a non-deterministic $\log n$-tape bounded Turing machine M_1. The machine M_1 operates as follows: for a given input w M_1

checks whether the input is of length $|w| = 2^p$, for some p, and whether it has the right form,

$$w = u\#^t \quad \text{and} \quad |u| = p.$$

This checking requires only $\log|w| \leqq p$ tape squares and is all done deterministically. If any of these conditions are not satisfied the input is rejected. If the format of w is found to be correct, then M_1 uses the

$$\log n = p = |u|$$

tape squares to simulate (behave like) M on the input u, and M_1 accepts the input $u\#^t$ iff M accepts u. Since M is linearly bounded we know that we have enough tape to simulate M and thus M_1 is a non-deterministic $\log n$-tape bounded T.M. which accepts $T(A)$. Therefore, by the hypothesis of the lemma, there exists a deterministic $\log n$-tape bounded T.M. which accepts the set $T(A)$; let it be denoted by M_2.

We now use M_2 to show that there exists a deterministic l.b.a. which accepts the set A. Let M_3 be a deterministic linearly bounded automaton which on input u simulates M_2 on input $u\#^p$, $p = 2^{|u|} - |u|$, and accepts input u iff M_2 accepts $u\#^p$. Thus M_3 accepts A. The simulation of M_2 in M_3 proceeds as follows: for input u M_3 behaves like M_2 as long as the read-head of M_2 stays on the input u. Note that M_2 can use for input $u\#^p$, $p = 2^{|u|} - |u|$, no more than

$$\log|u\#^p| = |u|$$

tape squares. Thus the input of length $|u|$ suffices for this part of the simulation. If the read-head of M_2 enters the tail of markers of the input $u\#^p$, then M_3 simply simulates those operations by counting on (say a special track) on the input u and this determines where the read-head of M_2 is and what it is doing. Again, on $|u|$ tape squares we can count up to the required number of markers, which is given by

$$2^{|u|} - |u|.$$

Thus M_3 is the desired deterministic linearly bounded automaton which accepts the set A. This completes the proof.

Next we establish the equivalence between $L(n) = \log n$-tape bounded T.M.'s and k-head automata which will be used to prove our main result. We first observe that any language accepted by a $L(n) = \log n$-tape bounded T.M. can be accepted by a k-head automaton which simply simulates the T.M. To see this, note that we can assume without loss of generality, that the working tape of the $\log n$-tape bounded T.M. consists of several binary tracks and that the last digit on the right end of the track is a one. Clearly, the binary number, which is written backwards on no more than $\log n$-tape squares on each track of the working tape of the T.M., can be represented as a head position on the input of length n of the k-head automaton which simulates the T.M. The k-head automaton has, besides the t heads needed to represent the content of the working tape of the T.M., 5 other heads. The first of these heads indicates by its distance from the left end of the input the distance of the read-write head from the left most binary symbol on the working tape of the T.M.; the seond head of the k-head automaton marks the tape square on the input tape which is beeing scanned at this instant by the read-head of the T.M., the remaining three of the additional five heads are used as bookkeeping heads by the automaton in the simulation of the Turing machine. For example, if the T.M. in an operation changes a zero to a one on a track of its working tape, then the read-head of the automaton representing this binary number as a distance on the input tape must be moved 2^p squares to the right, where p is the distance of the first read-head from the left end of the input of the automaton. Clearly, with the three bookkeeping heads the automaton can obtain from the distance p the distance 2^p and move the appropriate head 2^p squares to the right. Thus each operation of the T.M. can be simulated by the k-head automaton and an input will be accepted by the automaton iff it is accepted by the T. M. The simulation of a k-head automaton by a $L(n) = \log n$-tape bounded T.M. is straightforward and since these observations hold for deterministic as well as non-deterministic devices we get our next result.

Lemma. If any set accepted by a non-deterministic k-head automaton can also be accepted by a deterministic l-head automaton, then the deterministic and non-deterministic context-sensitive Language are the same.

Proof. The proof follows from the above observation and the previous lemma.

Next we show that it suffices to consider only 3-head non-deterministic automata.

Theorem. If every set accepted a non-deterministic 3-head automaton can be accepted by a deterministic k-head automaton, then the deterministic and non-deterministic context-sensitive languages are the same.

Proof. We will show that if every 3-head non-deterministic automaton can be replaced by a deterministic k-head automaton, then every non-deterministic automaton with read-only heads can be replaced by a deterministic automaton with more heads. Combining this result with the previous lemma we obtain a proof of this theorem.

Let $A \subseteq \Sigma^*$ be a set accepted by a non-deterministic k-head automaton M. Just as in the previous proof, we now construct a new set $P(A)$ from A which can be accepted by a 3-head non-deterministic automaton M_1. By the hypotheses of the theorem we then conclude that $P(A)$ is accepted by a deterministic automaton M_2 with more heads and we use this automaton to design a deterministic automaton M_3 to accept the set A.

$P(A)$ is obtained from A by attaching tails to sequences in A as described below. Let $a_1, a_2, \ldots, a_{k+1}$ be new symbols not in Σ and let

$$T(n) \subseteq \{a_1, a_2, \ldots, a_{k+1}\}^*$$

be constructed as follows:

1) take an n^k long sequence of a_1's,

2) replace the symbols in positions

$$n+1, 2n+1, 3n+1, \ldots, n^k - n + 1 \quad \text{by } a_2,$$

3) replace the symbols in positions

$$n^2+1,\ 2n^2+1,\ 3n^2+1, \ldots, n^k-n^2+1 \quad \text{by } a_3,$$

4) replace the symbols in positions

$$n^3+1,\ 2n^3+1,\ 3n^3+1, \ldots, n^k-n^3+1 \quad \text{by } a_4,$$

etc.

k) replace the symbols in positions

$$n^{k-1}+1,\ 2n^{k-1}+1,\ 3n^{2-1}+1, \ldots, n^k-n^{k-1}+1 \quad \text{by } a_k,$$

$k+1$) replace the first symbol by a_{k+1}.

The set $P(A)$ is defined by

$$P(A)=\{w \mid w=u\,T(|u|) \text{ and } u\in A\}.$$

To design a 3-head automaton M_1 which accepts the set $P(A)$ we will use one of the heads of this automaton on the tail $T(|u|)$ to encode the k head positions of M on the input u. By means of the other two "bookkeeping" heads M_1 will carry out the simulation of M on input u and accept $u\,T(|u|)$ iff M accepts u.

The encoding is achieved by assigning to the n^k tape squares of $T(n)$ n^k different k-tuples of distances. $(d_1, d_2, \ldots, d_k)$, with $1 \leqq d_i \leqq n$. The distance d_i of the r-th symbol of $T(n)$, $d_i(r)$, is assigned as follows: let $\sigma(r)$ denote the r-th symbol of the word $T(n)$ and let $p_i(r)$ be the largest j, $1\leqq j\leqq r$, such that the index of $\sigma(j)$ is larger than i; if no such j exists then $d_i(r)=1$, othervise $d_i(r)$ is equal to the number of positions s such that $p_i(r)\leqq s\leqq r$ and the index of $\sigma(s)$ is larger than or equal to i.

An arithmetic characterization of $d_i(r)$ can be obtained for r, $1\leqq r\leqq n^k$, if we let

$$r-1=r_1+r_2n^1+r_3n^2+\cdots+r_kn^k, \quad \text{with} \quad 0\leqq r_i\leqq n-1,$$

and set

$$d_i(r)=r_i+1.$$

It is seen that each tape square of the sequence $T(n)$ has a different k-tuple assigned to it and that if a head is placed on the square of $T(|u|)$ with distances

$$d_1, d_2, d_3, \ldots, d_k,$$

then using the bookkeeping heads M_1 can easily find (and remember) the d_i-th symbol of the sequence u for input $u\,T(|u|)$. Furthermore, if the head position has distances

$$d_1, d_2, d_3, \ldots, d_i, \ldots, d_k$$

then by means of the two bookkeeping heads M_1 can change it to the position with distances

$$d_1, d_2, d_3, \ldots, d_i\pm 1, \ldots, d_k.$$

To change d_1 to d_1+1 the automaton simply moves the encoding head one square to the right. To change d_i to d_i+1, for $1<i\leqq k$, the automaton places the

two bookkeeping heads on the nearest occurances of symbols a_p and a_q with $p > i$ and $q > i$ on the left and right of the encoding head, respectively. Now both bookkeeping heads are simultaneously moved to the right, one tape square at a time, until the left head coincides with the coding head (coincidence of head positions can be detected by the automaton). In this position the right bookkeeping head has distances

$$d_1, d_2, d_3, \ldots, d_i + 1, \ldots, d_k.$$

A similar method gives a position with distances

$$d_1, d_2, \ldots, d_i - 1, \ldots, d_k.$$

Using the above outlined procedures we see that the 3-head automaton M_1 can simulate on input $uT(|u|)$ the behavior of M on input u. Since M_1 accepts $uT(|u|)$ iff M accepts u we see that the 3-head automaton M_1 accepts the set $P(A)$.

By the hypotheses of the theorem we can replace this 3-head non-deterministic automaton M_1 by a t-head deterministic automaton M_2 which also accepts the set $P(A)$. To complete the proof we will now show that automaton M_2 can be used to prove that there exists a deterministic $(t+1) \cdot k$-head automaton M_3 which accepts the set A.

On input u the automaton M_3 will simulate the behavior of M_2 on input $uT(|u|)$ and accept u iff M_2 accepts $uT(|u|)$. This guarantees that M_3 is a deterministic automaton which accepts A. The simulation proceeds as follows: the first t heads of M_3 imitate the t heads of M_2 as long as M_2 keeps its heads on the u-part of the input $uT(|u|)$. When M_2 moves a head onto the $T(|u|)$-part of the input then M_3 uses k heads on input u, using their distance from the origin to encode the distance the M_2 head has travelled on $T(|u|)$. Since with k head positions on an input of length $n = |u|$ we can encode n^k numbers, we see that M_3 has enough heads to simulate the behavior of M_2 on input $uT(|u|)$ (recall that $|T(|u|)| = |u|^k$ and observe that each of the k-heads on u can represent a d_t distance of $T(|u|)$. Thus the simulation can be made quite direct.) Therefore M_3 accepts input u, after simulating M_2 on input $uT(|u|)$, iff M_2 accepts $uT(|u|)$ and thus we see that M_3 accepts the set A.

Thus we have shown that the hypotheses of the theorem implies that every non-deterministic k-head automaton can be replaced by a deterministic automaton with more heads. When this is combined with the preceding lemma we get the desired proof.

This result can easily be generalized to Turing machines which use more tape. We say that $L(n) \geq n$ is *tape constructable* iff there exists a T.M. which for each input of length n lays off $L(n)$ tape squares using only $L(n)$ tape squares.

Corollary. If every set accepted by a 3-head non-deterministic automaton can be accepted by a k-head deterministic automaton, then for every tape constructable $L(n)$ the deterministic and non-deterministic $L(n)$-tape bounded languages are the same.

A careful look at our proofs also shows that for $\log n$-tape bounded computations we get an if-and-only-if condition.

Corollary. Every set accepted by a non-deterministic 3-head automaton can be accepted by a deterministic k-head automaton if and only if the deterministic and non-deterministic $L(n) = \log n$-tape bounded languages are the same.

The preceding Theorem shows that a positive result about the power of non-deterministic 3-head automata could solve the classic linear bounded automata problem. What is surprising is that even for so simple a device as a 3-head automaton we are not at this time able to determine the power of non-determinacy.

It would be interesting to find out whether we can strengthen the result of the main theorem to an if-and-only-if condition as in the last corollary. Furthermore, it would be interesting to find out something about non-determinism in 2-head automata.

We conclude this section with simple result and discuss some other unsolved problems. The result shows that additional read-heads increase the power of an automaton.

Theorem. $C_D^k \subsetneq C_D^{2k+3}$.

Proof. In this proof we will use only automata with binary input alphabets and agree on a simple encoding of the state tables of all k-head deterministic automata with binary inputs (we use a straightforward tally notation to represent the $\left(\frac{1}{2}k(k-1)+2k+2\right)$-tuples of the state table of a machine: the $\left(\frac{1}{2}k(k-1)+2k+2\right)$-tuples contain the present state, the symbols read by the k-heads, the pattern of the read-head coincidences, the new state and the directions of motion for the k-heads; the zero symbol is used to separate these entries in the tuples, a double zero is used to separate different tuples and three zeros denote the end of this description of the automaton).

We now describe a $(2k+3)$-head automaton M which will accept a set not in C_D^k. M operates as follows: for input u M checks whether the front part of u has the format of a state table. If it does not u is rejected; otherwise, $u = vw$, where v describes a machine M^1, and M now uses $k+2$ heads to simulate what M^1 does on input $u = vw$. Note that k heads of M can be used just as M^1 uses its k heads and that two heads can determine from the state table description v what M^1 does in the given configuration. Thus $k+2$ heads of M suffice to simulate M^1. Unfortunately, M^1 may not halt and thus we cannot use the simulation to construct a set not in C_D^k by means of diagonalization. To overcome these difficulties M uses the $k+1$ remaining heads to count to $|u|^{k+1}$ and performs no more than $|u|^{k+1}$ simulation operations. If during this number of simulation operations M^1 halts and accepts or rejects, then M does the oposite; if M^1 does not halt in this number of operations than u is accepted.

Observe now that if $A \subseteq (0+1)^*$ is a set accepted by a k-head deterministic automaton M^1 with t states, then there exists arbitrary long inputs $u = vw$, where v describes M^1 and on which either M^1 halts in $t|u|^k$ operations or does not ever halt. Since for all sufficiently long sequences $u = vw$

$$t|u|^k < |u|^{k+1},$$

we see that for these long sequences M accepts u iff M^1 rejects and rejects it iff M^1 accepts. Thus M accepts a set not in C_D^k. This completes the proof.

We conjecture that

$$C_D^k \subsetneq C_D^{k+1} \quad \text{for} \quad k = 1, 2, 3, \ldots,$$

unfortunately, no proof has been obtained of this stronger hierarchy result.

It would be also interesting to determine whether for every $A \in C_D^k$ also $\bar{A} \in C_D^k$. Again we do not know the answer to this problem. If C_D^k should not be closed under complementation then it would be interesting to find out how many additional heads are needed to obtain the complements of all sets in C_D^k. We know that $2k$ heads suffice but we suspect that a considerably smaller number of heads is really required to accept the complements.

A further open problem is to determine whether any set $A \subseteq \{1\}^*$ which can be accepted by a non-deterministic k-head automaton can be accepted by a deterministic automaton with more heads. We have not been able to show that this result would imply the identity of the deterministic and non-deterministic context-sensitive languages.

Other Applications

The basic technique used in this paper can be summarized as follows: to determine whether some additional resource (non-determinism in our case) gives any more computing power to a given class of automata we can alter the accepted set A to $T(A)$ so that $T(A)$ is accepted by a weaker or simpler class of automata and then show that if the additional resource does not increase the computing power of these simpler automata then it cannot increase the computing power of the given class of automata.

In other words, by attaching the "tails" we simplified the set A, and thus have to deal with a simpler, and hopefully better understood set of computing devices. To further illustrate this technique we look at the problem of determining how much speed advantage can be gained in Turing machine computations by adding more tapes to the machine.

So far it is not known whether there exists a set A which can be accepted by a k-tape Turing machine in time $T(n)$ and not accepted in time $T(n)$ by any 2-tape T.M. (For definitions and related results see [2, 5, 6, 7].) We cannot solve this problem but we can show that if many-tape machines are faster than two-tape machines then there exist computations which are real-time for many-tape machines but not for two-tape machines. It can further be shown that if a speed advantage exists for the real-time computations, than this speed advantage bounds the possible speed advantage for longer computations.

Let $C_{T(n)}^k$ denote the class of all sets acceptable by k-tape T.M. in time bound $T(n)$. For definitions see [2, 7].

Theorem. If $C_n^2 = C_n^k$ and $T(n) \geqq n$ is the running-time of a one-tape Turing machine, then

$$C_{T(n)}^2 = C_{T(n)}^k.$$

Proof. We only give an outline. Let $A \in C_{T(n)}^k$ and $A \subseteq \Sigma^*$. Then we define

$$A^1 = \{w \mid w = u \#^{T(|u|) - |u|},\ u \in A\},$$

where $\sharp$ is a symbol not in Σ. We see that A^1 is in C_n^k and therefore by the hypotheses of the theorem A^1 is in C_n^2. Let M be the 2-tape T.M. which accepts A^1 in $T^1(n) = n$. Then we obtain from M a 2-tape T.M. M_1 which accepts the set A in time $T(n)$ as follows: for input u M_1 behaves like a one-tape T.M. with running time $T(n)$ (on one of its tapes) and writes on the input tape

$$u\sharp^{T(|u|)-|u|}.$$

This can be done in $c \cdot T(n)$ operations, after that M_1 behaves like M on this input, and can process this input in $T(n)$ operations. Thus from [8] we know that there exists a 2-tape T.M. which accepts A in time $T(n)$. Thus

$$C^2_{T(n)} = C^k_{T(n)},$$

as was to be shown.

Similarly, we can show for many other automata that any speed-advantages which can be gained by adding resources must already be present in simple computations (say, real-time). For example, under a somewhat more careful definition of $T(n)$, we can show that for Rasp's [9] the advantage gained for realtime computations from the addition of associative memory bounds the advantages which can be gained for $T(n)$ bounded computations. If no advantages can be gained from associative memories for real-time computations none can be gained for $T(n)$-bound computations.

References

1. Ginsburg, S.: The mathematical theory of context-free languages. New York: McGraw-Hill 1966.
2. Hopcroft, J. E., Ullman, J. D.: Formal languages and their relation to automata. Addison-Wesley, Reading, Mass., 1969.
3. Savitch, W. J.: Nondederministic tape-bounded Turing machines: Doctorial thesis, Univ. of Calif., Berkeley, Calif., 1969.
4. Hartmanis, J., Lewis II, P. M., Stearns, R. E.: Hierarchies of memory limited computations. IEEE Conference Record on Switching Circuit Theory and Logical Design, Ann Arbor, Michigan, p. 179–190, 1965.
5. Hennie, F. C., Stearns, R. E.: Two-tape simulation of multi-tape Turing machines. JACM **13**, 533–546 (1966).
6. Rabin, M. O.: Real time computation. Israel J. Math. **1**, 203–211 (1964).
7. Hartmanis, J., Hopcroft, J. E.: An overview of the theory of computational complexity. JACM **18**, 444–475 (1971).
8. Hartmanis, J., Stearns, R. E.: On the computational complexity of algorithms. Trans. Amer. Math. Soc. **117**, 285–306 (1965).
9. Hartmanis, J.: Computational complexity of random access stored program machines. J. Math. Systems Theory **5**, 232–245 (1971).

Juris Hartmanis
Gesellschaft für Mathematik
und Datenverarbeitung mbH Bonn
D-5205 St. Augustin, Schloß Birlinghoven
Postfach 1240
Federal Republic of Germany