THE JOURNAL OF SYMBOLIC LOGIC
Volume 55, Number 1, March 1990

EVERY RECURSIVE LINEAR ORDERING HAS A COPY IN DTIME-SPACE($n, \log(n)$)

SERGE GRIGORIEFF

§1. Introduction. This paper is a contribution to the following natural problem in complexity theory:

(*) *Is there a complexity theory for isomorphism types of recursive countable relational structures*? I.e. given a recursive relational structure $\mathscr{R}$ over the set **N** of nonnegative integers, is there a nontrivial lower bound for the time-space complexity of recursive structures isomorphic (resp. recursively isomorphic) to $\mathscr{R}$?

For unary recursive relations R, the answer is trivially negative: either R is finite or coinfinite or $\langle \mathbf{N}, R\rangle$ is recursively isomorphic to $\langle \mathbf{N}, \{x \in \mathbf{N}: x \text{ is even}\}\rangle$.

The general problem for relations with arity 2 (or greater) is open.

Related to this problem, a classical result (going back to S. C. Kleene [4], 1955) states that every recursive ordinal is in fact primitive recursive.

In [3] Patrick Dehornoy, using methods relevant to computer science, improves this result, showing that every recursive ordinal can be represented by a recursive total ordering over **N** which has linear deterministic time complexity relative to the binary representation of integers. As he notices, his proof applies to every recursive total order type α such that the isomorphism type of α is not changed if points are replaced by arbitrary finite nonempty subsets of consecutive points.

In this paper we extend Dehornoy's result to *all* recursive total orderings over **N** and get minimal complexity for both time and space simultaneously.

THEOREM. *Let a be an integer, $a \geq 2$. Every recursive total ordering $\langle \mathbf{N}, R\rangle$ has an isomorphic copy $\langle \mathbf{N}, S\rangle$ such that, relative to the a-ary representation of integers, S is computable in realtime using* log *space.*

Moreover, if the order type of R is not of the form $\omega + \mathbf{Z}.\alpha + \omega^*$ for some α, then one can suppose the isomorphism to be recursive.

Our proof proceeds in two steps.

First, we show (§4.1, theorem) that any recursive linear ordering $\langle \mathbf{N}, R\rangle$ has a recursively isomorphic copy $\langle \mathbf{A}, S\rangle$ such that **A** and S are in DTIME-SPACE ($n, \log(n)$). The idea is convert the algorithmic complexity of R into high dispersion of the domain **A**.

Received June 29, 1988; revised January 16, 1989.
Ce travail a bénéficié du soutien du PRC "Mathématiques et Informatique" du CNRS.

0022-4812/90/5501-0017/$02.70

Second, we show (§§5 and 6) how to fill the set $\mathbf{N}\backslash\mathbf{A}$. Two *exhaustive* cases are considered (cf. 5.4 and 6.1): First, we assume that $\langle \mathbf{N}, R\rangle$ has order type $\omega + \mathbf{Z}.\lambda + \omega^*$ for some λ. Second, we assume that there is an R-increasing sequence which is recursive and cofinal in $\langle \mathbf{N}, R\rangle$ or which has a supremum in $\langle \mathbf{N}, R\rangle$, or that there is an R-decreasing sequence which is recursive and coinitial in $\langle \mathbf{N}, R\rangle$ or which has an infimum in $\langle \mathbf{N}, R\rangle$. In the second case we get a recursive isomorphism.

REMARKS. 1°) This result is optimal in the following sense: a linear speed-up for space complexity—which does not affect the computation time—is always valid; however, realtime computability in space $o(\log)$ implies computability by finite automaton (cf. Stearns, Hartmanis and Lewis [9]; see also Wagner and Wechsung [11, Theorems 8.15 and 8.28]).

2°) In recent work, D. Cenzer and J. B. Remmel [1] give contributions to the above problem (*). In particular, they prove that there exists a recursive linear ordering on $\mathbf{N}$ with isomorphism type $\omega + \omega^*$ which is not *recursively* isomorphic to any polynomial time linear ordering with domain $\mathbf{N}$ (cf. also Remark 7.2). This has been generalized by Remmel (private communication) to linear orderings with isomorphism type $\omega + \mathbf{Z}.\lambda + \omega^*$.

Thus, the second assertion in Theorem 7.1 cannot be improved.

3°) For related questions, cf. also R. Watnick [10], M. Moses [5] and J. B. Remmel [6], [7].

§2. Some definitions.

2.1. All Turing machines considered below have semi-infinite input tapes and worktapes. In order to be recognized during the computation, the first cell of each tape receives special symbols which duplicate those (including the blank one) used for the other cells.

2.2. The length of a word u is denoted $|u|$.

DEFINITION. A Turing machine M with m input tapes (and some more worktapes and output tapes) is said to be *realtime* if it is deterministic and if the following conditions hold:

a) The head of each input tape of M constantly moves rightwards until it meets the first blank to the right of the input. Afterwards, it stays still.

b) When the heads of all the input tapes scan a blank cell then M enters a final state and stops.

Thus, if $u_1, \ldots, u_m$ are the initial contents of the input tapes of a realtime Turing machine, the whole computation involves exactly $\operatorname{Sup}\{|u_1|, \ldots, |u_m|\}$ computation steps.

In the special case where there is no worktape nor output tape, M is merely a finite automaton.

2.3. The following definition presents several complexity classes (some classical, others less usual).

DEFINITION. Let Σ be a finite alphabet and Σ^* be the set of all words over Σ. Let t, s and α be nondecreasing functions over $\mathbf{N}$ such that $t(n) \geq \operatorname{Sup}\{\alpha(n), s(n), n\}$ for all n.

1°) An m-ary function F with domain a subset D of some $[\Sigma^*]^m$ and values in Σ^* has *deterministic time-space complexity* (t, s) (or, briefly, is in the class DTIME-SPACE(t, s)) if there exists a deterministic Turing machine M with m input tapes, an

output tape (and some more worktapes) such that, for all inputs $u_1,\dots,u_m$ distributed on these m input tapes:

i) M halts within no more than $t(\mathrm{Sup}\{|u_1|,\dots,|u_m|\})$ computation steps.

ii) The head of each input tape moves between cell 1 and the first cell to the right of the input and can change no symbol.

iii) M uses no more than $s(\mathrm{Sup}\{|u_1|,\dots,|u_m|\})$ cells on its output tape and worktapes.

iv) M halts in an accepting state if and only if $(u_1,\dots,u_m)$ is in D.

v) When (and if) M halts in an accepting state, the contents of its output tape is the word $F(u_1,\dots,u_m)$.

2°) We say that F has *deterministic time-space complexity* (t,s) *with anticipation of the output at time* α (or, briefly, is in the class DTIME-SPACE(t,s)-ANTICIP(α)) if there exists a deterministic Turing machine M satisfying conditions i) to v) above and such that, for some subset P of the set of states of M, called the set of prefinal states, which contains the set of final (accepting or rejecting) states, the following conditions hold:

vi) When (and if) M enters a prefinal state then all subsequent states are prefinal and the contents of the output tape remains unchanged until M halts.

vii) If $(u_1,\dots,u_m)$ is in the domain D then M enters some prefinal state after at most $\alpha(\mathrm{Sup}\{|u_1|,\dots,|u_m|\})$ computation steps.

Thus, the word $F(u_1,..,u_m)$ is written on the output tape as soon as machine M gets in the prefinal states (which are distinguished ones), and this happens at time no greater than $\alpha(\mathrm{Sup}\{|u_1|,\dots,|u_m|\})$.

3°) F has *realtime-space s complexity* (or, briefly, is in the class REALTIME-SPACE(s)) if there exists a *realtime* Turing machine M with m input tapes (and some more worktapes) satisfying conditions iii) to v) above.

4°) The function F has *realtime-space s complexity with anticipation of the output at time* α (or, briefly, is in the class REALTIME-SPACE(s)-ANTICIP(α)) if there exists a *realtime* Turing machine M satisfying conditions iii) to vii) above.

In particular, the length of $F(u_1,\dots,u_m)$ is at most $\alpha(\mathrm{Sup}\{|u_1|,\dots,|u_m|\})$.

5°) An m-ary relation R over Σ^* (or a subset of Σ^*) is in the class DTIME-SPACE(t,s) if there exists a deterministic Turing machine M with m input tapes (and some more worktapes) satisfying conditions i), ii), iii) above and the condition

iv*) M halts in an accepting state if and only if $(u_1,\dots,u_m)$ is in R.

R is in the class REALTIME-SPACE(s) if there exists a realtime Turing machine M with m input tapes (and some more worktapes) satisfying conditions iii) and iv*).

6°) For functions and relations dealing with integers, the above classes are defined *relative to the representation of integers within some base* $a \geq 2$, i.e. over the alphabet $\{0,\dots,a-1\}$.

2.4. REMARK 1. Due to the linear speed-up for space complexity (which does not affect the computation time), condition iii) in 2.3.1°) above can be replaced by

iii*) M uses no more than $s(\mathrm{Sup}\{|u_1|,\dots,|u_m|\})$ cells on *each one* of its output tape and worktapes.

REMARK 2. For low time complexities, in particular for REALTIME complexity, the base dependence in 6°) of the above definition seems a priori necessary (see A. Cobham [2] for a related result which, due to a result in [9], can be viewed as a base dependence for *space* complexity $o(\mathrm{loglog})$).

REMARKS 3.1°) If a function F is in one of the above classes then its domain D, considered as an m-ary relation over Σ^* or $\mathbf{N}$, also belongs to this class.

2°) If a total function (i.e. with domain some $[\Sigma^*]^m$ or $\mathbf{N}^m$) is in DTIME-SPACE(t, s)-ANTICIP(α), then it is in fact in DTIME-SPACE(α, s).

3°) The class REALTIME-SPACE(s) and the class REALTIME-SPACE(s)-ANTICIP(α) are included in DTIME-SPACE($n, s(n)$) for all α.

4°) Let m be a positive integer. The class of m-ary relations which belong to REALTIME-SPACE(s) is closed under Boolean operators.

2.5. It will be convenient in the sequel to consider an m-ary function F with domain a subset D of $\mathbf{N}^m$ as an *m-ary $\mathbf{N}$-valued relation over D*. Though formally the very same objects, they have to be distinguished as concerns the classical notions of transportation of structures via a bijection Φ from $\mathbf{N}$ onto a set $\mathbf{A}$:

a) If F is considered as a *function* (with partial domain D) in the structure $\langle \mathbf{N}, F\rangle$ then, in order that Φ be an isomorphism between $\langle \mathbf{N}, F\rangle$ and $\langle \mathbf{A}, \Phi(F)\rangle$, the image $\Phi(F)$ has to be the function (with partial domain $\Phi(D)$) such that Graph($\Phi(F)$) is the image of Graph(F) via the bijection $(\Phi, \ldots, \Phi, \Phi)$ from $\mathbf{N}^{m+1}$ onto $\mathbf{A}^{m+1}$. Thus, $\Phi(F) = \Phi \circ F \circ (\Phi^{-1}, \ldots, \Phi^{-1}) \restriction (\Phi, \ldots, \Phi)(\mathbf{D})$.

b) If F is considered as an *$\mathbf{N}$-valued relation* over D in the structure $\langle \mathbf{N}, F\rangle$ then, in order that Φ be an isomorphism between $\langle \mathbf{N}, F\rangle$ and $\langle \mathbf{A}, \Phi(F)\rangle$, the image $\Phi(F)$ has to be the $\mathbf{N}$-valued relation over the subset $\Phi(D)$ such that Graph($\Phi(F)$) is the image of Graph(F) via the bijection $(\Phi, \ldots, \Phi, \mathrm{Id})$ from $\mathbf{N}^{m+1}$ onto $\mathbf{A}^m \times \mathbf{N}$, where Id is the identity function over $\mathbf{N}$. Thus, $\Phi(F) = F \circ (\Phi^{-1}, \ldots, \Phi^{-1}) \restriction (\Phi, \ldots, \Phi)(D)$.

Ordinary relations can be considered as $\mathbf{N}$-valued relations with values in $\{0, 1\}$.

The notions of time-space complexity for $\mathbf{N}$-valued relations are exactly those of the corresponding functions.

§3. Some technical facts.

3.1. We use the notations Bin(p), 0^p and $u \hat{} v$ to denote the binary representation of the integer p, the word consisting of p successive letters 0, and the word obtained by concatenating u and v. We denote by ω (resp. ω^*, resp. $\mathbf{Z}$, resp. $\mathbf{Q}$) the order types of the set of positive integers (resp. negative integers, resp. arbitrary integers, resp. rational numbers).

We denote by $\alpha.\beta$ the order type of the sum of a family of copies of α indexed by the ordered set β, i.e. the order type of the *right to left* lexicographical ordering on the cartesian product $\alpha \times \beta$:

$$(a, b) < (a', b') \quad \text{if and only if} \quad b < b' \text{ or } b = b' \text{ and } a < a'.$$

We shall need the following fact in §5.4.

PROPOSITION. *Let a be an integer, $a \geq 2$. There exist total orderings over $\{0, 1, \ldots, a-1\}^*$ (resp. $\mathbf{N}$) which have respective order types $\mathbf{Q}$, $\mathbf{Z}.\mathbf{Q}$ and $\omega + \mathbf{Z}.\mathbf{Q} + \omega^*$ and are recognizable by finite automata, i.e. are in* REALTIME-SPACE(0) *(resp. relative to the a-ary representation of integers).*

PROOF. 1°) We restrict to the case $a = 2$. Let I be the set of dyadic rational numbers in $]-1, +1[$. We define two bijections $\varphi: \{0, 1\}^* \to I$ and $\psi: \mathbf{N} \to I \cup \{1/3\}$ as follows:

$\varphi(\text{empty word}) = 0$ and $\varphi(a_1 a_2 \cdots a_n) = \sum\{(2a_i - 1)/2^i : 1 \leq i \leq n\}$ with $a_i \in \{0, 1\}$ for all i. $\psi(0) = 1/3$ and $\psi(x) = \varphi(u)$ if $x \in \mathbf{N} \setminus \{0\}$ and $\mathrm{Bin}(x) = 1 \hat{} u$.

Let R and S be the relations

$$R = \{(u, v) \in [\{0,1\}^*]^2 : \varphi(u) < \varphi(v)\},$$
$$S = \{(x, y) \in \mathbf{N}^2 : \psi(x) < \psi(y)\}.$$

The structures $(\{0,1\}^*, R)$ and $(\mathbf{N}, S)$ are dense total orderings without endpoints, hence isomorphic to the ordered structure of the set of rational numbers.

It is easy to see that that R is recognizable by a finite automaton. As for the binary representation of S, notice that $\psi(0) = 1/3 = \sum\{(-1)^i/2^{i+1} : i \in \mathbf{N}\}$ = limit of the sequence $(\varphi[(10)^p])_{p \in \mathbf{N}}$.

2°) To get orderings with order type $\mathbf{Z}.\mathbf{Q}$, one can consider the following bijections:

$$\xi : \{0,1\}^* \to \mathbf{Z} \times [\mathrm{I} \cup \{1/3\}] \quad \text{and} \quad \eta : \mathbf{N} \to \{(1/2, 0)\} \cup \mathbf{Z} \times [\mathrm{I} \cup \{1/3\}].$$

Here $\xi(1^p) = ((-1)^p\lceil p/2\rceil, 1/3)$, where p may be zero, $\xi(1^p \hat{} 0u) = ((-1)^p\lceil p/2\rceil, \varphi(u))$, $\lceil p/2\rceil$ being the smallest integer greater than or equal to $p/2$, $\eta(0) = (1/2, 0)$, and $\eta(x) = \xi(u)$ if $x \in \mathbf{N}\backslash\{0\}$ and $\mathrm{Bin}(x) = 1 \hat{} u$.

3°) Easy modifications of the above bijections lead to orderings with order type $\omega + \mathbf{Z}.\mathbf{Q} + \omega^*$.

3.2. DEFINITION. Let λ be the injection from $\{\mathrm{Bin}(x) : x \in \mathbf{N}\}$ into $\{00, 11, 01\}^*$ defined by

$$\lambda(a_1 a_2 \cdots a_n) = (10)^{\Delta(1)} \hat{} a_1 a_1 \hat{} (10)^{\Delta(2)} \hat{} a_2 a_2 \hat{} \cdots \hat{} (10)^{\Delta(n)} \hat{} a_n a_n$$

where $\Delta(1) = 1$, $\Delta(i) = i2^i$ for $i \geq 2$.

The reason to introduce such a function λ is as follows:

a) A sequence of words can be trivially coded using concatenation and a new letter as separator.

b) Due to the base dependence stressed in 2.4, Remark 2, we want to keep codes within the $\{0, 1\}$ alphabet. Coding 0 and 1 as 00 and 11, we get 01 and 10 as "auxiliary letters". This leads us to code $a_1 a_2 \cdots a_n$ as $a_1 a_1 a_2 a_2 \cdots a_n a_n$.

(c) Such codes give rise to a realtime decoding function which requires linear space (since we *do* count space on the output tape). In order to get a log-space complexity, we have to lengthen codes so as to make them exponentially long. This can be done using the "letter" 10.

d) However, it is not sufficient to add long suffixes: words which are not codes also have to be recognized within log-space. This requires that from a given prefix with length k of the code of u, we can get at most $\log(k)$ bits of u.

e) This is the reason for inner padding factors: for any i, the space needed to store the prefix $a_1 a_2 \cdots a_i$ and move a head right to it (i.e. $i + 1$ cells) has to be at most the log of the length of the part of the input already read. This leads us to code $a_1 a_2 \cdots a_n$ as $\lambda(a_1 a_2 \cdots a_n)$ with Δ such that $i + 1 \leq \log[|\lambda(a_1 a_2 \cdots a_i)|]$ for all i. The particular choice of Δ is a mere convenience.

PROPOSITION. *The function* λ^{-1} (*with domain* $\mathrm{Range}(\lambda)$) *is in* REALTIME-SPACE(log).

PROOF. 1°) We first define a "wasting time" one-tape Turing machine A with input alphabet $\{0, 1\}$.

a) Started with its head scanning cell $i + 1$, the computation of A involves $2i \times 2^i$

steps, and the final contents of the tape and the position of the head are exactly the initial ones.

b) The working alphabet of A is $\{(0,0),(0,1),(1,0),(1,1)\}$, so as to get two levels on the tape.

c) The action of A is to write the successive integers $1,2,\ldots,2^i-1$ as binary words with length i on the second level of cells 1 through i, retaining in the first level the initial contents. The head moves from cell $i+1$ to cell 1 to perform the incrementation and then moves back to cell $i+1$.

d) During its very last return, A restores the initial contents (memorized in the first level).

2°) We describe below a *realtime* Turing machine L with one input tape and one output tape which computes λ^{-1} within log-space.

a) A finite automaton mechanism allows L

◆ to analyse the input w as a concatenation of successive factors:

$$w = (10)^{k_1}\hat{}\,a_1a_1\hat{}\,(10)^{k_2}\hat{}\,a_2a_2\hat{}\cdots\hat{}\,(10)^{k_n}\hat{}\,a_na_n\hat{}\,w',$$

where $n \geq 0$ and 10 is not a prefix of the word w'; and

◆ to check if $a_1a_2\cdots a_n$ is in $\{\mathrm{Bin}(x)\colon x \in \mathbf{N}\}$ (i.e. $n \neq 0$ and neither 00 nor 01 is a prefix).

b) If L is not in the dead state then the input factor a_ia_i is read while the head of the output tape scans cell i; then L copies a_i on the output tape, moves its output head onto cell $i+1$ and starts to simulate A on its output tape.

c) This simulation of A allows L to compare k_i and $\Delta(i)$ (recall that L, being realtime, constantly moves its input head rightwards). If $k_i \neq \Delta(i)$ then L enters a dead state which leads to rejection and forbids any further move of the output head.

d) L halts in an accepting state if and only if it is not in the dead state and the factor w' is empty.

3.3. PROPOSITION. *For every recursive function t from $\mathbf{N}$ into $\mathbf{N}$ there exists a strictly increasing function $t^\#$ such that $t \leq t^\#$ and*

$$\mathrm{DTIME}(t) \subseteq \mathrm{DTIME\text{-}SPACE}(t^\#, \log(t^\#))$$

*and a deterministic one-tape Turing machine $M_{t^\#}$ which works on any input $v \in \{0,1\}^*$ in time **exactly** $t^\#(|v|)$, visiting (at most) $\log(t^\#(|v|))$ cells.*

PROOF. Let M be a one-tape machine which, on input v, computes the unary representation of $t(|v|)$. One can suppose that the computation time of M is strictly increasing with the length of the input.

Machine $M_{t^\#}$ is a one-tape machine which alternatively simulates one step of machine M and then the whole computation of machine A (described in the preceding proof).

We define $t^\#(k)$ as the computation time of $M_{t^\#}$ on inputs of length k.

Let $\mathbf{N}$ be a machine working in time t. Modifying $M_{t^\#}$ so as to simulate $\mathbf{N}$ during the sole computation steps when $M_{t^\#}$ simulates M, we get a machine simulating $\mathbf{N}$ and working in time $t^\#$ and space $\log(t^\#)$.

§4. Representation of a recursive structure over a sparse subset of N. The following general theorem has been independently obtained by D. Cenzer and J. Remmel [1] in a weaker form dealing solely with linear time complexity.

4.1. THEOREM. *Let α and t be nondecreasing recursive functions from* **N** *into* **N**, *and* $K > 0$. *We suppose that α is unbounded and that $\alpha(n) \leq n$ for all n. There exists a recursive and increasing bijection Φ between* **N** *and a subset* **A** *of* **N** *such that*:

1°) i) *The set* **A** *is in* REALTIME-SPACE(log).

ii) *Φ maps any* **N**-*valued relation (over a subset of* $\mathbf{N}^m$) *which is in* DTIME(t) *onto an* **N**-*valued relation (over a subset of* $\mathbf{A}^m$) *which is in* REALTIME-SPACE(log)-ANTICIP(α). [*I.e. if* $D \subseteq \mathbf{N}^m$ *and* R: $D \to \mathbf{N}$ *and R (hence also D) is in* DTIME(t), *then* $S = R \circ (\Phi^{-1}, \ldots, \Phi^{-1}) \restriction (\Phi, \ldots, \Phi)(D)$ *is in* REALTIME-SPACE(log)-ANTICIP(α). *In particular, Φ maps sets in* DTIME(t) *onto sets in* REALTIME-SPACE(log).]

iii) *The integers $\Phi(0)$ and $\Phi(x+1) - \Phi(x)$, x in* **N**, *are all greater than K.*

2°) *In particular, let $\mathscr{R} = \langle X, R_1, \ldots, R_p \rangle$ be an* **N**-*valued relational structure with domain X included in* **N** *and such that $X, R_1, \ldots, R_p$ are all in* DTIME(t). *Then, the restriction F of Φ to X defines an isomorphism from $\mathscr{R}$ onto an* **N**-*valued structure $\mathscr{S} = \langle Y, S_1, \ldots, S_p \rangle$ such that*

i) **A** *is in* REALTIME-SPACE(log) *and $S_1, \ldots, S_p$ are in* REALTIME-SPACE(log)-ANTICIP(α), *and*

ii) $F(x) > K$ *and* $F(y) - F(x) > K$ *for all x and y in X such that* $x < y$.

Subsections 4.3 to 4.7 are devoted to a proof of part 1°) of this proposition.

4.2. REMARK. The above theorem is false if functions are considered (instead of **N**-valued relations). For example, let Suc be the successor function. The graph of Suc is a relation in REALTIME. However, for any isomorphic copy $\langle \mathbf{A}, \sigma \rangle$ of the structure $\langle \mathbf{N}, \mathrm{Suc} \rangle$—where Suc is considered as a function and not as an **N**-valued relation—there exists x in **A** such that $|\mathrm{Bin}(\sigma(x))| > |\mathrm{Bin}(x)|$, hence $\sigma(x)$ cannot be written (hence computed) in time $|\mathrm{Bin}(x)|$.

4.3. Recall that $u \hat{} v$ denotes the word obtained by concatenating u and v. Let $t^{\#}$ and $M_{t^{\#}}$ be defined from t as in Proposition 3.3.

Since the function α (considered in the statement of the proposition) is recursive and *unbounded*, one can define a strictly increasing recursive function β over **N** as follows:

$$\beta(p) = \text{the smallest integer } q \text{ such that } \alpha(q) \geq 4p2^p + t^{\#}(p) + K,$$

where K is the constant in the statement of Theorem 4.1. Let $\beta^{\#}$ and $M_{\beta^{\#}}$ be defined from β as in Proposition 3.3.

Since α is nondecreasing and $\beta \leq \beta^{\#}$, it is clear that $\beta^{\#}$ satisfies $\alpha(\beta^{\#}(p)) \geq 4p2^p + t^{\#}(p) + K$.

4.4. We define a recursive, injective and increasing function Φ from **N** into **N** as follows: $\Phi(x)$ is the integer with binary representation

$$\mathrm{Bin}(\Phi(x)) = \lambda(\mathrm{Bin}(x)) \hat{} 01 \hat{} 1^{|\mathrm{Bin}(x)|} \hat{} 0^{t^{\#}(|\mathrm{Bin}(x)|)} \hat{} 1^{\beta^{\#}(|\mathrm{Bin}(x)|)}.$$

We let **A** be the range of Φ.

The definition of Φ is motivated by the following idea:

(◆) When $\mathrm{Bin}(\Phi(x))$ is read by some adequate Turing machine, then

a) $\lambda(\mathrm{Bin}(x))$ can be recovered as the longest prefix in $\{00, 11, 10\}^*$,

b) $\mathrm{Bin}(x)$ can be recovered from $\lambda(\mathrm{Bin}(x))$ and stored on some tapes,

c) the heads on these tapes can be back while the factor $1^{|\mathrm{Bin}(x)|}$ is read,

d) any computation involving $t(|\mathrm{Bin}(x)|)$ steps can be simulated in space

$\log[t^{\#}(|\mathrm{Bin}(x)|)]$ while the factor $0^{t^{\#}(|\mathrm{Bin}(x)|)}$ is being read, and

e) due to a pertinent choice of β, the reading of the suffix $1^{\beta^{\#}(|\mathrm{Bin}(x)|)}$ lasts long enough so that the output of the preceding simulation is obtained with convenient anticipation.

4.5. Since $\beta^{\#}(p) \geq \alpha(\beta^{\#}(p)) \geq K$, there are at least K digits to the right of $\lambda(\mathrm{Bin}(x))$ in $\mathrm{Bin}(\Phi(x))$.

Thus $\Phi(0) \geq 2^K \geq K$ and $\Phi(x+1) - \Phi(x) \geq 2^K > K$, which proves iii) in Theorem 4.1.1°).

4.6. Let L be a realtime Turing machine with one input tape and one output tape which computes λ^{-1} within log-space (cf. 3.2).

We describe a realtime Turing machine U, with one input tape and two worktapes, recognizing **A**. On input word u from alphabet $\{0, 1\}$, machine U acts as follows:

i) A finite automaton mechanism allows U to analyse the input u as a concatenation of successive factors: $u = u_1 \hat{} u_2$ or $u = u_1 \hat{} 01 \hat{} 1^p \hat{} 0^q \hat{} 1^r \hat{} u_3$, where p, q, r are in **N**, u_1 is the longest prefix of u lying in $\{00, 11, 10\}^*$, $|u_2| \leq 1$ and 1 is not a prefix of u_3.

ii) If $u = u_1 \hat{} u_2$ then machine U stops and rejects (recall that machine U is realtime, hence stops when the input has been read). If u_3 is not empty then U enters a dead state, which leads to rejection and forbids any further move of the worktape heads.

iii) While the input head reads the factor u_1, U simulates L on input u_1 and copies the output v on both worktapes 1 and 2.

iv) If L rejects then machine U enters the above dead state. If L accepts, then worktapes 1 and 2 contain v and their heads scan the last letter of v. Then, machine U checks whether p, q, r are equal to $|v|$, $s^{\#}(|v|)$ or $\beta^{\#}(|v|)$, as follows:

a) While the input head reads the factor 1^p, machine U moves its worktape heads leftwards. The equality $p = |v|$ holds if and only if the heads of worktapes 1 and 2 reach the first letter of v at the very same computation step when the head of the input tape of U leaves the factor 1^p.

b) While the input head reads the factor 0^q, machine U simulates on worktape 1 the computation of machine $M_{t^{\#}}$ (with input v). The equality $q = s^{\#}(|v|)$ holds if and only if this simulation is completed exactly when the head of the input tape leaves the factor 0^q.

c) While the input head reads the factor 1^r, machine U simulates on worktape 2 the computation of machine $M_{\beta^{\#}}$ (with input v). The equality $r = \beta^{\#}(|v|)$ holds if and only if this simulation is completed exactly when the head of the input tape leaves the factor 1^r.

v) If and when one of these three equalities is recognized not to be valid, then either the input head reads a blank and machine U stops and rejects, or machine U enters the dead state. If all equalities hold—and if u_3 is empty (cf. ii))—then U halts and accepts.

It is clear that machine U uses no more than $\log(|u|)$ cells on each of its worktapes and recognizes the set $\{\lambda(v) \hat{} 01 \hat{} 1^{|v|} \hat{} 0^{t^{\#}(|v|)} \hat{} 1^{\beta^{\#}(|v|)}$: v is in $\{\mathrm{Bin}(x)$: $x \in \mathbf{N}\}\}$, i.e. the set **A**. Using 2.4, Remark 1, this proves that **A** is in REALTIME-SPACE(log).

4.7. We now show that Φ maps any m-ary **N**-valued relation R lying in DTIME(t) onto an **N**-valued relation $\Phi(R)$ lying in REALTIME-SPACE(log)-ANTICIP(α).

Since DTIME(t) $\subseteq$ DTIME-SPACE($t^{\#}$, log($t^{\#}$)) (cf. Proposition 3.3), there exists a Turing machine P which computes R within time $t^{\#}$ and space log($t^{\#}$). Let P have m input tapes, p worktapes and one output tape.

Let U be the above *realtime* Turing machine which recognizes the set **A** and works in log-space.

Developing the ideas sketched in 4.4 (◆), we describe a *realtime* Turing machine V, with m input tapes, $3m + p$ worktapes and one output tape, which computes the **N**-valued relation S with anticipation of the output at time α.

On input words $u_1, \ldots, u_m$, distributed on the m input tapes, machine V acts as follows:

i) The ith input tape and two worktapes are devoted to the simulation of the computation of U on input u_i. Thus, V can check whether $u_1, \ldots, u_m$ are in $\{\text{Bin}(x)\colon x \in \mathbf{A}\}$. If V has to simulate a dead state or a rejection of U on some of the u_i's, then V enters a dead state.

ii) For $i = 1, \ldots, m$, phases iii) and iv)a) of the simulations of U on input u_i are duplicated on a devoted worktape : v_i is written and the head is then moved back to cell 1; then the head stays still up to the moment when all heads of these duplication tapes scan their first cells. If all inputs u_i are in **A**, this happens at time step $\text{Sup}\{|\lambda(v_i)\hat{}01\hat{}1^{|v_i|}|\colon i = 1, \ldots, m\}$.

iii) Then machine V starts simulating machine P on inputs $v_1, \ldots, v_m$ written on these duplication tapes. This uses p more worktapes; the output tape of V simulates that of P.

iv) Being realtime, machine V stops when all inputs $u_1, \ldots, u_m$ have been read. It stops in an accepting state if and only if the m simulations of U on inputs $u_1, \ldots, u_m$ all lead to accepting U-states and that of P on inputs $v_1, \ldots, v_m$ leads to an accepting P-state. The output of V is exactly that one which comes from the simulation of P.

W now prove that V computes $\Phi(R)$ in log-space.

A) If some input u_i is not the representation of an integer in **A**, then machine V rejects.

B) Suppose now that each input u_i is in **A**. Let $u_i = \text{Bin}(y_i)$, $y_i = \Phi(x_i)$ and $v_i = \text{Bin}(x_i)$. Then V starts the simulation of P on inputs $v_1, \ldots, v_m$ when it starts reading the factor $0^{t^{\#}(|v_i|)}$ of the longest input u_i. Since P works in time $t^{\#}$, this simulation can be completed while V reads this factor. Thus, V does output the value $R(x_1, \ldots, x_m)$, hence computes $\Phi(R)(y_1, \ldots, y_m)$.

C) V delivers its output computation while reading the factor $0^{t^{\#}(|v_i|)}$ of the longest input u_i. Thus, this output is obtained by time τ, where

$$\tau \leq \text{Sup}\{|\lambda(v_i)\hat{}01\hat{}1^{|v_i|}|\colon i = 1, \ldots, m\} + t^{\#}(\text{Sup}\{|v_i|\colon i = 1, \ldots, m\}).$$

Let $n = \text{Sup}\{|v_i|\colon i = 1, \ldots, m\}$.

From the definition of λ (cf. 3.2), we get $|\lambda(v)| \leq 4|v|(2^{|v|} - 1)$ whence,

$\tau \leq 4n(2^n - 1) + n + 2 + t^{\#}(n) \leq 4n2^n + t^{\#}(n)$,

$\tau \leq \alpha(\beta^{\#}(n))$ by the very definition of β (cf. 4.3),

$\tau \leq \text{Sup}\{\alpha(\beta^{\#}(|v_i|))\colon i = 1, \ldots, m\}$ since α and $\beta^{\#}$ are increasing,

$\tau \leq \alpha(\text{Sup}\{|u_i|\colon i = 1, \ldots, m\})$ since $\beta^{\#}(|v_i|) \leq |\text{Bin}(\Phi(x_i))| = |u_i|$ for all i.

Thus, machine V delivers the value $R(x_1, \ldots, x_m)$ with anticipation at time no greater than $\alpha(\text{Sup}\{|u_i|\colon i = 1, \ldots, m\})$.

D) It is clear that machine U uses no more than $\log(\mathrm{Sup}\{|u_i|: i = 1,\ldots,m\})$ cells on each one of its worktapes.

Points A) to D) (and 2.4, Remark 1) prove that **A** is in REALTIME-SPACE(log). This completes the proof of Theorem 4.1.

§5. Restricted versions of the main theorem.

5.1. From Theorem 4.1 we know that it is possible to get a copy in REALTIME-SPACE(log) of any recursive, relational or **N**-valued relational structure over **N**, the domain of the copy being a sparse subset of **N**.

The problem now is to go from such a sparse domain back to the domain **N**. Our strategy to do this is as follows:

1) Isolate a "very simple" infinite part X of the original structure over **N**.

2) Apply Theorem 4.1 to the restriction to $\mathbf{N}\backslash X$ of the original structure, thus getting a copy in REALTIME-SPACE(log) of this restriction over a sparse (hence coinfinite) subset **A** of **N**.

3) Copy the restriction to X of the original structure over the set $\mathbf{N}\backslash\mathbf{A}$.

If the "very simple" infinite part X is also simply related to the $\mathbf{N}\backslash X$ part, then the two structures obtained in 2) and 3) can be combined into a REALTIME-SPACE(log) copy over **N** of the original structure.

5.2. First, we give a direct application of this method to the case of recursive ordinals which (strengthens and) gives another proof of Dehornoy's result [3].

THEOREM. *Let a be an integer, $a \geq 2$. Every infinite recursive ordinal α can be represented by a total ordering S over* **N** *which is in* REALTIME-SPACE(log) *relative to the a-ary representation of integers.*

PROOF. 1°) The case $\omega \leq \alpha \leq \omega + \omega$ is obvious.

2°) If $\alpha \geq \omega + \omega$ then $\alpha = \omega + \beta$, where β is an infinite recursive ordinal. Using Theorem 4.1 with a recursive ordered structure $\langle \mathbf{N}, R\rangle$ of type β, we get a total ordering S' over a coinfinite subset **A** of **N** such that **A** and S' are in REALTIME-SPACE(log) and $\langle \mathbf{A}, S'\rangle$ has type β.

Let $S = <\restriction \mathbf{N}\backslash\mathbf{A} \cup [(\mathbf{N}\backslash\mathbf{A}) \times \mathbf{A}] \cup S'$ (where $<$ denotes the restriction of the usual ordering). It is clear that S is in REALTIME-SPACE(log). Also, S is a total ordering over **N** for which $\mathbf{N}\backslash\mathbf{A}$ is an initial segment and **A** is a final segment. Thus, S has order type $\omega + \beta$, i.e. α.

REMARKS. 1°) It is important to notice that if ρ is a recursive total ordering over **N** which has an initial segment of type ω, then this initial segment is *not* necessarily recursive. In fact, using a priority argument, S. Tennenbaum has proved (see J. G. Rosenstein [8, p. 453], or R. Watnick [10]) that there is a recursive copy of $\omega + \omega^*$ in which the ω part is not recursive.

2°) Of course, in case ρ is an ordinal, then every proper initial segment has a supremum, hence is of the form $\{x: \rho(x, b)\}$ for some b and so is recursive.

5.3. The above proof can be applied to get the following particular extension of Theorem 5.2:

THEOREM. *Let $\langle \mathbf{N}, R\rangle$ be a recursive ordering for which there exists a partition of* **N** *into three recursive subsets U, V, W such that U is an initial segment of R, W is a final segment of R, and the restriction of R to V has order type $\omega + \omega^*$. Then there exists a* REALTIME-SPACE(log) *total ordering S over* **N** *which is isomorphic to R.*

PROOF. Suppose $U \cup W$ is infinite (otherwise the result is trivial).

Applying Theorem 4.1 (with $K > 2$) to the relational structure

$$\langle U \cup W, R \restriction (U \cup W), U, W \rangle,$$

we get an isomorphic copy $\langle \mathbf{A}, S', X, Y \rangle$ in REALTIME-SPACE(log). Due to condition iii) and the fact that $K > 2$, we see that $\mathbf{N} \setminus \mathbf{A}$ contains infinitely many odd and infinitely many even integers. Thus, S' can be extended to a total ordering S over $\mathbf{N}$ as follows:

a) Order $(\mathbf{N} \setminus \mathbf{A}) \cap (2\mathbf{N} + 1)$ via the usual $<$ ordering, and order $(\mathbf{N} \setminus \mathbf{A}) \cap 2\mathbf{N}$ via the reverse ordering $>$.

b) Let X, $(\mathbf{N} \setminus \mathbf{A}) \cap (2\mathbf{N} + 1)$, $(\mathbf{N} \setminus \mathbf{A}) \cap 2\mathbf{N}$, Y be successive intervals of S.

It is clear that S is in REALTIME-SPACE(log) and that $\langle \mathbf{N}, R \rangle$ and $\langle \mathbf{N}, S \rangle$ are isomorphic.

5.4. Recall that $\alpha.\beta$ denotes the order type of the sum of a family of copies of α indexed by the ordered set β (cf. 3.1).

THEOREM. *Let λ be any total order type and let $\langle \mathbf{N}, R \rangle$ be a recursive total ordering with order type $\alpha = \omega + \mathbf{Z}.\lambda + \omega^*$. There exists a* REALTIME-SPACE(log) *total ordering S over $\mathbf{N}$ and an isomorphism F between $\langle \mathbf{N}, R \rangle$ and $\langle \mathbf{N}, S \rangle$.*

PROOF. We consider four cases which exhaust all possibilities.

Case 1. *λ has a maximal element.* Let λ be of the form $\lambda = \mu + 1$. Since $\mathbf{Z}$ has order type $\omega^* + \omega$, we see that $\alpha = \omega + \mathbf{Z}.\mu + \omega^* + \omega + \omega^*$. Thus, there is an element a in $\mathbf{N}$ (namely, the integer in $\langle \mathbf{N}, R \rangle$ which corresponds to the first element in the last ω part of α) such that the sets $U = \{x \in \mathbf{N}: R(x, a)\}$ and $V = \{x \in \mathbf{N}: R(a, x) \text{ or } x = a\}$ have respective order types $\omega + \mathbf{Z}.\mu + \omega^*$ and $\omega + \omega^*$.

Since U and V are recursively defined from R, they are recursive. Setting $W = \varnothing$ and applying Theorem 5.3, we get the desired isomorphic structure $\langle \mathbf{N}, S \rangle$ lying in REALTIME-SPACE(log).

Case 2. *λ has a minimal element.* This case is similar to the previous one.

Case 3. *λ has two successive elements.* Let λ be of the form $\lambda = \mu + 2 + \nu$. Then $\alpha = \omega + \mathbf{Z}.\mu + \omega^* + \omega + \omega^* + \omega + \mathbf{Z}.\nu + \omega^*$ Thus, there are elements a and b in $\mathbf{N}$ such that the sets $U = \{x \in \mathbf{N}: R(x, a)\}$, $V = \{x \in \mathbf{N}: [R(a, x) \text{ and } R(x, b)] \text{ or } x = a \text{ or } x = b\}$ and $W = \{x \in \mathbf{N}: R(b, x)\}$ have respective order types $\omega + \mathbf{Z}.\mu + \omega^*$, $\omega + \omega^*$ and $\omega + \mathbf{Z}.\nu + \omega^*$. Since U, V and W are recursively defined from R, they are recursive subsets of $\mathbf{N}$. Applying Theorem 5.3, we get the desired isomorphic structure lying in REALTIME-SPACE(log).

Case 4. *λ is dense without endpoints.* Then λ is the order type of the set of rational numbers, and we conclude via Proposition 3.1.

§6. Recursive linear orderings with good recursive sequences.

6.1. We now apply Theorem 4.1 to a type of orderings satisfying a condition which appears to cover the cases not relevant to Theorem 5.4.

THEOREM. *Let R be a recursive total ordering over $\mathbf{N}$ such that there exists a recursive sequence which is*

either *strictly R-increasing and cofinal in $\langle \mathbf{N}, R \rangle$,*

or *strictly R-increasing and has a supremum in $\langle \mathbf{N}, R \rangle$,*

or *strictly R-decreasing and coinitial in $\langle \mathbf{N}, R \rangle$,*

or *strictly R-decreasing and has an infimum in $\langle \mathbf{N}, R \rangle$.*

Then $\langle \mathbf{N}, R\rangle$ *is recursively isomorphic to some structure* $\langle \mathbf{N}, S\rangle$ *in* REALTIME-SPACE(log).

The proof of this theorem runs through subsections 6.2 to 6.15 below.

We treat only the case of an R-increasing recursive sequence, the R-decreasing case being similar.

6.2. Let ξ be an R-increasing recursive sequence which either is cofinal in $\langle \mathbf{N}, R\rangle$ or has a supremum m in $\langle \mathbf{N}, R\rangle$. We can suppose that ξ is also strictly increasing (hence cofinal) with respect to $<$ (if not, just replace $\xi(p)$ by $\mathrm{Sup}\{\xi(i): i \leq p\}$). For such a ξ, strictly increasing with respect to both $<$ and R, the restrictions of R and $<$ to the range $\mathbf{X}$ of ξ coincide.

Let $\mathbf{X}$ be the range of ξ. Since ξ is $<$-increasing and recursive, its range is an infinite recursive subset of $\mathbf{N}$.

We define a subset D of $\mathbf{N}$ as follows: $D = \{x \in \mathbf{N}\backslash\mathbf{X}: R(x, m)\}$ if m is the R-supremum of the sequence ξ, and $D = \mathbf{N}\backslash\mathbf{X}$ if ξ is cofinal in $\langle \mathbf{N}, R\rangle$. Being recursively defined from R and $\mathbf{X}$, the set D is also recursive.

By its very definition, D is the set of elements in $\mathbf{N}\backslash\mathbf{X}$ which are majorized by some element in the range $\mathbf{X}$ of ξ. Thus, it is possible to define unary functions φ and ψ with domain D and values in $\mathbf{N}$ as follows: $\psi(d) =$ the ($<$ or R) smallest element y of $\mathbf{X}$ such that $R(d, y)$, and $\varphi(d) =$ the $<$-smallest integer p such that $R(d, \xi(p))$.

It is clear that a) φ and ψ are recursive, b) $\psi = \xi \circ \varphi$, i.e. if $d \in D$ then $\psi(d) =$ the $(\varphi(d) + 1)$th element of $\mathbf{X}$, and c) if $d \in D$ and $x \in \mathbf{X}$ then

$$R(d, x) \quad \text{if and only if} \quad \psi(d) \leq x,$$
$$R(x, d) \quad \text{if and only if} \quad \psi(d) > x.$$

6.3. From the two substructures $\langle \mathbf{N}\backslash\mathbf{X}, R \restriction \mathbf{N}\backslash\mathbf{X}\rangle$ and $\langle \mathbf{X}, R \restriction \mathbf{X}\rangle$ (this last one being the same as $\langle \mathbf{X}, < \restriction \mathbf{X}\rangle$) one can recover the original structure $\langle \mathbf{N}, R\rangle$ via φ, using the following partition of R into five subrelations:

$$(\blacklozenge) \qquad R = [R \restriction \mathbf{N}\backslash\mathbf{X}] \cup [< \restriction \mathbf{X}] \cup R_1 \cup R_2 \cup [\mathbf{X} \times [\mathbf{N}\backslash(\mathbf{X} \cup D)]],$$

where $R_1 = \{(d, x): d \in D$ and $x \in \mathbf{X}$ and the $(\varphi(d) + 1)$st element of $\mathbf{X}$ is at most $x\}$ and $R_2 = \{(x, d): d \in D$ and $x \in \mathbf{X}$ and the $(\varphi(d) + 1)$st element of $\mathbf{X}$ is strictly greater than $x\}$.

REMARK. In case ξ is R-cofinal then the set $\mathbf{N}\backslash(\mathbf{X} \cup D)$ is empty; hence the factor $\mathbf{X} \times [\mathbf{N}\backslash(\mathbf{X} \cup D)]$ in $(\blacklozenge)$ is empty.

6.4. To follow the strategy described in 5.1, it is convenient to consider the recursive $\mathbf{N}$-valued relational structure $\langle \mathbf{N}\backslash\mathbf{X}, R \restriction \mathbf{N}\backslash\mathbf{X}, \varphi\rangle$, where φ is considered as a unary $\mathbf{N}$-valued relation over the recursive subset D of $\mathbf{N}$.

We apply Theorem 4.1 to the $\mathbf{N}$-valued structure $\langle \mathbf{N}\backslash\mathbf{X}, R \restriction \mathbf{N}\backslash\mathbf{X}, \varphi\rangle$, the anticipation function $\alpha(n) = \mathrm{Sup}\{0, [(\log_2(n/9))/2]\}$, and the constant $K = 2^{36}$. We get a structure $\langle A, T, \theta\rangle$ and a recursive and increasing isomorphism F between $\langle \mathbf{N}\backslash\mathbf{X}, R \restriction \mathbf{N}\backslash\mathbf{X}, \varphi\rangle$ and $\langle \mathbf{A}, T, \theta\rangle$ such that:

a) $\mathbf{A}$ and T are in REALTIME-SPACE(log),

b) θ is in REALTIME-SPACE(log)-ANTICIP(α), and

c) any two points of $\mathbf{A}$ are distant at least 3, and the first element of $\mathbf{A}$ is greater than 2^{36}.

6.5. We extend the bijection F between $\mathbf{N}\backslash\mathbf{X}$ and $\mathbf{A}$ to a recursive bijection G over $\mathbf{N}$ as follows:

$$G(p) = F(p) \quad \text{if } p \in \mathbf{N}\backslash\mathbf{X},$$

$$G(\text{the } (n+1)\text{th element of } \mathbf{X}) = \text{the } (n+1)\text{th element of } \mathbf{N}\backslash\mathbf{A}.$$

Let $S = G(R)$; this G defines an isomorphism between the structures $\langle \mathbf{N}, R\rangle$ and $\langle \mathbf{N}, S\rangle$. From equality (◆) (cf. 6.3) we see that

$$(◆◆) \quad S = G(R \restriction \mathbf{N}\backslash\mathbf{X}) \cup G(< \restriction \mathbf{X}) \cup G(R_1) \cup G(R_2) \cup G[\mathbf{X} \times [\mathbf{N}\backslash(\mathbf{X} \cup D)]].$$

In order to prove the theorem, we have to prove that S is in REALTIME-SPACE(log). We shall show that each of the above five components of S is in REALTIME-SPACE(log).

6.6. Since F is an isomorphism between $\langle \mathbf{N}\backslash\mathbf{X}, R \restriction \mathbf{N}\backslash\mathbf{X}, \varphi\rangle$ and $\langle \mathbf{A}, T, \theta\rangle$, it is clear that $G(R \restriction \mathbf{N}\backslash\mathbf{X}) = T$. Also, $G(< \restriction \mathbf{X}) = < \restriction \mathbf{N}\backslash\mathbf{A}$ by the very definition of G. Since T and $\mathbf{A}$ are in REALTIME-SPACE(log), so are $G(R \restriction \mathbf{N}\backslash\mathbf{X})$ and $G(< \restriction \mathbf{X})$.

6.7. From the definition of D we get the following definition of $G(D)$: $G(D) = F(D) = \mathbf{A}$ if ξ is cofinal in $(\mathbf{N}, R)$, and $G(D) = F(D) = \{a \in \mathbf{A}: T(a, F(m))\}$ if m is the supremum of the sequence ξ. In particular, the set $G(D)$ is in REALTIME-SPACE(log).

Since $G[\mathbf{X} \times [\mathbf{N}\backslash(\mathbf{X} \cup D)]] = G(\mathbf{X}) \times G[\mathbf{N}\backslash(\mathbf{X} \cup D)]] = \mathbf{N}\backslash\mathbf{A} \times \mathbf{A}\backslash G(D)$ we see that $G[\mathbf{X} \times [\mathbf{N}\backslash(\mathbf{X} \cup D)]] = \varnothing$ if ξ is cofinal in $\langle \mathbf{N}, R\rangle$, and

$$G[\mathbf{X} \times [\mathbf{N}\backslash(\mathbf{X} \cup D)]] = \mathbf{N}\backslash\mathbf{A} \times \{a \in \mathbf{A}: T(F(m), a) \text{ or } a = F(m)\}$$

if m is the supremum of the sequence ξ. Since T and $\mathbf{A}$ are in REALTIME-SPACE(log), so is the set $\{a \in \mathbf{A}: T(F(m), a) \text{ or } a = F(m)\}$. Thus, the relation $G[\mathbf{X} \times [\mathbf{N}\backslash(\mathbf{X} \cup D)]]$ is in REALTIME-SPACE(log).

6.8. Let $T_1 = G(R_1)$ and $T_2 = G(R_2)$. The definition of R_1 shows that

$$T_1 = \{(G(d), G(x)): d \in D \text{ and } x \in \mathbf{X} \text{ and the } (\varphi(d)+1)\text{th element of } \mathbf{X} \text{ is at most } x\}.$$

Since F is an isomorphism between $(\mathbf{N}\backslash\mathbf{X}, \varphi)$ and $(\mathbf{A}, \theta)$, we have $\varphi = \theta \circ G \restriction D$; hence

$$T_1 = \{(a, y) \in G(D) \times \mathbf{N}\backslash\mathbf{A}: \text{the } (\theta(a)+1)\text{th element of } \mathbf{X} \text{ is at most } G^{-1}(y)\}.$$

Since $G \restriction \mathbf{X}$ exchanges the structures $\langle \mathbf{X}, < \restriction \mathbf{X}\rangle$ and $\langle \mathbf{N}\backslash\mathbf{A}, < \restriction \mathbf{N}\backslash\mathbf{A}\rangle$, we see that

$$(*) \quad T_1 = \{(a, y) \in G(D) \times \mathbf{N}\backslash\mathbf{A}: \text{the } (\theta(a)+1)\text{th element of } \mathbf{N}\backslash\mathbf{A} \text{ is at most } y\}.$$

Similarly, we have

$$(**) \quad T_2 = \{(y, a) \in \mathbf{N}\backslash\mathbf{A} \times G(D): \text{the } (\theta(a)+1)\text{th element of } \mathbf{N}\backslash\mathbf{A} \text{ is strictly greater than } y\}.$$

6.9. Using equality (*) of 6.8, we now show that T_1 is in REALTIME-SPACE(log).

Since $G(D)$ and $\mathbf{N}\backslash\mathbf{A}$ are in REALTIME-SPACE(log) and θ is in REALTIME-SPACE(log)-ANTICIP(α), there exist *realtime* Turing machines U, V and W, working in *log-space*, such that: U recognizes $G(D)$, V recognizes $\mathbf{N}\backslash\mathbf{A}$, and W computes θ with *anticipation* of the output at time α. Machines U, V and W have one input tape and p worktapes; moreover, W has one output tape.

Equality (*) suggests an obvious algorithm to recognize T_1:

i*) Given inputs u and v, check if they represent (in binary) a pair $(a, y) \in G(D) \times \mathbf{N}\backslash\mathbf{A}$.

ii*)a) Compute $\theta(a)$.

ii*)b) Check if $0, 1, 2, 3, \ldots$ are in $\mathbf{N}\backslash\mathbf{A}$, up to the obtention of $\theta(a) + 1$ elements of $\mathbf{N}\backslash\mathbf{A}$.

ii*)c) Compare y with the $(\theta(a) + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$.

All parts of this algorithm can be performed by suitable simulations of U, V and W.

Parts i*) and ii*)a) can be started simultaneously. As soon as an output η is obtained in ii*)a) then, while still performing i*), points ii*)b) and ii*)c) can be performed with η in place of $\theta(a)$. If $a \in G(D)$ then η is $\theta(a)$. If $a \notin G(D)$ then $\theta(a)$ is not defined, and when i^*) is completed then M will know about it and will just forget all about the computation done with η.

The main problem to get the REALTIME-SPACE(log) character is whether ii*)b), c) can be completed while performing i*). It happens that a good enough anticipation function α gives the solution.

We describe below a realtime Turing machine M which performs i*) and deals with ii*)b), c) up to the completion of i*). We shall then prove that M has enough time to complete ii*)b), c).

Machine M has two input tapes and $4p + 4$ worktapes, and acts as follows on inputs u, v.

i) a) M checks if u and v are in $\{\text{Bin}(x): x \in \mathbf{N}\}$. If so, let a and y be such that $u = \text{Bin}(a)$, $v = \text{Bin}(y)$. Then,

b) M checks if a is in $G(D)$ via a simulation of U on input u, using p worktapes.

c) M checks if y is in $\mathbf{N}\backslash\mathbf{A}$ via a simulation of V on input v, using p more worktapes.

i bis) In order to be able, later on, to compare y to some integer (point ii)c)), machine M makes a copy on some other worktape of input v (which is $\text{Bin}(y)$) as it is being read on input tape 2.

ii)a) In parallel with i), machine M (using p more worktapes) simulates W on input u up to the obtention of an anticipated output (i.e. up to the simulation of a prefinal state of W). The realtime character of U and W allows the input tape of M containing u to be used for both simulations of W and of U (point i)b), above).

ii)b) When (and if) an anticipated output is obtained—i.e. when a prefinal state of W is simulated—which is of the form $\text{Bin}(\eta)$, where $\eta \in \mathbf{N}$, then M (considers η as the potential value of $\theta(a)$ and) starts computing the $(\eta + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$ via the following iterative process:

1) On two devoted worktapes of M are written the binary representations of some integers j and k whose values are initially set up to 0: j is to vary from 0 up to the $(\eta + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$, while k counts the number of elements of $\mathbf{N}\backslash\mathbf{A}$ strictly less than the current value of j.

2) M checks if j is in $\mathbf{N}\backslash\mathbf{A}$ via a simulation of V on input $\text{Bin}(j)$ (using p more worktapes).

3) When this simulation is completed then

- either this simulation ends in a rejecting state of V (i.e. j belongs to $\mathbf{A}$); then M leaves unchanged the value of k, increases the value of j by one unit and goes back to point 2),
- or this simulation ends in an accepting state of V (i.e. j belongs to $\mathbf{N}\backslash\mathbf{A}$); then machine M compares the value of k with that of η. If $k = \eta$ then j is the $(\eta + 1)$th

element of $\mathbf{N}\backslash\mathbf{A}$ and M stops this iterative process, else j is less than the $(\eta + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$ and M increases by one unit both values of j and k and goes back to point 2).

ii)c) When the $(\eta + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$ has been found then it is the current value of j.

• *Case* 1. Input v, i.e. $\mathrm{Bin}(y)$, on input tape 2 is so long that it has not yet been completely read. Then the length of v is strictly greater than that of $\mathrm{Bin}(j)$; hence the $(\eta + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$ is strictly less than y. Thus, provided the simulations i)b) and i)c) (which have to be continued up to time step $\mathrm{Sup}\{|u|, |v|\}$) lead to acceptance, i.e. provided (a, y) is in $G(D) \times \mathbf{N}\backslash\mathbf{A}$, in which case the anticipated output η is really $\theta(a)$, we know that (a, y) is in R_1.

• *Case* 2. Input v on input tape 2 has already been completely read, and duplicated on some worktape (cf. iv). Then machine M compares, relative to the usual $<$-ordering, the value of y with that of the $(\eta + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$.

iii) Machine M stops in an accepting state if and only if the following conditions hold:

1) All checks described in i) are positive (i.e. (a, y) is in $G(D) \times \mathbf{N}\backslash\mathbf{A}$).

2) There is enough time to complete the computations described in ii)b) and ii)c).

3) The $(\eta + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$ is less than or equal to y, i.e. either Case 1 of ii)c) occurs or the comparison done in Case 2 leads to this inequality.

6.10. Let T_1^0 be the relation recognized by machine M. It is clear from the description of M that T_1^0 is included in T_1.

To prove that every pair (a, y) in T_1 is in T_1^0, we have to show that if machine M starts with the binary representations u and v of a and y as inputs then conditions 1), 2) and 3) in iii) hold. The only nontrivial one is 2). Since ii)a) is started without delay and ii)b) and ii)c) are performed as soon as ii)a) is completed, we see that condition 2) is implied by the following fact:

Fact. For every element a in $G(D)$ the total time necessary to perform the computation described in ii)a), ii)b) *and* ii)c) *above is less than $|u|$ (hence, is less than the total duration.* $\mathrm{Sup}(|u|, |v|)$ *of the computation of M).*

Subsections 6.11 to 6.14 are devoted to a proof of this fact. All along, we shall suppose that $u = \mathrm{Bin}(a)$ and $v = \mathrm{Bin}(y)$ and $(a, y) \in T_1$.

6.11. First, we note that only $\alpha(|u|)$ computation steps are necessary to perform ii)a) (and thus obtain the anticipated output in the simulation of V on input u). Let N be the $(\theta(a) + 1)$th element of $\mathbf{N}\backslash\mathbf{A}$. The computation time to perform ii)c) is zero in Case 1. In Case 2, ii)c) asks to compare y and N, hence requires at most $|\mathrm{Bin}(N)|$ steps.

6.12. We now get an upper bound of the time required to perform ii)b). The iterative process in ii)b) is performed for values of j less than $\mathbf{N}$. For each such j the computation of M can be analysed as follows:

First, $|\mathrm{Bin}(j)|$ steps are used to simulate the computation of V on input $\mathrm{Bin}(j)$.

After completing this simulation, the head of the tape containing $\mathrm{Bin}(j)$ scans the first cell to the right of $\mathrm{Bin}(j)$. Thus, only $|\mathrm{Bin}(j)|$ more steps are necessary to replace j by $j + 1$ while pulling back the head on the first cell.

Since $k \leq j$, at most $2|\mathrm{Bin}(j)| + 1$ steps are necessary to replace k by $k + 1$ (if necessary) and pull back the head.

At most $2|\mathrm{Bin}(j)| + 1$ steps are necessary to compare k with $\theta(a)$ and pull back the heads.

Since $1 \leq |\mathrm{Bin}(j)|$, we see that the passage from j to $j+1$ takes at most $8|\mathrm{Bin}(j)|$ steps. Thus, the total computation time induced by the iterative process of point ii)b) is at most

$$8[|\mathrm{Bin}(0)| + |\mathrm{Bin}(1)| + \cdots + |\mathrm{Bin}(N)|],$$

hence at most $8(N+1)|\mathrm{Bin}(N)|$.

6.13. We know that **A** is sparse: any two distinct points of **A** are distant of at least 3 and the first element of **A** is positive (cf. 6.4). Thus, the $(n+1)$th element of $\mathbf{N}\backslash\mathbf{A}$ is less than or equal to $2n$. In particular $\mathbf{N} \leq 2\theta(a)$. Since W anticipates its output $\theta(a)$ at time α, we have $|\mathrm{Bin}(\theta(a))| \leq \alpha(|u|)$. Hence,

$$\alpha(|u|) = \mathrm{Sup}\{0, [\log_2(|u|/9)/2] - 1\} = [\log_2(|u|/9)/2] - 1$$

and

$$\tau \leq 9 \times 4^{\alpha(|u|)+1} \leq 9 \times 4^{\log_2(|u|/9)/2} = |u|.$$

6.14. Thus, the total time τ necessary to perform ii)a), ii)b) and ii)c) can be bounded as follows:

$$\begin{aligned}\tau &\leq \alpha(|u|) + |\mathrm{Bin}(\mathbf{N})| + 8(\mathbf{N}+1)|\mathrm{Bin}(\mathbf{N})| \\ &\leq \alpha(|u|) + [\alpha(|u|) + 1] + 8 \times 2^{\alpha(|u|)+1} \times [\alpha(|u|) + 1] \\ &\leq [\alpha(|u|) + 1][2 + 8 \times 2^{\alpha(|u|)+1}] \\ &\leq [2^{\alpha(|u|)+1}][9 \times 2^{\alpha(|u|)+1}] = 9 \times 4^{\alpha(|u|)+1}.\end{aligned}$$

Since the input u is the binary expansion of an element of $G(D)$ and the first element of **A** (hence of $G(D)$) has been taken greater than 2^{36}, we see that $|u| \geq 36$. Thus,

$$\alpha(|u|) = \mathrm{Sup}\{0, \llcorner\log_2(|u|/9)/2\lrcorner - 1\} = \llcorner\log_2(|u|/9)/2\lrcorner - 1$$

and

$$\tau \leq 9 \times 4^{\alpha(|u|)+1} \leq 9 \times 4^{\log_2(|u|/9)/2} = |u|.$$

This proves Fact 6.10 and hence that T_1 is recognized by machine M.

6.15. We observe that M uses only log-space on each of its worktapes:

- This is obvious for those tapes performing simulations of worktapes or output tapes of machines U, V or W, since these machines work within log-space.
- As for the tapes containing the variables j and k, we know from 6.13 that $k \leq j \leq 2\eta$, whence $|\mathrm{Bin}(k)| \leq |\mathrm{Bin}(j)| \leq |\mathrm{Bin}(\eta)| + 1 \leq \log(|u|) + 1$ since η is produced by W.

Since M is realtime, this shows that T_1 is in REALTIME-SPACE(log).

The proof that T_2 is in REALTIME-SPACE(log) is quite similar. Using (**) (cf 6.8), we see that only condition ii)c) has to be modified in an obvious way in 6.9.

Along with (◆◆) (cf. 6.5), 6.6 and 6.7, this proves that S is in REALTIME-SPACE(log).

This concludes the proof of Theorem 6.1.

§7. The main theorem.

THEOREM. 1°) *Let* $a \in \mathbf{N}$, $a \geq 2$. *Every recursive total ordering* $\langle \mathbf{N}, R \rangle$ *has an isomorphic copy* $\langle \mathbf{N}, S \rangle$ *for which* S *is in* REALTIME-SPACE(log) *relative to the* a*-ary representation of integers.*

2°) *Moreover, if the order type of* R *is not of the form* $\omega + \mathbf{Z}.\ \alpha + \omega^*$ *for some* α, *then one can suppose the isomorphism to be recursive.*

PROOF. We observe the two following easy facts:

1°) Every total ordering R over $\mathbf{N}$ satisfies (at least) one of the following conditions:

1) R *has order type* $\omega + \mathbf{Z}.\ \alpha + \omega^*$ *for some* α.

2) *There is no* R*-greatest element.*

3) *There is no* R*-least element.*

4) *There exists* c *such that* $\{x\colon R(x, c)\}$ *is nonempty and has no* R*-greatest element.*

5) *There exists* c *such that* $\{x\colon R(c, x)\}$ *is nonempty and has no* R*-least element.*

2°) If R is recursive, then, for $i = 2$ to 5, condition i) above implies the existence of a recursive sequence ξ satisfying condition i^*) below:

2*) ξ *is strictly* R*-increasing and cofinal in* $\langle \mathbf{N}, R \rangle$.

3*) ξ *is strictly* R*-decreasing and coinitial in* $\langle \mathbf{N}, R \rangle$.

4*) ξ *is strictly* R*-increasing and has a supremum is* $\langle \mathbf{N}, R \rangle$.

5*) ξ *is strictly* R*-decreasing and has an infimum in* $\langle \mathbf{N}, R \rangle$.

For instance, in case condition 2) holds, we can recursively define ξ as follows: $\xi(0) = 0$, and $\xi(n + 1) =$ the $<$-least element y such that $R(\xi(n), y)$ and $R(i, y)$ hold for all $i \leq n$.

3°) If condition 1) holds then we conclude via Theorem 5.4, else we conclude via Theorem 6.1.

REFERENCES

[1] DOUGLAS CENZER and JEFFREY REMMEL, *Polynomial-time complexity of models* (to appear).

[2] ALAN COBHAM, *On the base-dependence of sets of numbers recognizable by finite automata*, ***Mathematical Systems Theory***, vol. 3 (1969), 186–192.

[3] PATRICK DEHORNOY, *Turing complexity of the ordinals*, ***Information Processing Letters***, vol. 23 (1986), pp. 167–170.

[4] STEPHEN C. KLEENE, *On the forms of the predicates in the theory of constructive ordinals.* II, ***American Journal of Mathematics***, vol. 77 (1955), pp. 405–428.

[5] MICHAEL MOSES, *Recursive linear orders with recursive successivities*, ***Annals of Pure and Applied Logic***, vol. 27 (1984), pp. 253–264.

[6] JEFFREY REMMEL, *Recursive isomorphism types of recursive Boolean algebras*, this JOURNAL, vol. 46 (1981), pp. 572–594.

[7] ———, *Recursively categorical linear orderings*, ***Proceedings of the American Mathematical Society***, vol. 83 (1981), pp. 387–391.

[8] JOSEPH G. ROSENSTEIN, ***Linear orderings***, Academic Press, New York, 1982.

[9] R. E. STEARNS, J. HARTMANIS and P. M. LEWIS, *Hierarchies of memory limited computations*, ***Proceedings of the IEEE conference on switching, circuit theory and logical design***, IEEE, 1965, pp. 179–190.

[10] RICHARD WATNICK, *A generalization of Tennenbaum's theorem on effectively finite recursive linear orderings*, this JOURNAL, vol. 49 (1984), pp. 563–569

[11] KLAUS WAGNER and GERD WECHSUNG, ***Computational complexity***, Reidel, Dordrecht, 1986.

LABORATOIRE D'INFORMATIQUE THÉORIQUE ET PROGRAMMATION
UNIVERSITÉ PARIS-VII
75251 PARIS, FRANCE