PUBLICATIONS DE L'INSTITUT MATHÉMATIQUE
Nouvelle série, tome 1 (15), p. 31—65.

BASIC THEOREMS ON TURING ALGORITHMS

Vladeta Vučković

(Received 25. XII 1961)

1. Introduction. In [1] we exposed briefly an approach to the theory of algorithms by the use of *Turing* machines. We left most theorems there withuot proof and we exhibited there only one example of concrete *Turing* algorithm. In this paper we shall give complete proofs of all theorems which were only mentioned in [1] and we shall exhibit effectively some more algorithms for computation of particular word-functions. In the same time we shall expose the principles of the theory of *Turing* algorithms with all needed details; so, omst parts of this paper can be read without previous knowledge of [1].

2. Background. The origin of this paper and the paper [1] was a study of the excellent monograph [2] of *Martin Davis*. The set-up of *M. Davis* suggested the idea to apply his version of *Turing* machines not only for the computation of arithmetical functions but of word-functions in general. The only problem was the question of arithmetization, and for this sake our theory of recursive word-functions, as exposed in [3], seemed the best suited. In [1] we exhibited this arithmetization with sufficient details; therefore we shall not reproduce it here anew.

Our aim here is twofold: to exhibit effectively all necessary algorithms and to do this with a minimum of changes in the original treatment of *M. Davis*. Naturally, we could not adopt any of concrete machines of [2]; but in some proofs we had only to change the machines in question. This was impossible only in proof that all $\mathfrak{A}$-primitive rceursive word-functions are $\mathfrak{A}$-algorithmic (i. e. $\mathfrak{A}$-computable). Also, the equivalence with *Markov*'s normal algorithms has not its countepart in [2].

As to the originality of our exposition we mention, that, after all, the idea to apply *Turing* machines for the computation of word-functions goes back to *Turing* himself. Many other versions are known (f. i. [6], [4]), but no one was effectively developed in all details. We believe this to be a reason why the theory of algorithms is developed only in terms of normal algorithms, and we believed to do justice to *Turing*'s original achievement by developing it in all details. So we believe to have given ground enough for further achievements in this direction.

3. Relations to normal algorithms. The theory of *Turing* algorithms, as developed here, is not meant to be a concurrence to the theory of *Markov*'s normal algorithms. We elaborate only an approach which was neglected, as we believe, because of some disadvantages in the first beginnings. Namely, some of the first *Turing* algorithms are very long, and generally every *Turing* algorithm is longer than the corresponding normal algorithm. But once this first tedious work being done, the further development presents no greater difficulties and the normal form theorem, enumeration theorem etc, are obtained in a straightforward manner which has not its counterpart in the theory of normal algorithms.

After all, *Turing* algorithms have the advantage to be much more in the spirit of the contemporary technical trends than any other algorithms, these last being all more graphical than mechanical.

4. First definitions. We employ a fixed alphabet

$$(4.1) \qquad \mathfrak{S} = \{S_0, S_1, S_2, \ldots S_{n-1}\}\ ,\ n > 1,$$

and we study the word functions with arguments and values in the set $\Omega(\mathfrak{S})$, where

$$(4.2) \qquad \Omega(\mathfrak{S}) = \text{the set of all words written with letters of } \mathfrak{S}.$$

By convention, we shall sometimes denote S_0 by 1. (If $n=1$ $\Omega(\mathfrak{S})$ will be reduced to the set of all numerals, and we shall have the classical case of computable arithmetical functions).

The mos. of definitions here are taken directly from *Davis* [2], with some minor changes. Therefore we list them without many comments.

Definition 4.1. *An expression is a finite sequence (possibly empty) of symbols chosen from the list:*

$$q_1, q_2, q_3, \ldots ;$$
$$S_0, S_1, \ldots ,S_{n-1};$$
$$S_n, S_{n+1} \ldots ;$$
$$R, L.$$

We call $\{S_0, S_1, \ldots , S_{n-1}\}$ the *printing alphabet* and $\{S_n, S_{n+1}, \ldots ,\}$ the *auxiliary alphabet*. In this one the letter S_n will be in most cases denoted by O, representing an open square, or a blank. S_{n+1} will be often denoted by $*$. This symbol will serve as auxiliary to write m-tuples of words.

Definition 4.2. *A quadruple is one expression having one of the following forms:*

(1) $q_i S_j S_k q_l$

(2) $q_i S_j R q_l$

(3) $q_i S_j L q_l$

(4) $q_i S_j q_k q_l$

Definition 4.3. *A Turing algorithm over $\mathfrak{S}$ (or a Turing machine over $\mathfrak{S}$) is a finite, nonempty set of quadruples that contains no two quadruples whose first two symbols are the same.*

Algorithms are denoted by La.in or German capitals: Z, U, $\mathfrak{X}$, ...

The q_i's and S_i's occuring in the quadruples of a *Turing* algorithm Z are called the *internal configurations* and the *alphabet of* Z respectively. In the alphabet we d:stinguish the *printing alphabet* $\{S_0, S_1, \ldots , S_{n-1}\} = \mathfrak{S}$ and the *auxiliary alphabet* which consists of all S_i's that are not in $\mathfrak{S}$ but occur in the alphabet of Z.

If none of the quadruples of Z is of the type (4), we call *Z a simple Turing algorithm.*

Definition 4. 4. *An instantaneous description is an expression containing exactly one* q_i, *neither R or L, and is such that* q_i *is not the rightmost symbol.*

If Z is a *Turing* algorithm and α is an instantaneous description, then α is an *instantaneous description of* Z if the q_i occuring in α is an internal configuration of Z and if the $S'_i s$ that occur in α are part of the alphabet of Z.

An expression that consists entirely of the letters S_i is called a *tape expression.*

Definition 4. 5. *Let Z be a Turing algorithm and* α *one instantaneous description of Z. If* q_i *is the internal configuration occuring in* α *and* S_j *is the symbol immediately to the right of* q_i, *then we call* q_i *the internal configuration of Z at* α, *and we call* S_j *the symbol scanned by Z at* α. *Removing* q_i *from* α *we get the expression on the tape of Z at* α.

Definition 4. 6. *Let Z be a Turing algorithm, and let* α, β *be instantaneous descriptions. We write* $Z:\alpha \vdash \beta$, *or (when no ambiguity can result)* $\alpha \vdash \beta$ *to mean that one of the following alternatives holds:*

(1) *There exist expressions P and Q (possibly empty) such that* α *is* $Pq_i S_j Q$, β *is* $Pq_l S_k Q$ *and Z contains* $q_i S_j S_k q_l$;

(2) *There exist expressions P and Q (possibly empty) such that* α *is* $Pq_i S_j S_k Q$, β *is* $PS_j q_l S_k Q$, *where Z contains* $q_i S_j R q_l$;

(3) *There exists an expression P (possibly empty) such that* α *is* $Pq_i S_j$, β *is* $PS_j q_l O$, *where Z contains* $q_i S_j R q_l$;

(4) *There exist expressions P and Q (possibly empty) such that* α *is* $PS_k q_i S_j Q$, β *is* $Pq_l S_k S_j Q$, *where Z contains* $q_i S_j L q_l$;

(5) *There exists an expression Q (possibly empty) such that* α *is* $q_i S_j Q$, β *is* $q_l O S_j Q$ *where z contains* $q_i S_j L q_l$.

Comparing with the definiton 1. 7 of Ch. I of [2] we see that *Turing* algoritms „work" exactly in the same manner as Turing machines of *M. Davis.*

We mention the trivial

Theorem 4. 1. *If* $Z:\alpha \vdash \beta$ *and* $Z:\alpha \vdash \gamma$ *then* $\beta=\gamma$.

If $Z:\alpha \vdash \beta$ *and* $Z \subset Z'$ *then* $Z':\alpha \vdash \beta$.

Definition 4. 7. *An instantaneous description* α *iz called terminal with respect to Z if for no* β *we have* $Z:\alpha \vdash \beta$.

Definition 4. 8. *By a computation of a Turing algorithm Z is meant a finite sequence* $\alpha_1, \alpha_2, \ldots, \alpha_p$ *of instantaneous descriptions such that* $Z:\alpha_i \vdash \alpha_{i+1}$ *for* $1 \leqslant i < p$ *and such that* α_p *is terminal with respect to Z. In such a case we write*

$$Z:\alpha_1 \vdash \alpha_2 \vdash \cdots \vdash \alpha_{p-1} \vdash \cdot \alpha_p,$$

or $Z:\alpha_1 \models \alpha_{p-1} \vdash \cdot \alpha_p$, *or* $Z:\alpha_1 \models \cdot \alpha_p$, *or* $\alpha_p = \mathrm{Res}_Z(\alpha_1)$ *and we call* α_1 *the resultant of* α_1 *with respect to Z.*

By convention, the internal configuration at α_1 will be taken as q_1.

$Z:\alpha \models \beta$ denotes that there is a sequence $\alpha_1, \alpha_2, \ldots, \alpha_k$ of instantaneous descriptions, such that $Z:\alpha_i \vdash \alpha_{i+1}$ for $1 \leqslant i < k$ and $\alpha=\alpha_1$ and $\beta \vdash \alpha_k$, but β is not terminal with respect to Z.

5. Algorithmic and partially algorithmic word-functions. To adopt the notion of *Turing* algorithm for the computation of word-functions we need to change slightly the corresponding definitions of *Davis.*

Definition 5. 1. *Let* $A_1, A_2, \ldots, A_m$ *be words* (of $\Omega(\mathfrak{S})$, as always in the sequel — if not explicitely declared otherwise). *With the m-tuple* $(A_1, A_2, \ldots, A_m)$ *we associate the tape expression* $\overline{(A_1, A_2, \ldots, A_m)}$, *where*

$$\overline{(A_1, A_2, \cdots, A_m)} = A_1 * A_2 * \cdots * A_m.$$

(Remember that $*$ is the letter S_{n+1} of the auxiliary alphabet).
If f. i. A_i is empty, then

$$\overline{(A_1, A_2, \ldots, A_i, \ldots A_m)} = A_1 * A_2 * \ldots * A_{i-1} * O * A_{i+1} * \cdots * A_m,$$

i. e. for the empty word we write the letter $O = S_n$ of the auxiliary alphabet.

We point that in [1] we defined $\overline{(A_1, \ldots, A_m)}$ to be $A_1 O A_2 O \cdots O A_m$. Here we adopt the above definition as more suitable. It is obvious that with it nothing is changed in principle in the exposition of the paper [1].

Definition 5. 1 serves for inputs. For outputs we give

Definition 5. 2. *Let* M *be any expression. Then* $< M >$ *is the word in* $\Omega(\mathfrak{S})$ *obtained from* M *so that all symbols in* M *which do not belong to the printing alphabet* $\mathfrak{S}$ *are deleted, and the remaining letters of* $\mathfrak{S}$ *are put together, in the same order as they appear in* M.

F. i. if $n \geqslant 8$,

$$\langle S_3 * O S_2 q_{11} S_5 S_1 S_7 \rangle = S_3 S_2 S_5 S_1 S_7$$

and

$$\langle q_5 S_{n+5} S_{n+2} * * O \rangle = 0,$$

i. e. if M does not contain any letter of $\mathfrak{S}$ then $\langle M \rangle$ is the empty word.

Definition 5. 3. *Let* Z *be a Turing algorithm over* $\mathfrak{S}$. *Then, for each* m, *we associate with* Z *an m-ary word-function* $\Psi_Z(X_1, X_2, \ldots, X_m)$, *with domain and range in* $\Omega(\mathfrak{S})$ *as follows:*

For each m-tuple $(A_1, A_2, \ldots, A_m)$ *we set* $\alpha_1 = q_1 \overline{(A_1, A_2, \ldots, A_m)}$ *and we distinguish between two cases:*

(1) *There exists a computation or* $Z, \alpha_1, \alpha_2, \ldots, \alpha_p$. *In this case we set*

$$\Psi_Z(A_1, A_2, \ldots, A_m) = \langle \alpha_p \rangle = \langle \mathrm{Res}_Z(\alpha_1) \rangle.$$

(2) *There exists no computation beginning with* α_1, *i. e.* $\mathrm{Res}_Z(\alpha_1)$ *is undefined. In this case we leave* $\Psi_Z(A_1, A_2, \ldots, A_m)$ *undefined.*

Definition 5. 4. *An m-ary word-function* $f(X_1, \ldots, X_m)$ *with domain and range in* $\Omega(\mathfrak{S})$ *is partially (Turing) algorithmic it there exists a Turing algorithm* Z *over* $\mathfrak{S}$ *such that*

$$f(X_1, \ldots, X_m) \simeq \Psi_Z(X_1, \ldots, X_m).$$

(Here $\simeq$ means: if the right side is defined so is the left and both are equal, and if the right side is not defined also the left one is not defined). *In this case we say that* Z *computes* f. *If, in addition,* f *is a total word-function, then it is called (Turing) algorithmic.*

We shall always use short expressions ,,partially algorithmic" and ,,algorithmic" for ,,partially *Turing* algorithmic" and ,,*Turing* algorithmic" respectively.

6. Some elementary algorithms. We exhibit here some algorithms which will be needed in the later parts.

Example 6. 1 *Addition of a fixed word.* We construct the *Turing* algorithm ${}_AZ$ such that $\Psi_{{}_AZ}(X) = X + A = AX$. (For the definition of $X + Y$ see [3], formula (4. 2)). Let the word A be $S_{i_1} S_{i_2}, \ldots, S_{i_k}$, where all S_{i_ν} are $\in \mathfrak{S}$ for $\nu = 1, 2, \ldots, k$. ${}_AZ$ will consist of the following quadruples:

$$q_1 O O q_2$$

$$q_1 S_\nu S_\nu q_2, \quad \nu = 0, 1, \ldots, n-1.$$

$$q_2 S_\nu L q_2, \quad \nu = 0, 1, \ldots, n-1.$$

$$\left.\begin{matrix} q_\nu O S_{i_{k-(\nu-2)}} q_{\nu+1} \\ q_{\nu+1} S_{i_{k-(\nu-2)}} L q_{\nu+1} \end{matrix}\right\} \nu = 2, 3, 4, \ldots, k+1 \qquad \text{(print } A\text{)}$$

$$q_{k+2} O R q_{k+3}.$$

Let first $\alpha_1 = q_1 O$. Then

$$\begin{aligned} {}_AZ: q_1 O &\vdash q_2 O \\ &\vdash q_3 S_{i_k} \\ &\vdash q_3 O S_{i_k} \\ &\vdash q_4 S_{i_{k-1}} S_{i_k} \\ &\models q_{k+1} O S_{i_2} S_{i_3} . \ldots . S_{i_k} \\ &\vdash q_{k+z} S_{i_1} S_{i_2} . \ldots . S_{i_k} \\ &\vdash q_{k+2} O S_{i_1} S_{i_2} . \ldots . S_{i_k} \\ &\vdash \cdot O q_{k+3} S_{i_1} S_{i_2} . \ldots . S_{i_k} = O q_{k+3} A. \end{aligned}$$

So

$$\operatorname{Res}_{{}_AZ}(q_1 O) = O q_{k+3} A$$

i. e.

$$\langle \operatorname{Res}_{{}_AZ}(q_1 O) \rangle = A = O + A.$$

Let now $\alpha_1 = q_1 S_{j_1} S_{j_2} . \ldots . S_{j_p}$, where

$$S_{j_1} S_{j_2} . \ldots . S_{j_p} \in \Omega(\mathfrak{S}) - O.$$

Then

$$\begin{aligned} {}_AZ: q_1 S_{j_1} S_{j_2} . \ldots . S_{j_p} &\vdash q_2 S_{j_1} S_{j_2} . \ldots . S_{j_p} \\ &\vdash q_2 O S_{j_1} S_{j_2} . \ldots . S_{j_p} \\ &\models \cdot O q_{k+3} S_{i_1} . \ldots . S_{ik} S_{j_1} . \ldots . S_{j_p} \end{aligned}$$

and

$$\langle \operatorname{Res}_{{}_AZ}(q_1 S_{j_1} S_{j_2} . \ldots . S_{j_p}) \rangle = S_{j_1} S_{j_2} . \ldots . S_{j_p} + A.$$

Example 6. 2. *Identity.* The function $f(X) = X$ can be regarded as the function $\varphi(X) = X + O$. So the algorithm ${}_OZ$:

$$q_1 O O q_2$$

$$q_1 S_\nu S_\nu q_2, \quad \nu = 0, 1, \ldots, n-1,$$

computes this function.

Example 6. 3. *Unrestricted addition.* Let A be the same word as in example 6. 1. Let ${}_{\infty A}z$ be the algorithm

$$ {}_AZ \cup \{q_{k+3}\, S_\nu\, S_\nu\, q_1 \mid \nu = 0,\ 1,\ \dots\ ,\ n-1,\ n\}. $$

(Take into account that $S_n = O$). Obviously the algorithm ${}_{\infty A}z$ repeats without end the work of the algorithm ${}_AZ$, i. e. it adds the word A without comming ever to a halt. So $\mathrm{Res}_{{}_{\infty A}z}(q_1 X)$ does not exist for any word $X \in \Omega(\mathfrak{S})$: the function $\Psi_{{}_{\infty A}z}(X)$ is never defined.

Example 6. 4. *Inversed addition of a fixed word.* The algorithm Z_A will compute the function $f(X) = A + X = XA$, i. e. it writes the word A at the end of every word. Let A be as in example 6. 1. Z_A consists of quadruples:

$$
\begin{array}{ll}
q_1\, S_\nu\, R\, q_1, & \nu = 0,\ 1,\ \dots\ ,\ n-1 \\
q_1\, O\, S_{i_1}\, q_2 & \\
q_\nu\, S_{i_{\nu-1}}\, R\, q_\nu, & \nu = 2,\ 3,\ \dots\ ,\ k \\
q_\nu\, O\, S_{i_\nu}\, q_{\nu+1}, & \nu = 2, 3,\ \dots\ ,\ k.
\end{array}
$$

Let $\alpha_1 = q_1\, S_{j_1} S_{j_2} \cdot\ \dots\ \cdot S_{je}$. Then

$$
\begin{aligned}
Z_A : q_1\, S_{j_1} S_{j_2} \cdot\ \dots\ \cdot S_{jc} &\vdash S_{j_1}\, q_1\, S_{j_2} \cdot\ \dots\ \cdot S_{je} \\
&\models S_{j_1} S_{j_2} \cdot\ \dots\ \cdot S_{je}\, q_1\, O \\
&\vdash S_{j_1} S_{j_2} \cdot\ \dots\ \cdot S_{je}\, q_2\, S_{i_1} \\
&\vdash S_{j_1} S_{j_2} \cdot\ \dots\ \cdot S_{je}\, S_{i_1}\, q_2\, O \\
&\vdash S_{j_1} S_{j_2} \cdot\ \dots\ \cdot S_{je}\, S_{i_1}\, q_3\, S_{i_2} \\
&\models \cdot S_{j_1} S_{jz} \cdot\ \dots\ \cdot S_{je}\, S_{i_1} S_{i_2} \cdot\ \dots\ \cdot S_{i_{k-1}}\, q_{k+1}\, S_{i_k}
\end{aligned}
$$

So $<\mathrm{Res}_{Z_A}(q_1 X)> = XA = A + X$.

Remark that by adding to Z_A the quadruples

$$
\begin{array}{ll}
q_{k+1}\, S_\nu\, L\, q_{k+1} & \nu = 0,1,\ \dots\ ,\ n-1 \\
q_{k+1}\, O\, R\, q_{k+2} &
\end{array}
$$

we had then an algorithm Z_A' such that

$$ \mathrm{Res}\, z_A'(q_1 X) = O q_{k+2}\, XA. $$

Example 6.5 *Anihilator.* The algorithm Z_{nih} computes the zero-function $Z(X) = O$, i. e. it transforms every word into the empty word. Z_{nih} *is*

$$
\begin{array}{ll}
q_1\, S_\nu\, O\, q_2, & \nu = 0,1,\ \dots\ ,\ n-1 \\
q_2\, O\, R\, q_1. &
\end{array}
$$

We have

$$
\begin{aligned}
Z_{nih} : q_1\, S_{j_1} \cdot\ \dots\ \cdot S_{jk} &\vdash q_2\, O\, S_{j_2} \cdot\ \dots\ \cdot S_{jk} \\
&\vdash O\, q_1\, S_{j_2} \cdot\ \dots\ \cdot S_{jk} \\
&\models O\, O\, O \cdot\ \dots\ \cdot O\, q_1\, O,
\end{aligned}
$$

and $<\mathrm{Res}_{Z_{nih}}(q_1 X)> = O$.

Example 6.6. *Constant.* The algorithm Z_{nih+A} computes the function $f(X)=A$, i. e. it transforms every word into the word A.

Let $A=S_{i_1}S_{i_2}. \ldots . S_{i_k}$. Then Z_{nih+A} is

$q_1 S_\nu O q_2$, $\nu=0, 1, \ldots, n-1$.

$q_2 O R q_1$ (erase the word on the tape)

$q_1 O S_{i_1} q_3$ (begin to print the word A)

$$\left.\begin{array}{l} q_\nu S_{i_{\nu-2}} R q_\nu \\ q_\nu O S_{i_{\nu-1}} q_{\nu+1} \end{array}\right\} \begin{array}{l} \nu=3,4, \ldots, k+1. \\ \text{(finish to print the word } A) \end{array}$$

Obviously $< \mathrm{Res}_{Z_{nih+A}}(q_1 X) > =A$ for every word $X \in \Omega(\mathfrak{S})$.

Example 6.7. *Predecessor.* Z is the algorithm which erases the first letter of every word (and transforms the empty word into itself):

$$q_1 S_\nu O q_2 \quad \nu=0,1, \ldots, n-1.$$

Example 6.8. *First coordinate.* We construct now the algorithm Z_1 which will give the first word X_1 of every m-tuple $(X_1, \ldots, X_m)$.

$q_1 O R Q_1$

$q_1 S_\nu R q_1$, $\nu=0,1, \ldots n-1$.

$q_1 * O q_2$

$$\left.\begin{array}{ll} q_i O R q_i & \\ q_i S_\mu O q_i, & \mu=0, 1, \ldots, n-1 \\ q_i * O q_{i+1} & \end{array}\right\} i=2, 3, \ldots, m-1$$

$q_m O R q_{m+1}$

$q_{m+1} S_\mu O q_{m+2}$ $\mu=0, 1, \ldots, n-1$

$q_{m+2} O R q_{m+1}$

Let $X_1=S_{i_1}S_{i_2}. \ldots . S_{i_k}$. We have

$$\begin{aligned} Z_1 : & q_1 X_1 * X_2 * \cdots * X_m \\ & \vdash S_{i_1} q_1 S_{i_2}. \ldots . S_{i_k} * X_2 * \cdots * X_m \\ & \models X_1 q_1 * X_2 * \cdots * X_m \\ & \vdash X_1 q_2 O X_2 * \cdots * X_m \\ & \models X_1 OOO. \ldots . O q_m O X_m \\ & \vdash X_1 OOO. \ldots . OO q_{m+1} X_m \\ & \models \cdot X_1 OOO. \ldots . OOO q_{m+1} O. \end{aligned}$$

So

$$\Psi_{Z_1}(X_1, \ldots, X_m) = \langle \mathrm{Res}_{Z_1}(q_1 \overline{(X_1, \ldots, X_m)}) \rangle = X_1.$$

Example 6. 9. *Other, intermediate coordinates.* For $i=2, 3, \ldots, m-1$ Z_i will give the i-th coordinate of every m-tuple $(X_1, \ldots, X_i, \ldots, X_m)$. Z_i is:

$$\left.\begin{array}{ll} q_\nu O R q_\nu & \\ q_\nu S_\mu O q_\nu, & \mu=0, 1, \ldots, n-1 \\ q_\nu * O q_{\nu+1} & \end{array}\right\} \nu=1, 2, \ldots, i-1, i+1, \ldots, m+1$$

$$q_i O R q_i$$

$$q_i S_\mu R q_i \quad \mu=0,1, \ldots, n-1$$

$$q_i * O q_{i+1}$$

$$q_m O R q_{m+1}$$

$$q_{m+1} S_\mu O q_{m+2}, \quad \mu=0,1, \ldots, n-1$$

$$q_{m+2} O R q_{m+1}.$$

It is easy to show that

$$< \operatorname{Res} z_i[q_1(\overline{X_1, \ldots, X_i, \ldots, X_m})] > = X_i.$$

Example 6. 10. *Last coordinate.* The algorithm Z_m computes the last coordinate X_m of every m-tuple $(X_1, X_2, \ldots X_m)$. Z_m is

$$\left.\begin{array}{ll} q_\nu O R q_\nu & \\ q_\nu S_\mu O q_\nu, & \mu=0,1, \ldots, n-1 \\ q_\nu * O q_{\nu+1} & \end{array}\right\} \nu=1,2, \ldots, m-1.$$

We shall not exhibit more simple *Turing* algorithms because we shall need only the quoted ones.

7. Relativization. In this section we introduce also the quadruples of the type (4) in the definition 4. 2. As always we take $\mathfrak{S}=\{S_0, S_1, \ldots, S_{n-1}\}$ as the printing alphabet. Here we introduce a set $\mathfrak{A}$ of words which are $\in \Omega(\mathfrak{S})$, and we show how to relate the quadruples of the type (4) to such a set. (Compare [2], sect. 4 of Ch. 1).

Definition 7. 1. *Let α, β, be instantaneous descriptions. Then we write $Z: \alpha \underset{\mathfrak{A}}{\vdash} \beta$ if there exist expressions P and Q (possibly empty) such that α is $Pq_i S_j Q$, where Z contains $q_i S_j q_k q_l$, and either*

(1) $\langle\alpha\rangle \in \mathfrak{A}$ *and* β *is* $Pq_k S_j Q$, or

(2) $\langle\alpha\rangle \notin \mathfrak{A}$ *and* β *is* $Pq_l S_j Q$.

Definition 7. 2. *Let α be an instantaneous description of the form $Pq_i S_j Q$. Then, α is final with respect to Z if Z is a Turing algorithm which contains no quadruple whose initial pair of symbols is $q_i S_j$.*

Theorem 7. 1. *α is final with respect to Z if and only if* (1) *α is terminal with respect to Z, and* (2) *no matter which set $\mathfrak{A}$ is chosen, there is no β such that $Z: \alpha \underset{\mathfrak{A}}{\vdash} \beta$.*

Definition 7. 3. *By an* $\mathfrak{A}$*-computation of a Turing algorithm (or machine) Z is meant a finite sequence* $\alpha_1, \alpha_1, \ldots . \alpha_p$ *of instantaneous descriptions such that, for each i,* $1 \leqslant i < p$, *either* $Z: \alpha_i \vdash \alpha_{i+1}$ *or* $Z: \alpha_i \underset{\mathfrak{A}}{\vdash} \alpha_{+1}$, *and such that* α_p *is final. In this case we write* $\alpha_p = \mathrm{Res}_{Z:\mathfrak{A}}(\alpha_1)$, *and we call* α_p *the* $\mathfrak{A}$*-resultant* α_1 *with respect to Z.*

Obviously, if Z is a simple *Turing* algorithm then an $\mathfrak{A}$-computation is simply a computation, because Z does not contain any quadruple of the form $q_i S_j q_k q_e$.

Definition 7. 4. *Let Z be a Turing algorithm. Then, for each m, we associate with Z an m-ary word-function (which, in general, depends on the set* $\mathfrak{A}$) $\Psi_{Z:\mathfrak{A}}(X_1, X_2, \ldots, X_m)$ *as follows:*

For each m-tuple $(A_1, A_2, \ldots, A_m)$ *of words of* $\Omega(\mathfrak{S})$ *we set* $\alpha_1 = q_1\ (A_1, A_2, \ldots, A_m)$ *and we distinguish between two cases:*

(1) *There exists an* $\mathfrak{A}$*-computation of Z,* $\alpha_1, \alpha_2, \ldots, \alpha_p$. *In this case we set*

$$\Psi_{Z:\mathfrak{A}}(A_1, A_2, \ldots, A_m) = \langle\alpha_r\rangle = \langle \mathrm{Res}_{Z:\mathfrak{A}}(\alpha_1)\rangle.$$

(2) *There exists no* $\mathfrak{A}$*-computation of Z, beginning with* α_1, *i.e.* $\mathrm{Res}_{Z:\mathfrak{A}}(\alpha_1)$ *is undefined. In this case we leave* $\Psi_{Z:\mathfrak{A}}(A_1, A_2, \ldots, A_m)$ *undefined also.*

Obviously, if Z is simple then, for any set $\mathfrak{A}$,

$$\Psi_{Z:\mathfrak{A}}(X_1, \ldots, X_m) \simeq \Psi_Z(X_1, \ldots, X_m).$$

Definition 7. 5. *An m-ary word-function* $f(X_1, \ldots, X_m)$ *is partially* $\mathfrak{A}$*-algorithmic if there exists a Turing algorithm Z such that*

$$f(X_1, \ldots, X_m) \simeq \Psi_{Z:\mathfrak{A}}(X_1, \ldots, X_m).$$

In this case we say that Z $\mathfrak{A}$*-computes (or* $\mathfrak{A}$*-generates) f.*

If, in addition, f is total, then it is called $\mathfrak{A}$*-algorithmic.*

Theorem 7. 2.*To every Turing algorithm Z, there corresponds a simple Turing algorithm Z′ such that*

$$\Psi_{Z'}(X_1, \ldots, X_m) \simeq \Psi_{Z:\emptyset}(X_1, \ldots, X_m)$$

Proof. The same as in [2] Th. 4. 4 of Ch. 1.

We quote also

Theorem 7. 3. *If* $f(X_1, \ldots, X_m)$ *is (partially) algorithmic, then it is (partially)* $\mathfrak{A}$*-algorithmic for any set* $\mathfrak{A}$.

If $f(X_1, \ldots, X_m)$ *is (partially)* $\emptyset$*-algorithmic, then tt is (partially) algorithmic.*

($\emptyset$ is the symbol for the empty set of words).

On the ground of this theorem, every theorem about $\mathfrak{A}$-algorithmicity is in the same time a theorem about algorithmicity (with $\mathfrak{A} = \emptyset$). So it suffices to give only theorems about $\mathfrak{A}$-algorithmicity, where $\mathfrak{A}$ is some set of words of $\Omega(\mathfrak{S})$.

Definition 7. 6. *A set* $\mathfrak{L}$ *of words which are* $\in \Omega(\mathfrak{S})$ *is* ($\mathfrak{A}$—) *algorithmic if its characteristic function* $C_{\mathfrak{L}}(X)$ *is* ($\mathfrak{A}$—) *algorithmic.*[1]

[1] $C_{\mathfrak{L}}(X) = O$ for $X \in \mathfrak{L}$ and $C_{\mathfrak{L}}(X) = S_0$ for $X \notin \mathfrak{L}$.

Theorem 7. 4. *Every set* $\mathfrak{A}$ *of words of* $\Omega(\mathfrak{S})$ *is* $\mathfrak{A}$*-algorithmic.*

Proof. Let Z consist of the quadruples

$$q_1 S_\nu q_2 q_3, \quad \nu = 0, 1, \ldots, n-1$$

$$q_2 S_\nu O q_4, \quad \nu = 0, 1, \ldots, n-1$$

$$q_4 O R q_2$$

$$q_3 S_\nu S_0 q_5, \quad \nu = 0, 1, \ldots, n-1$$

$$q_5 S_0 R q_2.$$

Let now $S_{i_1} S_{i_2}, \ldots, S_{i_k} \in \mathfrak{A}$. Then

$$\begin{aligned} Z: q_1 S_{i_1} S_{i_2} \ldots . S_{i_k} &\underset{\mathfrak{A}}{\vdash} q_2 S_{i_1} S_{i_2} \ldots S_{i_k} \\ &\vdash q_4 O S_{i_2} \ldots . S_{i_k} \\ &\vdash O q_2 S_{i_2} \ldots S_{i_k} \\ &\models O O O \ldots . O q_2 S_{i_k} \\ &\vdash O O O \ldots . O q_4 O \\ &\vdash \cdot O O O \ldots . O O q_2 O, \end{aligned}$$

and $\langle \operatorname{Res}_{Z:\mathfrak{A}} (q_1 S_{i_1} \ldots . S_{i_k}) \rangle = O$.

Let now $S_{i_1} S_{i_2} \ldots . S_{i_k} \notin \mathfrak{A}$. Then

$$\begin{aligned} Z: q_1 S_{i_1} S_{i_2} \ldots . S_{i_k} &\underset{\mathfrak{A}}{\vdash} q_3 S_{i_1} S_{i_2} \ldots . S_{i_k} \\ &\vdash q_5 S_0 S_{i_2} \ldots . S_{i_k} \\ &\vdash S_0 q_2 S_{i_2} \ldots . S_{i_k} \\ &\models \cdot S_0 O O \ldots . O q_2 O, \end{aligned}$$

and $\langle \operatorname{Res}_{Z:\mathfrak{A}} (q_1 S_{i_1} \ldots . S_{i_k}) \rangle = S_0$.

So $\Psi_Z : \mathfrak{A}(X) = C_{\mathfrak{A}}(X)$.

8. Regularization. With this section we begin to prove general theorems about *Turing* algorithms, which where only quoted in [1]. As we mentioned before, the most of them are simple (but technically more involved) generalisations of the corresponding theorems of [2].

In following we suppose that the printing alphabet is always

$$\mathfrak{S} = \{S_0, S_1, \ldots, S_{n-1}\}.$$

With *Davis* we shall adopt in the future the convention of systematical omitting of the final occurences of open square O in an instantaneous description. So, f. i. we shall write $S_1 q_3 S_2 S_0$ for $S_1 q_3 S_2 S_0 OOOOO$. (But, $S_1 q_5 O$ f. i. is not to be written as $S_1 q_5$). On the other hand, we shall not omit initial occurences of O. The reasons for this last convention are in the role which will be played by these initial occurences of O.

We give first some more definitions, which are slightly different from similar definitions of *Davis* ([2]).

Definition 8. 1. *If Z is a Turing algorithm we write $\Theta(Z)$ for the largest number i such that q_i is an internal configuration of Z.*

Definition 8. 2. *A Turing algorithm Z is called m-regular* $(m > 0)$ *if* (1) *There is an $s > 0$ such that, whenever*

$$\mathrm{Res}_{Z:\mathfrak{A}}[\overline{q_1(A_1, \ldots, A_m)}]$$

is defined, it has the form

$$q_{\Theta(Z)}\overline{(B_1, \ldots, B_s)},$$

or at most the form

$$Oq_{\Theta(Z)}\overline{(B_1, \ldots, B_s)},$$

for suitable words $B_1, B_2, \ldots, B_s$, and (2) *No quadruple of Z begins with $q_{\Theta(Z)}$.*

We point that our allowance of O at the beginning of the

$$q_{\Theta(Z)}\overline{(B_1, \ldots, B_s)}$$

is not essential; we allowed it only to shorten some algorithms, but we shall prove that it can always be deleted by a new algorithm.

Definition 8. 3. *Let Z be a Turing algorithm. Then $Z^{(p)}$ is the Turing algorithm obtained from Z by replacing each internal configuration q_i, at all its occurrences in quadruples of Z by q_{p+i}.*

We prove now the first theorem about regularization.

Theorem 8. 1. *For every Turing algorithm Z, we can find a Turing algorithm Z' such that, for each m, Z' is m-regular, and, in fact*

$$\mathrm{Res}_{Z':\mathfrak{A}}[\overline{q_1(A_1, \ldots, A_m)}] \simeq Oq_{\Theta(Z')}\Psi_{Z:\mathfrak{A}}(A_1, \ldots, A_m).$$

Proof. The idea of the proof is the same as in the proof of the corresponding Lemma 1 of Ch. 2 of [2]. We begin by putting markers λ and ρ to the ends of input. Then we let Z work, taking into account the eventual disturbing influence of λ and ρ. After this, we erase all auxiliar letters and put the remaining ones close one to another, going to the beginning as to have only one O before $q_{\Theta(Z')}$.

We introduce the letters λ and ρ as the first two letters in the part of the auxiliary alphabet, beginning with S_{n+2}, that are not in the alphabet of Z. (S_n is excluded, as it denotes open squares, and $S_{n+1} = *$ is excluded also, as it serves to represent m-tuples $(A_1, \ldots, A_m)$).

Let Z_1 be the algorithm:

$q_1 S_\nu L q_2$, $\quad \nu = 0, 1, \ldots, n-1, n \quad (S_n = O)$

$q_2 O \lambda q_2$ $\quad$ (print λ on the left end)

$q_2 \lambda R q_3$

$q_3 S_\nu R q_3$, $\quad \nu = 0, 1, \ldots, n-1$

$q_3 O R q_4$

${}_3 * R q_3$

$q_4 * R q_3$

$q_4 O L q_5$

$q_5 O \rho q_5$ (print ρ on the right end)

$q_5 \rho L q_6$

$q_6 S_\nu L q_6$ $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_6 \lambda R q_7$ (move left until λ is reached).

We have:

$$
\begin{aligned}
Z_1 : & \, q_1 A_1 * A_2 * \ldots * A_m \\
& \vdash q_2 O A_1 * A_2 * \ldots * A_m \\
& \vdash q_2 \lambda A_1 * A_2 * \ldots * A_m \\
& \vdash \lambda q_3 A_1 * A_2 * \ldots * A_m \\
& \models \lambda A_1 * A_2 * \ldots * A_m q_3 O \\
& \vdash \lambda A_1 * A_2 * \ldots * A_m O q_4 O \\
& \vdash \lambda A_1 * A_2 * \ldots * A_m q_5 O \\
& \vdash \lambda A_1 * A_2 * \ldots * A_m q_5 \rho \\
& \models q_6 \lambda A_1 * A_2 * \ldots * A_m \rho \\
& \vdash \cdot \lambda q_7 A_1 * A_2 * \ldots * A_m \rho .
\end{aligned}
$$

Thus, the effect of Z_1 is to seal the initial instantaneous description with the letters λ and ρ.

Now, $Z^{(6)}$ will behave precisely like Z except that it will begin in the internal configuration q_7 instead of q_1 and the index of all of its other internal configurations will be similarly advanced. Thus, we set $k = \Theta(Z^{(6)})$, and we let Z_2 consist of all the quadruples of $Z^{(6)}$ and, in addition, the following quadruples, where q_i may be any internal configuration of $Z^{(6)}$:

$q_i \lambda O q_{k+i}$ (erase the marker λ)

$q_{k+i} O L q_{2k+i}$

$q_{2k+i} O \lambda q_{2k+i}$ (print λ one square to the left)

$q_{2k+i} \lambda R q_i$ (return to the main computation)

$q_i \rho O q_{3k+i}$ (erase the marker ρ)

$q_{3k+i} O R q_{4k+i}$

$q_{4k+i} O \rho q_{4k+i}$ (print ρ one square to the right)

$q_{4k+i} \rho L q_i$ (return to the main computation).

These last quadruples serve to neutralize the eventual influence of λ and ρ on the work of the algorithm $Z^{(6)}$. Now, either $\mathrm{Res}_{Z:\mathfrak{A}} [q_1 (\overline{A_1, \ldots A_m})]$ is defined, in which case we have

(1) $$Z_2 : \lambda q_7 (\overline{A_1, \ldots, A_m}) \rho \models_{\mathfrak{A}} \cdot \lambda \alpha \rho ,$$

where

(2) $$< \alpha > = < \mathrm{Res}_{Z:\mathfrak{A}} [q_1 (\overline{A_1, \ldots, A_m})] > ,$$

or $\text{Res}_{Z:\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}]$ is undefined, in which case $\text{Res}_{Z_2:\mathfrak{A}}[\lambda q_7\overline{(A_1, \ldots, A_m)}\rho]$ is likewise undefined.

Let $L = 5k+1$, and let Z_3 consist of all quadruples of the form

$$q_i S_j S_j q_L$$

where q_i is any internal configuration of Z_2, where S_j belongs to the alphabet of Z_2 (so as to the printing alphabet also to the auxiliar alphabet, inclusive $S_n = 0$ and $S_{n+1} = *$), and where no quadruple beginning with $q_i S_j$ belongs to Z_2.

If $\lambda P q_i Q \rho$ is a final instantaneous description with respect to Z_2, we have

(3) $$Z_3 : \lambda P q_i Q \rho \vdash \cdot \lambda P q_L Q \rho .$$

Let now Z_4 consists of the following quadruples, where S may be any letter in the alphabet of Z, different from $S_0, S_1, \ldots, S_{n-1}$, 0, λ and ρ. (So it can be also $S_{n+1} = *$):

$q_L S_\nu L q_L$, $\nu = 0, 1, \ldots, n-1, n$

$q_L S L q_L$ (move leftward looking for λ)

$q_L \lambda R q_{L+1}$

$q_{L+1} S O q_{L+1}$ (erase all letters different from the quoted ones)

$q_{L+1} S_\nu R q_{L+1}$, $\nu = 0, 1, \ldots, n-1, n$

$q_{L+1} \rho \rho q_{L+2}$

$q_{L+2} \rho L q_{L+2}$

$q_{L+2} S_\nu L q_{L+2}$, $\nu = 0, 1, \ldots, n-1, n$

$q_{L+2} \lambda R q_{L+3}$ (after erasing, prepare to transport to the left)

$q_{L+3} O R q_{L+3}$

$q_{L+3} S_\nu O q_{L+\nu+4}$, $\nu = 0, 1, \ldots, n-1$ (memorize a letter)

$q_{L+\nu+4} O L q_{L+\nu+4}$, $\nu = 0, 1, \ldots, n-1$

$q_{L+\nu+4} S_\mu R q_{L+n+\nu+4}$, $\mu, \nu = 0, 1, \ldots, n-1$

$q_{L+\nu+4} \lambda R q_{L+n+\nu+4}$, $\nu = 0, 1, \ldots, n-1$

$q_{L+n+\nu+4} O S_\nu q_{L+3n+3}$, $\nu = 0, 1, \ldots, n-1$ (print the memorized letter)

$q_{L+3n+3} S_\nu R q_{L+3}$, $\nu = 0, 1, \ldots, n-1$ (return to the transport)

$q_{L+3} \rho O q_{L+3n+4}$ (the transport being finished, erase ρ)

$q_{L+3n+4} S_\nu L q_{L+3n+4}$, $\nu = 0, 1, \ldots, n-1, n$ (go left)

$q_{L+3n+4} \lambda O q_{L+3n+5}$ (erase λ)

$q_{L+3n+5} O R q_{L+3n+6}$ (terminate with maximal q_i).

If there is a $\text{Res}_{Z_3:\mathfrak{A}}$ then we have

$$Z_4 : \lambda P q_L Q \rho \models \lambda q_{L+1} P Q \rho$$

and now Z_4 does erase all letters which are different from

$$S_0, S_1, \ldots, S_{n-1}, O, \lambda, \rho .$$

For definiteness, let us have

$$Z_4 : \lambda P q_L Q \rho \models \lambda q_{L+3} OOS_{i_1} S_{i_2} OS_{i_3} OO\rho$$

where $0 \leq i_1,\ i_2,\ i_3 \leq n-1$. We have further

$$Z_4 : \lambda q_{L+3} OOS_{i_1} S_{i_2} OS_{i_3} OO\rho$$
$$\models \lambda OOq_{L+3} S_{i_1} S_{i_2} OS_{i_3} OO\rho$$
$$\vdash \lambda OO q_{L+i_1+4} OS_{i_2} OS_{i_3} OO\rho$$
$$\models q_{L+i_1+4} \lambda OOO S_{i_2} OS_{i_3} OO\rho$$
$$\vdash \lambda q_{L+n+i_1+4} OOO S_{i_2} OS_{i_3} OO\rho$$
$$\vdash \lambda q_{L+3n+3} S_{i_1} OO S_{i_2} OS_{i_3} OO\rho$$
$$\vdash \lambda S_{i_1} q_{L+3} OO S_{i_2} OS_{i_3} OO\rho$$
$$\models \lambda S_{i_1} S_{i_2} S_{i_3} OOOOO q_{L+3}\rho$$
$$\vdash \lambda S_{i_1} S_{i_2} S_{i_3} OOOOO q_{L+3n+4} O$$
$$\models q_{L+3n+4} \lambda S_{i_1} S_{i_2} S_{i_3} \quad \text{(we omit } O\text{-s at the end)}$$
$$\vdash q_{L+3n+5} O S_{i_1} S_{i_2} S_{i_3}$$
$$\vdash \cdot O q_{L+3n+6} S_{i_1} S_{i_2} S_{i_3}$$

Finally, let $Z' = Z_1 \cup Z_2 \cup Z_3 \cup Z_4$. Then, combining (1) to (3) with the role of Z_1 and Z_4 we have

$$\mathrm{Res}_{Z':\mathfrak{A}} [q_1 \overline{(A_1, \ldots, A_m)}] = O q_{L+3n+6} \langle \mathrm{Res}_{Z:\mathfrak{A}} [q_1 \overline{(A_1, A_2, \ldots, A_m)}] \rangle =$$
$$= O q_{\Theta(Z')} \Psi_{Z:\mathfrak{A}} (A_1, A_2, \ldots, A_m),$$

which proves the theorem.

We show now that it is possible to omit the symbol O at the beginning.

Theorem 8. 2. *For every m-regular Turing algorithm Z for which*

$$\mathrm{Res}_{Z:\mathfrak{A}} [q_1 \overline{(A_1, \ldots, A_m)}] = Oq_{\Theta(Z)} A,$$

where $A \in \Omega(\mathfrak{S})$, *we can find an m-regular Turing algorithm* Z' *such that*

$$\mathrm{Res}_{Z':\mathfrak{A}} [q_1 \overline{(A_1, \ldots, A_m)}] = q_{\Theta(Z')} A.$$

Proof. We introduce the doubling alphabet $\lambda_0, \lambda_1, \ldots, \lambda_{n-1}$, where λ_0 is the first letter in the sequence $S_{n+2}, S_{n+3}, \ldots$ which is not in the auxiliary alphabet of Z, and λ_1 the next one, and so on. Let $N = \Theta(Z)$ and let Z_1 be the algorithm:

$q_N OLq_{N+4n+6}$ (if A is empty, terminate)

$q_N S_\nu Oq_{N+1+\nu}$, $\nu = 0, 1, \ldots, n-1$ (memorize the first letter)

$q_{N+1+\nu} OLq_{N+n+\nu+1}$, $\nu = 0, 1, \ldots, n-1$

$q_{N+n+\nu+1} O\lambda_\nu q_{N+2n+1}$, $\nu = 0, 1, \ldots, n-1$ (print the first letter in doubling alphabet)

$q_{N+2n+1} \lambda_\nu Rq_{N+2n+2}$ $\nu = 0, 1, \ldots, n-1$

$q_{N+2n+2} ORq_{N+2n+3}$

$q_{N+2n+3} S_\nu O q_{N+2n+\nu+4}$, $\nu=0, 1, \ldots, n-1$ (memorize a letter)

$q_{N+2n+\nu+4} O L q_{N+3n+\nu+4}$, $\nu=0, 1, \ldots, n-1$ (go left one sqnare)

$q_{N+3n+\nu+4} O S_\nu q_{N+4n+4}$, $\nu=0, 1, \ldots, n-1$ (print the letter)

$q_{N+4n+4} S_\nu R q_{N+2n+2}$ (search for other letters)

$q_{N+2n+3} O L q_{N+4n+5}$

$q_{N+4n+5} S_\nu L q_{N+4n+5}$, $\nu=0, 1, \ldots, n-1$

$q_{N+4n+5} \lambda_\nu S_\nu q_{N+4n+6}$, $\nu=0,1, \ldots, n-1$ (terminate).

Let $Z'=Z \cup Z_1$ and let

$$\mathrm{Res}_{Z:\mathfrak{A}}[q_1 \overline{(A_1, \ldots, A_m)}] = O q_N S_{i_1} S_{i_2} \ldots\ldots S_{ik},$$

where $0 \leqslant i_1, i_2, \ldots, i_k \leqslant n-1$. Then

$$\begin{aligned}
&Z' : q_1 \overline{(A_1, \ldots, A_m)} \\
&\models_{\mathfrak{A}} O q_N S_{i_1} S_{i_2} \ldots\ldots S_{ik} \\
&\vdash O q_{N+1+i_1} O S_{i_2} S_{i_3} \ldots S_{i_k} \\
&\vdash q_{N+n+1+i_1} O O S_{i_2} \ldots\ldots S_{i_k} \\
&\vdash q_{N+2n+1} \lambda_{i_1} O S_{i_2} \ldots\ldots S_{i_k} \\
&\vdash \lambda_{i_1} q_{N+2n+2} O S_{i_2} \ldots\ldots S_{i_k} \\
&\vdash \lambda_{i_1} O q_{N+2n+3} S_{i_2} S_{i_3} \ldots\ldots S_{i_k} \\
&\vdash \lambda_{i_1} O q_{N+2n+i_2+4} O S_{i_3} \ldots\ldots S_{ik} \\
&\vdash \lambda_{i_1} q_{N+3n+i_2+4} O O S_{i_3} \ldots\ldots S_{i_k} \\
&\vdash \lambda_{i_1} q_{N+4n+4} S_{i_2} O S_{i_3} \ldots\ldots S_{i_k} \\
&\vdash \lambda_{i_1} S_{i_2} q_{N+2n+2} O S_{i_3} \ldots\ldots S_{i_k} \\
&\vdash \lambda_{i_1} S_{i_2} O q_{N+3n+3} S_{i_3} \ldots\ldots S_{i_k} \\
&\models \lambda_{i_1} S_{i_2} S_{i_3} \ldots\ldots S_{i_k} q_{N+3n+2} O \\
&\vdash \lambda_{i_2} S_{i_2} S_{i_3} \ldots\ldots S_{i_k} O q_{N+2n+3} O \\
&\vdash \lambda_{i_1} S_{i_2} S_{i_3} \ldots\ldots S_{i_k} q_{N+4n+5} O \\
&\models q_{N+4n+5} \lambda_{i_1} S_{i_3} S_{i_2} \ldots\ldots S_{i_k} \\
&\vdash \cdot q_{N+4n+6} S_{i_1} S_{i_2} \ldots\ldots S_{i_k} = q_{\Theta(Z')} S_{i_1} \ldots\ldots S_{i_k},
\end{aligned}$$

which proves the theorem 8. 2.

Combining theorems 8. 1. and 8. 2. we get

Theorem 8. 3. *For every Turing algorithm Z, we can find a Turing algorithm Z', such that, for each m, Z' is m-regular, and, in fact*

$$\mathrm{Res}_{Z':\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}] \simeq q_{\Theta(Z')}\Psi_{Z:\mathfrak{A}}(A_1, \ldots, A_m).$$

We remark also that in the theorem 8. 2. the condition $A \in \Omega(\mathfrak{S})$ can be replaced also by $A \in \Omega(\mathfrak{S} \cup \{*\})$. So, we have

Theorem 8. 4. *For every m-regular Turing algorithm Z for which*

$$\mathrm{Res}_{Z:\mathfrak{A}}[q_1\overline{(A_1, A_2, \ldots, A_m)}] = O q_{\Theta(Z)}\overline{(B_1, \ldots, B_s)},$$

where $B_i \in \Omega(\mathfrak{S})$, $i = 1, 2, \ldots, s$, *we can find an m-regular Turing algorithm Z' such that*

$$\mathrm{Res}_{Z':\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}] = q_{\Theta(Z')}\overline{(B_1, \ldots, B_s)}.$$

9. Untouched variables. In this section we prove the analogue of lemma 2, Ch. 2 of [2]. The proof iz more involved, because we have to employ a larger doubling alphabet. All words are suposed to be $\in \Omega(\mathfrak{S})$.

Theorem 9. 1. *For each m-regular Turing algorithm Z and each* $p > 0$, *there is a* $(p+m)$ — *regular Turing algorithm* Z_p *such that, whenever*

$$\mathrm{Res}_{Z:\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}] = q_{\Theta(Z)}\overline{(B_1, \ldots, B_s)}$$

it is also the case that

$$\mathrm{Res}_{Z_p:\mathfrak{A}}[q_1\overline{(C_1, \ldots, C_p, A_1, \ldots, A_m)}] = q_{\Theta(Z_p)}\overline{(C_1, \ldots, C_p, B_1, \ldots, B_s)}$$

whereas, whenever $\mathrm{Res}_{Z:\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}]$ *is undefined, so is*

$$\mathrm{Res}_{Z_p:\mathfrak{A}}[q_1\overline{(C_1, \ldots, C_p, A_1, \ldots, A_m)}].$$

Proof. The idea of the proof is to write $C_1, C_2, \ldots, C_p$ in a doubling λ-alphabet and to let then Z work with the remaining arguments.

Let T be the index of the first letter S_i, $i \geqslant n+2$, which is not in the auxiliar alphabet of Z. We introduce the doubling alphabet

$$\lambda_0 = S_T,\ \lambda_1 = S_{T+1}, \ldots, \lambda_{n-1} = S_{T+n-1},$$

and the letters $\lambda = S_{T+n}$, $\rho = S_{T+n+1}$. If A is the word $S_{i_1} S_{i_2}, \ldots, S_{i_k}$, with $\bar{A}$ we denote the word $\lambda_{i_1}\lambda_{i_2}, \ldots, \lambda_{i_k}$, i.e. the word A written in the doubling alphabet.

Let U_1 be the algorithm:

$q_1 S_\nu L q_2$, $\nu = 0, 1, \ldots, n-1, n$

$q_2 O \lambda q_2$ (set marker λ at beginning)

$q_2 \lambda R q_3$

$$\left.\begin{array}{ll} q_i S_\nu \lambda_\nu q_i, & \nu = 0, 1, \ldots, n-1 \\ q_i \lambda_\nu R q_i, & \nu = 0, 1, \ldots, n-1 \\ q_i O R q_i & \\ q_i * R q_{i+1} & \end{array}\right\} i = 3, 4, \ldots, p+1$$

(transcript the first $p-1$ words into doubling alphabet)

$q_{p+2} S_\nu \lambda_\nu q_{p+2}$, $\nu = 0, 1, \ldots, n-1$ (transcript the p—th word into doubling alphabet)

$q_{p+2} O R q_{p+2}$

$q_{p+2} * \rho q_{p+2}$ (erase $*$ between the first p words and the following m words, and set ρ for it)

$q_{p+2} \rho R q_{p+3}$.

We have:

$U_1 : q_1 C_1 * C_2 * \cdots * C_p * A_1 * \cdots * A_m$

$\vdash q_2 O C_1 * C_2 * \cdots * C_p * A_1 * \cdots * A_m$

$\vdash q_2 \lambda C_1 * C_2 * \cdots * C_p * A_1 * \cdots * A_m$

$\vdash \lambda q_3 C_1 * C_2 * \cdots * C_p * A_1 * \cdots * A_m$

$\models \cdot \lambda \bar{C}_1 * \bar{C}_2 * \cdots * \bar{C}_p \rho q_{p+3} A_1 * \cdots * A_m,$

i. e. U_1 sets markes λ and ρ at the ends of $\overline{(C_1, \ldots, C_p)}$ and does transcript every word of this p-tuple into doubling alphabet. The internal configuration q_{p+3} stands at the beginning of the m-tuple $(A_1, \ldots, A_m)$.

Next, let $M = \Theta(Z^{(p+2)})$ and let U_2 consist of all the quadruples of $Z^{(p+2)}$ and, in addition, of the following quadruples, where q_i may be any internal configuration of $Z^{(p+2)}$; (in these quadruples $N = Max(M, n) \geqslant 3$, where n is the number of letters of the alphabet $\mathfrak{S}$. We write there $q(i)$ for q_i, for typographical grounds. We profit the fact that $2^N > N$ for $N \geqslant 3$):

$q_i \rho \rho q(2^N \cdot 3^i)$ (interrupt computation)

$q(2^N \cdot 3^i) \rho L q(2^N \cdot 3^i)$

$q(2^N \cdot 3^i) \lambda_\nu L q(2^N \cdot 3^i), \nu = 0, 1, \ldots, n-1$ (search for λ)

$q(2^N \cdot 2^i) * L q(2^N \cdot 3^i)$

$q(2^N \cdot 3^i) OLq(2^N \cdot 3^i)$

$q(2^N \cdot 3^i) \lambda Oq(2^{N+1} \cdot 3^i)$ (erase λ)

$q(2^{N+1} \cdot 3^i) OLq(2^{N+2} \cdot 3^i)$ (go left)

$q(2^{N+2} \cdot 3^i) O\lambda q(2^{N+2} \cdot 3^i)$ (set λ one square left)

$q(2^{N+2} \cdot 3^i) \lambda Rq(2^{N+3} \cdot 3^i)$

$q(2^{N+3} \cdot 3^i) ORq(2^{N+4} \cdot 3^i)$

$q(2^{N+4} \cdot 3^i) ORq(2^{N+5} \cdot 3^i)$

$q(2^{N+5} \cdot 3^i) * Oq(2^{N+6} \cdot 3^i)$

$q(2^{N+6} \cdot 3^i) OLq(2^{N+7} \cdot 3^i)$ (transport $*$ one square left)

$q(2^{N+7} \cdot 3^i) O * q(2^{N+7} \cdot 3^i)$

$q(2^{N+7} \cdot 3^i) * Rq(2^{N+3} \cdot 3^i)$

$q(2^{N+4} \cdot 3^i) \lambda_\nu Oq(2^{N+4} \cdot 3^i \cdot 5^{\nu+1}),$ $\nu = 0, 1, \ldots, n-1$

$q(2^{N+4} \cdot 3^i) * Oq(2^{N+4} \cdot 3^i \cdot 5^{n+1})$

$q(2^{N+4}\cdot 3^i\cdot 5^{\nu+1})\,OLq\,(2^{N+5}\cdot 3^i\cdot 5^{\nu+1}),\quad \nu=0,\,1,\,\ldots,\,n-1,\,n$

$q(2^{N+5}\cdot 3^i\cdot 5^{\nu+1})\,O\lambda_\nu q\,(2^{N+8}\cdot 3^i),\quad \nu=0,\,1,\,\ldots,\,n-1$ (transport left for one square)

$q(2^{N+5}\cdot 3^i\cdot 5^{n+1})\,O*q\,(2^{N+8}\cdot 3^i)$

$q(2^{N+8}\cdot 3^i)\,\lambda_\nu Rq\,(2^{N+3}\cdot 3^i),\quad \nu=0,\,1,\,\ldots,\,n-1$

$q(2^{N+8}\cdot 3^i)*Rq\,(2^{N+3}\cdot 3^i)$

$q(2^{N+4}\cdot 3^i)\,\rho\,Oq\,(2^{N+9}\cdot 3^i)$

$q(2^{N+9}\cdot 3^i)\,OLq\,(2^{N+10}\cdot 3i)$

$q(2^{N+10}\cdot 3^i)\,O\,\rho\,q\,(2^{N+10}\cdot 3^i)$

$q(2^{N+5}\cdot 3^i)\,\rho\,O\,q\,(2^{N+9}\cdot 3^i)$

$q(2^{N+10}\cdot 3^i)\,\rho\,Rq_i$ (return to the main computation).

For the sake of definiteness we shall exhibit the work of U_2 on a concrete example:

$U_2:\ \lambda\,O*\lambda_{i_1}\lambda_{i_2}*O\,q_i\,\rho\,A$

$\vdash \lambda\,O*\lambda_{i_1}\lambda_{i_2}*O\,q\,(2^N\cdot 3^i)\,\rho\,A$

$\models q(2^N\cdot 3^i)\,\lambda\,O*\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash q\,(2^{N+1}\cdot 3^i)\,OO*\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash q\,(2^{N+2}\cdot 3^i)\,OOO*\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash q\,(2^{N+2}\cdot 3^i)\,\lambda\,OO*\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda\,q\,(2^{N+3}\cdot 3^i)\,OO*\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda\,O\,q\,(2^{N+4}\cdot 3^i)\,O*\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda\,OO\,q\,(2^{N+5}\cdot 3^i)*\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda\,OO\,q\,(2^{N+6}\cdot 3^i)\,O\,\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda\,O\,q\,(2^{N+7}\cdot 3^i)\,OO\,\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda\,O\,q\,(2^{N+7}\cdot 3^i)*O\,\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda O*q\,(2^{N+3}\cdot 3^i)\,O\,\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda\,O*O\,q\,(2^{N+4}\cdot 3^i)\,\lambda_{i_1}\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda\,O*O\,q\,(2^{N+4}\cdot 3^i\cdot 5^{i_1+1})\,O\,\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda O*q\,(2^{N+5}\cdot 3^i\cdot 5^{i_1+1})\,OO\,\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda O*q\,(2^{N+8}\cdot 3^i)\,\lambda_{i_1}\,O\,\lambda_{i_2}*O\,\rho\,A$

$\vdash \lambda O*\lambda_{i_1}\,q\,(2^{N+3}\cdot 3^i)\,O\,\lambda_{i_2}*O\,\rho\,A$

$\models \lambda O*\lambda_{i_1}\lambda_{i_2}*Oq\,(2^{N+10}\cdot 3^i)\,\rho\,OA$

$\vdash \cdot\,\lambda O*\lambda_{i_1}\lambda_{i_2}*O\,\rho\,q_i\,OA.$

So, the effect of this part of U_2 is to transport everything left from ρ for one square left, as well as to let $Z^{(p+2)}$ work freely. In U_2 we can at most occupy the internal configuration $q(2^{N+10}\cdot 3^N\cdot 5^{N+1})=q(Q)$, where

$$Q=2^{N+10}\cdot 3^N\cdot 5^{N+1}.$$

Let $U_3=U_1\cup U_2$. Then we have

$$U_3: q_1 C_1 * C_2 * \cdots * C_p * A_1 * \cdots * A_m$$

$$\models \lambda \overline{C_1} * \overline{C_2} * * \cdots * \overline{C_p}\, \rho\, q_{p+3} A_1 * \cdots * A_m$$

$$\underset{\mathfrak{A}}{\models} \cdot \lambda \overline{C_1} * \overline{C_2} * \cdots * \overline{C_p}\, \rho\, q_M B_1 * B_2 * \cdots * B_s$$

whenever $\mathrm{Res}_{Z:\mathfrak{A}}[q_1\overline{(A_1, \ldots , A_m)}]$ is defined; otherwise there is no $\mathfrak{A}$-computation on quoted arguments.

Let U_4 be the algorithm

$q_M S_\nu L q_{Q+1}$, $\nu=0, 1, \ldots , n-1, n$

$q_{Q+1}\, \rho * q_{Q+2}$

$q_{Q+2} * L q_{Q+3}$

$q_{Q+3}\, O L q_{Q+3}$

$q_{Q+3} \lambda_\nu S_\nu q_{Q+4}$, $\nu=0, 1, \ldots , n-1$ (transcript every λ_ν into S_ν)

$q_{Q+4} S_\nu L q_{Q+3}$

$q_{Q+3} * L q_{Q+3}$

$q_{Q+3} \lambda O q_{Q+5}$

$q_{Q+5}\, O R q_{Q+6}$.

Let now Z'' be $U_3\cup U_4$. We have

$$Z'': q_1 C_1 * C_2 * \cdots * C_p * A_1 * \cdots * A_m$$

$$\underset{\mathfrak{A}}{\models} \lambda \overline{C_1} * \overline{C_2} * \cdots * \overline{C_p}\, \rho\, q_M B_1 * \cdots B_s$$

$$\vdash \lambda \overline{C_1} * \overline{C_2} * \cdots * \overline{C_p}\, q_{Q+1}\, \rho\, B_1 * \cdots * B_s$$

$$\vdash \lambda \overline{C_1} * \overline{C_2} * \cdots * \overline{C_p}\, q_{Q+2} * B_1 * \cdots * B_s$$

$$\models q_{Q+3} \lambda C_1 * C_2 * \cdots * C_p * B_1 * \cdots * B_s$$

$$\models \cdot O q_{\Theta(Z'')}\overline{(C_1, C_2, \ldots , C_p, B_1, \ldots , B_s)}.$$

Applying now theorem 8. 4 we have the assertion of the theorem 9. 1 proved.

10. The copying algorithms. These algorithms double some arguments and transport them appropiately.

Theorem 10. 1. *For each* $m>0$ *and* $p\geqslant 0$ *there exists an* $(m+p)$-*regular Turing algorithm* $\mathfrak{L}_p$ *such that*

$$\mathrm{Res}_{\mathfrak{L}_p}[q_1\overline{(B_1, \ldots , B_p, A_1, \ldots , A_m)}]$$

$$=q_{\Theta(\mathfrak{L}_p)}\overline{(A_1, \ldots , A_m, B_1, \ldots , B_p, A_1, \ldots , A_m)}.$$

Proof. The idea of the proof is to transport $(A_1, \dots, A_m)$ to the left by doubling it. We introduce the auxiliary letters $\lambda = S_{n+2}$ and $\tau = S_{n+3}$ and the doubling alphabet

$$\lambda_0 = S_{n+4},\ \lambda_1 = S_{n+5},\ \dots,\ \lambda_{n-1} = S_{2n+3},\ \lambda_n = S_{2n+4},\ \lambda_{n+1} = S_{2n+5}.$$

(λ_n will double O, λ_{n+1} will double $*$). We introduce also $\rho = S_{2n+6}$.

We treat first the case $p > O$. $\mathfrak{L}_p'$ will be:

$q_1 S_\nu L q_2, \quad \nu = 0, 1, \dots, n-1, n$

$q_2 O * q_2$ (print $*$ before B_1)

$q_2 * L q_{p+n+16}$

$q_{p+n+16} O \lambda q_{p+n+16}$

$q_{p+n+16} \lambda R q_{p+n+17}$ (print λ before $*$)

$q_{p+n+17} * R q_3$

(+)

$q_{2+i} S_\nu R q_{2+i}, \quad \nu = 0, 1, \dots, n-1, n$ } $i = 1, 2, \dots, p-1$ for $p \geqslant 2$ (for $p = 1$ these quadruples are omitted)

$q_{2+i} * R q_{2+i+1}$

$q_{p+2} O R q_{p+2}$

$q_{p+2} S_\nu R q_{p+2}, \quad \nu = 0, 1, \dots, n-1$

$q_{p+2} * \rho q_{p+3}$

$q_{p+3} \rho R q_{p+4}$ (B_p and A_1 are separated by ρ)

$q_{p+4} S_\nu R q_{p+5}, \quad \nu = 0, 1, \dots, n-1, n$

$q_{p+5} * R q_{p+4}$

$q_{p+5} S_\nu R q_{p+5}, \quad \nu = 0, 1, \dots, n-1$

$q_{p+5} O \tau q_{p+5}$ (put marker τ at the end)

$q_{p+5} \tau L q_{p+6}$

$p_{p+6} S_\nu \lambda_\nu q_{p+\nu+7}, \quad \nu = 0,1, \dots, n-1, n, n+1$

$q_{p+\nu+7} S_i L q_{p+\nu+7}, \quad \nu, i = 0,1, \dots, n-1, n, n+1$

$q_{p+\nu+7} \lambda_\nu L q_{p+\nu+7}, \quad \nu = 0,1, \dots, n-1, n, n+1$

$q_{p+\nu+7} \rho L q_{p+\nu+7}, \quad \nu = 0,1, \dots, n-1, n, n+1$

$q_{p+\nu+7} \lambda S_\nu q_{p+n+9}, \quad \nu = 0,1, \dots, n-1, n, n+1$

$q_{p+n+9} S_\nu L q_{p+n+10} \quad \nu = 0,1, \dots, n-1, n, n+1$

$q_{p+n+10} O \lambda q_{p+n+10}$

$q_{p+n+10} \lambda R q_{p+n+11}$

$q_{p+n+11} S_\nu R q_{p+n+11}, \quad \nu = 0,1, \dots, n-1, n, n+1$

$q_{p+n+11} \rho R q_{p+n+11}$

$q_{p+n+11} \lambda_\nu L q_{p+6}$, $\nu = 0,1, \ldots, n-1, n, n+1$ (return to copy)

$q_{p+6} \rho * q_{p+n+12}$

$q_{p+n+12} * R q_{p+n+13}$

$q_{p+n+13} \lambda_\nu S_\nu q_{p+n+12}$, $\nu = 0,1, \ldots, n-1, n, n+1$

$q_{p+n+12} S_\nu R q_{p+n+13}$, $\nu = 0,1, \ldots, n-1, n$

$q_{p+n+13} \tau O q_{p+n+14}$

$q_{p+n+14} O L q_{p+n+14}$

$q_{p+n+14} S_\nu L q_{p+n+14}$, $\nu = 0,1, \ldots, n=1, n, n+1$

$q_{p+n+14} \lambda O q_{p+n+15}$

$q_{p+n+15} O R q_{p+n+18}$ (terminate).

Now, we have

$\mathfrak{L}'_p: q_1 B_1 * B_2 * \cdots * B_p * A_1 * \cdots * A_m$

$\models \lambda * q_3 B_1 * \cdots * B_p * A_1 * \cdots * A_m$

$\models \lambda * B_1 * \cdots * B_p \rho q_{p+4} A_1 * \cdots * A_m$

$\models \lambda * B_1 * \cdots * B_p \rho A_1 * \cdots * A_m q_{p+5} \tau.$

Let now $A_m = S_{i_1} S_{i_2} \cdots S_{i_k}$; the work of $\mathfrak{L}'_p$ continues:

$\vdash \lambda * B_1 * \cdots * B_p \rho A_1 * \cdots * S_{i_1} S_{i_2} \cdots S_{i_{k-1}} q_{p+6} S_{i_k} \tau$

$\vdash \lambda * B_1 * \cdots * B_p \rho A_1 * \cdots * S_{i_1} S_{i_2} \cdots S_{i_{k-1}} q_{p+7+i_k} \lambda_{i_k} \tau$

$\models q_{p+7+i_k} \lambda * B_1 * \cdots * B_p \rho A_1 * \cdots * S_{i_1} S_{i_2} \cdots S_{i_{k-1}} \lambda_{i_k} \tau$

$\vdash q_{p+n+9} S_{i_k} * B_1 * \cdots * S_{i_1} \cdots S_{i_{k-1}} \lambda_{i_k} \tau$

$\vdash q_{p+n+10} O S_{i_k} * B_1 * \cdots * S_{i_1} \cdots S_{i_{k-1}} \lambda_{i_k} \tau$

$\vdash q_{p+n+10} \lambda S_{i_k} * B_1 * \cdots * S_{i_1} \cdots S_{i_{k-1}} \lambda_{i_k} \tau$

$\vdash \lambda q_{p+n+11} S_{i_k} * B_1 * \cdots * S_{i_1} \cdots S_{i_{k-1}} \lambda_{i_k} \tau$

$\models \lambda S_{i_k} * B_1 * \cdots * B_p \rho A_1 * \cdots * S_{i_1} \cdots S_{i_{k-2}} q_{p+6} S_{i_{k-1}} \lambda_{i_k} \tau$

$\models \lambda A_1 * \cdots * A_m * B_1 * \cdots * B_p q_{p+6} \rho \overline{A}_1 \lambda_{n+1} \overline{A}_2 \lambda_{n+1} \cdots \lambda_{n+1} \overline{A}_m \tau$

$\models \lambda A_1 * \cdots * A_m * B_1 * \cdots * B_p * A_1 * \cdots * A_m q_{p+n+13} \tau$

$\models \cdot O q_{p+n+18} \overline{(A_1, \ldots, A_m, B_1, \ldots, B_p, A_1, \ldots, A_m)}.$

Applying now theorem 8. 4 we have proved our theorem for $p > o$. For $p = 0$ we get $\mathfrak{L}'_0$ omitting in $\mathfrak{L}'_p$ all quadruples $(+)$ and the quadruples

$q_2 O * q_2$

$q_2 * L q_{n+16}$

$q_{n+17} * R q_3$

and adding the quadruples

$q_2\, O\, \rho\, q_2$

$q_2\, \rho\, L\, q_{n+16}$

$q_{n+17}\, \rho\, R\, q_4$

and in the same time writing in all other quadruples o for p. Then we have

$\mathfrak{L}'_0 : q_1\, A_1 * \cdots * A_m$

$\models \lambda\, \rho\, q_4\, A_1 * \cdots * A_m$

and other work exactly as before until

$\models \cdot\, O\, q_{n+18}\, A_1 * \cdots * A_m * A_1 * \cdots * A_m$

is reached. Now, apply theorem 8. 4.

Theorem 10. 2. (The transfer algorithms $\mathfrak{R}_p$). For each $m > 0$ and $p > 0$ there exists a $(p+m)$-regular *Turing* algorithm $\mathfrak{R}_p$ such that

$$\mathrm{Res}_{\mathfrak{R}_p}[q_1\, \overline{(B_1, \ldots, B_p, A_1, \ldots, A_m)}] = q_{\Theta(R_p)}\, \overline{(A_1, \ldots, A_m, B_1, \ldots, B_p)}\,.$$

Proof. We construct $\mathfrak{L}'_p$ exactly as in the proof of the theorem 10. 1 for $p > 0$, up to the quadruple $q_{p+6}\, \rho * q_{p+n+12}$, and we add the following quadruples (whose role is to erase the doubled part):

$q_{p+6}\, \rho\, O\, q_{p+n+12}$

$q_{p+n+12}\, O\, R\, q_{p+n+13}$

$q_{p+n+13}\, \lambda_\nu\, O\, q_{p+n+12}, \quad \nu = 0, 1, \ldots, n-1, n, n+1$

$q_{p+n+13}\, \tau\, O\, q_{p+n+14}$

$q_{p+n+14}\, S_\nu\, L\, q_{p+n+14}, \quad \nu = 0, 1, \ldots, n-1, n, n+1$

$q_{p+n+14}\, \lambda\, O\, q_{p+n+15}$

$q_{p+n+15}\, O\, R\, q_{p+n+18}.$

This algorithm we call $\mathfrak{R}'_p$. We have

$$\mathfrak{R}'_p : q_1\, B_1 * \cdots * B_p * A_1 * \cdots * A_m$$

$$\models \lambda\, A_1 * \cdots * A_m * B_1 * \cdots * B_p\, q_{p+6}\, \rho\, \overline{A}_1\, \lambda_{n+1} \cdots \lambda_{n+1}\, \overline{A}_m\, \tau$$

$$\models \lambda\, A_1 * \cdots * A_m * B_1 * \cdots * B_p\, O\, O \cdots O\, q_{p+n+13}\, \tau$$

$$\models \cdot\, O\, q_{p+n+18}\, A_1 * \cdots * A_m * B_1 \cdots * B_p.$$

Applying now theorem 8. 4 we have the proof of theorem 10. 2.

11. Conservation of arguments and composition.

Theorem 11. 1. *For each m-regular Turing algorithm Z, there is an m-regular Turing algorithm Z′ such that, whenever*

$$\mathrm{Res}_{Z:\mathfrak{A}}[q_1\, \overline{(A_1, \ldots, A_m)}] = q_{\Theta(Z)}\, \overline{(B_1, \ldots, B_s)}$$

it is also the case that

$$\mathrm{Res}_{Z':\mathfrak{A}}[q_1,\overline{(A_1, \ldots, A_m)}] = q_{\Theta(Z')}\overline{(B_1, \ldots, B_s, A_1, \ldots, A_m)}$$

whereas, whenever $\mathrm{Res}_{Z:\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}]$ *is undefined so is*

$$\mathrm{Res}_{Z':\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}].$$

Proof. Having now at our disposal all the algorithms which are employed by *Davis* in the proof of Lemma 3 of § 1, Ch. 2 of [2] we refer to the proof of this lemma. which can be here repeated literally.

The similar is for the proof of

Theorem 11. 2. *For each m-regular Turing algorithm Z, there is an m-regular Turing algorithm Z' such that, whenever*

$$\mathrm{Res}_{Z:\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}] = q_{\Theta(Z)}\overline{(B_1, \ldots, B_s)}$$

it is also the case that

$$\mathrm{Res}_{Z':\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}] = q_{\Theta(Z')}\overline{(A_1, \ldots, A_m, B_1, \ldots, B_s)}$$

whereas whenever $\mathrm{Res}_{Z:\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}]$ *is undefined so is also*

$$\mathrm{Res}_{Z':\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}].$$

Theorem 11. 3. (Composition). *Let* $Z_1, Z_2, \ldots, Z_p$ *be Turing algorithms and* $m > 0$. *Then, there exists an m-regular Turing algorithm Z' such that*

$$\mathrm{Res}_{Z':\mathfrak{A}}[q_1\overline{(A_1, \ldots, A_m)}]$$

$$= q_{\Theta(Z')}\overline{(\Psi_{Z_1:\mathfrak{A}}(A_1, \ldots, A_m), \Psi_{Z_2:\mathfrak{A}}(A_1, \ldots, A_m), \ldots, \Psi_{Z_p:\mathfrak{A}}(A_1, \ldots, A_m))}.$$

Proof. Induction on *p*, as the proof of Lemma 4, § 1, Ch. 2 of [2]. ıll needed algorithms (i.e. machines) are at our disposal now).

A straightforward corollary of the foregoing theorem is

Theorem 11. 4. *Let* $f(X_1, X_2, \ldots, X_m)$ *and all* $g_i(X_1, X_2, \ldots, X_p)$, $= 1, 2, \ldots, m$ *be (partially)* $\mathfrak{A}$*-algorithmic. Then, the function*

$$(X_1, \ldots, X_p) \simeq f(g_1(X_1, \ldots, X_p), g_2(X_1, \ldots, X_p), \ldots, g_m(X_1, \ldots, X_p))$$

also (partially) $\mathfrak{A}$*-algorithmic.*

Hence: the class of (partially) $\mathfrak{A}$-algorithmic functions is closed under ıe operation of composition.

12. Minimalisation. We recall that we call *numerals* all words written ith the only letter S_0. Obviously, the numerals correspond 1—1 to the natu- ıl numbers; therefore we write 1 for S_0, 2 for $S_0 S_0$, 3 for $S_0 S_0 S_0$ and so n. The empty word *O* corresponds to zero. Hence, numerals are words f $\Omega(\mathfrak{S}_0)$, where $\mathfrak{S}_0 = \{S_0\}$.

When we wish to denote that a word variable runs only over numerals (i. e. over the set $\Omega(\mathfrak{S}_0)$, we shall write it in italics: $x, y, z, \ldots$

The meaning of „the least numeral such that..." corresponds to „the least natural number such that...".

Definition 12. 1. *The word function* $f(y, X_1, \ldots, X_m)$ *is restrictively in* y *total if it is defined for all m-tuples of words* $X_1, X_2, \ldots, X_m$ *of* $\Omega(\mathfrak{S})$ *and for every numeral* $y \in \Omega(\mathfrak{S}_0)$.

Definition 12. 2. *The operation of minimalization associates with each funcion* $f(y, X_1, \ldots, X_m)$, *which is restrictively in* y *total, the function* $h(X_1, \ldots, X_m)$ *whose value, for given* $X_1, \ldots, X_m$ *is the least numeral* y, *if such exists, for which* $f(y, X_1, \ldots, X_m) = O$, *and which is undefined if no such* y *exists. We write*

$$h(X_1, \ldots, X_m) \simeq min_y [f(y, X_1, \ldots, X_m) = O].$$

Naturally, if $h(A_1, \ldots, A_m)$ exists, it is always a numeral. The function h (partially) maps the set $\underbrace{\Omega(\mathfrak{S}) \times \cdots \times \Omega(\mathfrak{S})}_{m}$ into the set $\Omega(\mathfrak{S}_0)$.

Definition 12. 3. *The function* $f(y, X_1, \ldots, X_m)$ *which is restrictively in* y *total, is called regular if*

$$h(X_1, \ldots, X_m) \simeq min_y [f(y, X_1, \ldots, X_m) = O]$$

is total.

Definition 12. 4. $f(y, X_1 \ldots, X_m)$ *is restrictively in* y $\mathfrak{A}$*-algorithmic if there is a Turing algorithm* Z *such that*

$$\Psi_{Z:\mathfrak{A}}(y, R_1, \ldots, X_m) = f(y, X_1, \ldots, X_m)$$

and $f(y, X_1, \ldots, X_m)$ *is restrictively in* y *total.*

Theorem 12. 1. *If* $f(y, X_1, \ldots, X_m)$ *is restrictively in* y $\mathfrak{A}$*-algorithmic, then*

$$h(X_1, \ldots, X_m) \simeq min_y [f(y, X_1, \ldots X_m) = O]$$

is partially $\mathfrak{A}$*-algorithmic. Moreover, if* $f(y, X_1, \ldots, X_m)$ *is regular,* $h(X_1, \ldots, X_m)$ *is* $\mathfrak{A}$*-algorithmic.*

Proof. The proof is a very easy version of the proof of Th. 2. 4, Ch. 2 of [2]. Only some minimal technical changes are necessary.

So, the class of partially $\mathfrak{A}$-algorithmic functions is closed under minimalization over a numerical variable of functions which are restrictively in that variable $\mathfrak{A}$-algorithmic. The class of $\mathfrak{A}$-algorithmic functions is closed under minimalization over a numerical variable of functions which are restrictively in that variable $\mathfrak{A}$-algorithmic and regular.

13. Primitive recursion. In this section we shall prove that the operation of primitive recursion over $\mathfrak{A}$-algorithmic functions generates also $\mathfrak{A}$-algorithmic functions. With this it will be proved that all ($\mathfrak{A}$—) primitive recursive word functions are ($\mathfrak{A}$—) algorithmic. (For the definitions of primitive recursion and $\mathfrak{A}$-primitive recursive function we refer to our papers [3] and [1]).

Theorem 13. 1. *Let* $a(X_1, \ldots, X_m)$ *and all* $b_i(Y, X_1, \ldots, X_m, Z)$ *be* $\mathfrak{A}$*-algorithmic for* $i = 0, 1, \ldots, n-1$. *Then the function* $f(Y, X_1, \ldots, X_m)$, *defined by*

$$f(O, X_1, \ldots, X_m) = a(X_1, \ldots, X_m),$$

$$f(S_i Y, X_1, \ldots, X_m) = b_i(Y, X_1, \ldots, X_m, f(Y, X_1, \ldots, X_m)),$$

$$i = 0, 1, \ldots, n-1$$

is also $\mathfrak{A}$*-algorithmic.*

Proof. Let the *Turing* algorithms Z_a, Z_i, $i=0, 1, \dots, n-1$, compute the functions a and b_i respectively ($i=0, 1, \dots, n-1$). The idea of the proof is the following one:

we construct first an algorithm $\mathfrak{D}$ which will, beginning with the given $(m+1)$-tuple $S_i Y * X_1 * \cdots * X_m$, write all $(m+1)$-tuples

$$Y * X_1 * \cdots * X_m, \quad v(Y) * X_1 * \cdots * X_m, \quad \dots,$$

up to $O * X_1 * \cdots * X_m$; then, we shall let work first Z_a on the last m-tuple, and then all Z_i needed on successive $(m+2)$-tuples

$$Z * X_1 * \cdots * X_m * f(Z, X_1, \dots, X_m)$$

as to come to the first $(m+2)$-tuple

$$Y * X_1 * \cdots * X_m * f(Y, X_1, \dots, X_m).$$

Let S_G be the first letter of the letters $S_{n+2}, S_{n+3}, \dots$ which is not in the alphabet of any of the algorithms $Z_a, Z_0, Z_1, \dots, Z_{n-1}$. Beginning with $\tau_0 = S_G$ we introduce the letters $\tau_1, \tau_2, \dots, \tau_{n-1}, \tau, \lambda, \rho, \sigma, \eta,$

$$\lambda_0, \dots, \lambda_{n-1}, \lambda_n, \lambda_{n+1},$$

to be the letters which follow τ_0 successively.

We suppose that the printing alphabet of $Z_a, Z_0, \dots, Z_{n-1}$ is

$$\mathfrak{S} = \{S_0, S_1, \dots, S_{n-1}\};$$

O is S_n and $*$ is S_{n+1} in all of them.

Let $\mathfrak{D}'$ consist of quadruples:

$q_1 S_\nu \tau_\nu q_2, \quad \nu = 0, 1, \dots, n-1$

$q_2 \tau_\nu R q_2, \quad \nu = 0, 1, \dots, n-1$

$q_2 S_\nu R q_2, \quad \nu = 0, 1, \dots, n-1, n$

$q_2 * R q_3$

$q_{2+i} S_\nu R q_{2+i}, \quad \nu = 0, 1, \dots, n-1, n, \quad i = 1, 2, \dots, m-1$

$q_{2+i} * R q_{3+i}, \quad i = 1, 2, \dots, m-1$

$q_{m+2} S_\nu R q_{m+3}, \quad \nu = 0, 1, \dots, n-1, n$

$q_{m+3} S_\nu S_\nu q_{m+2}, \quad \nu = 0, 1, \dots, n-1$

$q_{m+3} O \tau q_{m+3}$

$q_{m+3} \tau R q_{m+4}$

$q_{m+4} O \lambda q_{m+4}$

$q_{m+4} \lambda L q_{m+5}$

$q_{m+5} \tau L q_{m+5}$

$q_{m+5} S_\nu L q_{m+5} \quad \nu = 0, 1, \dots, n-1, n, n+1$

$q_{m+5} \tau_i R q_{m+6} \quad i = 0, 1, \dots, n-1$

$q_{m+6} * R q_{m+7}$

$q_{m+7}\, S_\nu\, R\, q_{m+7}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+7}\, \tau\, R\, q_{m+7}$

$q_{m+7}\, \lambda\, O\, q_{m+8}$

$q_{m+8}\, O\, R\, q_{m+9}$

$q_{m+9}\, O\, \lambda\, q_{m+10}$

$q_{m+10}\, \lambda\, L\, q_{m+11}$

$q_{m+11}\, O\, L\, q_{m+12}$

$q_{m+12}\, \tau\, O\, q_{m+13}$

$q_{m+13}\, O\, R\, q_{m+14}$

$q_{m+14}\, O\, \tau\, q_{m+15}$

$q_{m+15}\, \tau\, L\, q_{m+15}$

$q_{m+15}\, O\, L\, q_{m+16}$

$q_{m+16}\, S_\nu\, O\, q_{m+\nu+17}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+\nu+17}\, O\, R\, q_{m+n+\nu+19}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+\nu+n+19}\, O\, S_\nu\, q_{m+15}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+15}\, S_\nu\, L\, q_{m+15}$ $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+15}\, \tau_i\, R\, q_{m+2n+21}$, $i = 0, 1, \ldots, n-1$ (all letters, with exception of τ_i's are placed one square to right)

$q_{m+6}\, S_\nu\, S_\nu\, q_{m+2n+21}$, $\nu = 0, 1, \ldots, n-1$

$q_{m+2n+21}\, S_\nu\, \lambda_\nu\, q_{m+2n+\nu+22}$, $\nu = 0, 1, \ldots, n-1, n, n+1$ (begin the transport)

$q_{m+2n+\nu+22}\, \lambda_\nu\, R\, q_{m+2n+\nu+22}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+2n+\nu+22}\, S_i\, R\, q_{m+2n+\nu+22}$, $i, \nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+2n+\nu+22}\, \tau\, R\, q_{m+2n+\nu+22}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+2n+\nu+22}\, \lambda\, S_\nu\, q_{m+3n+24}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+3n+24}\, S_\nu\, R\, q_{m+3n+25}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+3n+25}\, O\, \lambda\, q_{m+3n+25}$

$q_{n+3n+25}\, \lambda\, L\, q_{m+3n+26}$

$q_{m+3n+26}\, S_\nu\, L\, q_{m+3n+26}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+3n+26}\, \tau\, L\, q_{m+3n+26}$

$q_{m+3n+26}\, \lambda_\nu\, R\, q_{m+2n+21}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+2n+21}\, \tau\, L\, q_{m+3n+27}$ (finish the transpont)

$q_{m+3n+27}\, \lambda_\nu\, S_\nu\, q_{m+3n+28}$, $\nu = 0, 1, \ldots, n-1, n, n+1$ (translate)

$q_{m+3n+28}\, S_\nu\, L\, q_{m+3n+27}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+3n+27}\, \tau_i\, R\, q_{m+3n+29}$, $i = 0, 1, \ldots, n-1$

$q_{m+3n+29} S_\nu R q_{m+3n+29}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+3n+29} \tau R q_{m+3n+30}$

$q_{m+3n+30} S_\nu R q_{m+3n+30}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+3n+30} \lambda O q_{m+3n+31}$ (erase λ)

$q_{m+3n+31} S_\nu L q_{m+3n+31}$, $\nu = 0, 1, \ldots, n-1, n, n+1$

$q_{m+3n+31} \tau R q_{m+3n+32}$.

Let $Y \neq O$. We have

$\mathfrak{D}' : q_1 S_i Y * X_1 * \cdots * X_m$

$\vdash q_2 \tau_i Y * X_1 * \cdots * X_m$

$\models \tau_i Y * X_1 * \cdots * q_{m+2} X_m$

$\models \tau_i Y * X_1 * \cdots * X_m \tau q_{m+4} \lambda$

$\models \tau_i q_{m+6} Y * X_1 * \cdots * X_m \tau \lambda$

$\vdash \tau_i q_{m+2n+21} Y * X_1 * \cdots * X_m \tau \lambda$;

If, by a bar $(\overline{X})$, we denote the translation into λ_i' s-alphabet, we have further

$\models \tau_i \overline{Y} \lambda_{n+1} \overline{X_1} \lambda_{n+1} * \cdots * \lambda_{n+1} \overline{X_m} q_{m+2n+21} \tau\; Y * X_1 * \cdots * X_m \lambda$

$\models q_{m+3n+29}\; Y * X_1 * \cdots * X_m \tau\; Y * X_1 * \cdots * X_m \lambda$

$\models \cdot \tau_i\; Y * X_1 * \cdots * X_m \tau q_{m+3n+32}\; Y * X_1 * \cdots * X_m$.

If $Y = O$, i. e. $S_i Y = S_i O = S_i$ we have

$\mathfrak{D}' : q_1 S_i * X_1 * \cdots * X_m$

$\models \cdot \tau_i O * X_1 * \cdots X_m \tau q_{m+3n+32} O * \cdots * X_m$.

So, be Y empty or not, we have always

$$\operatorname{Res}_{\mathfrak{D}'} [q_1 S_i Y * X_1 * \cdots * X_m] =$$
$$\tau_i Y * X_1 * \cdots * X_m \tau q_{m+3n+32} Y * X_1 * \cdots * X_m.$$

Take into account that $\mathfrak{D}'$ is operating always „to right", moving the tape expression not any square to the left.

Let now $\mathfrak{D}''$ consist of the quadruples

$$q_{m+3n+32} S_\nu S_\nu q_1, \quad \nu = 0, 1, \ldots, n-1,$$

and let $\mathfrak{D} = \mathfrak{D}' \cup \mathfrak{D}''$. What will be the effect of $\mathfrak{D}$ on $q_1 S_i Y * X_1 * \cdots * X_m$? Obviously it will repeat the effect of $\mathfrak{D}'$ on that part of the instantaneous description which beginns by $q_{m+3n+32}$ until we get one „last part" of the form $q_{m+3n+32} O * X_1 * \cdots * X_m$, for which there is no more computation.

So, if $Y = S_{i_1} S_{i_2}, \ldots, S_{i_k}$ we will have:

$$\mathfrak{D} : q_1 S_i Y * X_1 * \cdots * X_m$$

$$\models \cdot \tau_i Y * \overline{(X_1, \ldots, X_m)} \tau \tau_{i_1} S_{i_2} \ldots .\; S_{i_k} * \overline{(X_1, \ldots, X_m)} \tau \ldots .$$

$$\tau \tau_{i_k} O * \overline{(X_1, \ldots, X_m)} \tau q_{m+3n+32} O * X_1 * X_2 * \cdots * X_m,$$

which is final.

It is easy to show that we can concieve the algorithm Z_a as working not on m-tuples $\overline{(X_1, \ldots, X_m)}$, but on $(m+1)$-tuples $\overline{(O, X_1, \ldots, X_m)}$. (For this, we have only to introduce the algorithm $\mathfrak{J}$ which consists of quadruples $q_1 O R q_1$, $q_1 * O q_2$, $q_2 O R q_3$ and to work with $\mathfrak{J} \cup Z_a^{(2)}$).

So, we shall regard the algorithm Z_a as working on $(m+1)$-tuples $\overline{(O, X_1, \ldots, X_m)}$ computing $a(X_1, \ldots, X_m)$. Other algorithms

$$Z_i, \; i=0, 1, \ldots, n-1$$

act onto $(m+2)$-tuples $\overline{(Y, X_1, \ldots, X_m, Z)}$ computing $b_i(Y, X_1, \ldots, X_m, Z)$.

Now, we have to include all these algorithms; but we have to take into account the necessity to move left if any of their internal configurations meets τ on the left. Also, after the computation of any of these algorithms we have to change this τ into $*$ and (erasing the τ_i which was after τ) to put the result left, close to this $*$.

Let $N = m + 3n + 32$ (So $Z_a^{(N-1)}$ will work on $\tau q_N O * X_1 * \cdots * X_m$ and, if undisturbed from left, finish with $\tau q_{\Theta(Z_a^{(N-1)})} a(X_1, \ldots, X_m)$. But, using th. 8. 1. we shall suppose Z_a (and only Z_a) to work so that it finishes with $O q_{\Theta(Z_a)} a(X_1, \ldots, X_m)$. So $Z_a^{(N-1)}$ will finish with

$$\tau O q_{\Theta(Z_a^{(N-1)})} a(X_1, \ldots, X_m)).$$

Let now

$$N_0 = \Theta(Z_a^{(N-1)}),$$
$$N_1 = \Theta(Z_0^{(N_0)}),$$
$$N_i = \Theta(Z_i^{(N_{i-1})}), \qquad i = 2, 3, \ldots, n-1,$$

and let

$$Z' = Z_a^{(N-1)} \cup Z_0^{(N_0)} \cup Z_1^{(N_1)} \cup, \ldots, \cup Z_{n-1}^{(N_{n-1})}.$$

We have to take into account that $Z_a^{(N-1)}$ has q_{N0} as the internal configuration with maximal index, and that $Z_0^{(N0)}$ has q_{N0+1} as the internal configuration with lowest index. Similar is the situation with other algorithms; so, the internal configurations

$$q_{N0}, q_{N1}, q_{N2}, \ldots, q_{Nn-1}$$

do not occur in Z'. Let now $T = \Theta(Z')$. We introduce the algorithm $\mathfrak{T}$, whose role is to transport the part after q_{Ni} one square left, to transform τ before it into $*$ and to seek for first τ_i on the left.

$\mathfrak{T}$ will consist of following quadruples:

$q_{Ni} S_\nu S_\nu q_{T+1}$, $\quad i = 0, 1, \ldots, n-1,; \nu = 0, 1, \ldots, n-1, n$

$q_{T+1} S_\nu R q_{T+1}$, $\quad \nu = 0, 1, \ldots, n-1$

$q_{T+1} O \sigma q_{T+2}$

$q_{T+2} \sigma L q_{T+2}$

$q_{T+2} S_\nu L \; q_{T+3}$, $\quad \nu = 0, 1, \ldots, n-1$

$q_{T+3} S_\nu S_\nu q_{T+2}$, $\quad \nu = 0, 1, \ldots, n-1$

$q_{T+2} O L q_{T+4}$

$q_{T+4}\tau R q_{T+5}$ (if τ does not exist, terminate)

$q_{T+5}ORq_{T+6}$

$q_{T+6}S_\nu O q_{T+\nu+7}$, $\nu=0, 1, \dots, n-1$

$q_{T+\nu+7}OLq_{T+n+\nu+7}$, $\nu=0, \dots, n-1$

$q_{T+n+\nu+7}OS_\nu q_{T+2n+7}$, $\nu=0, 1, \dots, n-1$

$q_{T+2n+7}S_\nu R q_{T+2n+8}$, $\nu=0, 1, \dots, n-1$

$q_{T+2n+8}ORq_{T+6}$

$q_{T+6}\sigma O q_{T+2n+9}$

$q_{T+2n+9}S_\nu L q_{T+2n+9}$, $\nu=0, 1, \dots, n-1, n$

$q_{T+2n+9}\tau * q_{T+2n+10}$

$q_{T+2n+10}S_\nu L q_{T+2n+10}$, $\nu=0, 1, \dots, n-1, n, n+1$

$q_{T+2n+10}\tau_i O q_{T+2n+i+11}$, $i=0, 1, \dots, n-1$

$q_{T+2n+i+11}ORq_{N_i+1}$, $i=0, 1, \dots, n-1.$

We demonstrate the work of $\mathfrak{T}$:

$\mathfrak{T}: P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m\tau O q_{N_s}\, S_{j_1}S_{j_2}.\ \dots\ .\ S_{j_l}$

$\vdash P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m\tau O q_{T+1}\, S_{j_1}S_{j_2}.\ \dots\ .\ S_{j_l}$

$\models P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m\tau O S_{j_1}.\ \dots\ .\ S_{j_l}q_{T+2}\sigma$

$\models P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m q_{T+4}\tau O S_{j_1}S_{j_2}.\ \dots\ .\ S_{j_l}\sigma$

$\models P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m\tau O q_{T+6}\, S_{j_1}S_{j_2}.\ \dots\ .\ S_{j_l}\sigma$

$\vdash P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m\tau O q_{T+j_1+7}\, OS_{j_2}.\ \dots\ .\ S_{j_l}\sigma$

$\vdash P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m\tau q_{T+n+j_1+7}\, OOS_{j_2}.\ \dots\ .\ S_{j_l}\sigma$

$\vdash P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m\tau q_{T+2n+7}S_{j_1}OS_{j_2}.\ \dots\ .\ S_{j_l}\sigma$

$\models P\tau\tau_{i_\nu}\, Y*X_1*\cdots*X_m q_{T+2n+9}\tau S_{j_1}S_{j_2}.\ \dots\ .\ S_{j_l}$

$\models\cdot P\tau O q_{Ni_\nu+1}Y*X_1*\cdots*X_m*S_{j_1}S_{j_2}.\ \dots\ .\ S_{j_l}.$

Obviously, the effect of $\mathfrak{T}$ is to arrange the rightmost part of the tape expression for the computation with the due $Z_i^{(Ni)}$.

Therefore $Z''=Z'\cup\mathfrak{T}$ will compute for every algorithm $Z_i^{(Ni)}$ as its τ_i appears. We have still to take into accouut the eventual disturbing rôle of the left part.

Let $M=\Theta(Z'')$ and let $\mathfrak{L}$ be the algorithm: (In these quadruples q_i runs over all internal configurations of Z'; we write still once q_i as $q(i)$):

$q_i\tau\eta q(2^{M+1}\cdot 3^i)$

$q(2^{M+1}\cdot 3^i)\eta L q(2^{M+2}\cdot 3^i)$

$q(2^{M+2}\cdot 3^i)S_\nu L q(2^{M+2}\cdot 3^i)$, $\nu=0, 1, \dots, n-1, n, n+1$

$q(2^{M+2}\cdot 3^i)\tau_\nu L q(2^{M+3}\cdot 3^i)$, $\nu=0, 1, \dots, n-1$

$q\,(2^{M+3}\cdot 3^i)\,\tau L q\,(2^{M+2}\cdot 3^i)$

$q\,(2^{M+3}\cdot 3^i)\,O R q\,(2^{M+4}\cdot 3^i)$

$q\,(2^{M+4}\cdot 3^i)\,\tau_\nu O q\,(2^{M+4}\cdot 3^i\cdot 5^{\nu+1}),\quad \nu=0,1,\ldots,n-1$

$q\,(2^{M+4}\cdot 3^i\cdot 5^{\nu+1})\,O L q\,(2^{M+5}\cdot 3^i\cdot 5^{\nu+1}),\quad \nu=0,1,\ldots,n-1$

$q\,(2^{M+5}\cdot 3^i\cdot 5^{\nu+1})\,O\tau_\nu q\,(2^{M+5}\cdot 3^i\cdot 5^{n+1}),\quad \nu=0,1,\ldots,n-1$

$q\,(2^{M+5}\cdot 3^i\cdot 5^{n+1})\,\tau_\nu R q\,(2^{M+3}\cdot 3^i),\quad \nu=0,1,\ldots,n-1$

$q\,(2^{M+4}\cdot 3^i)\,S_\nu O q\,(2^{M+4}\cdot 3^i\cdot 7^{\nu+1}),\quad \nu=0,1,\ldots,n-1,n,n+1$

$q\,(2^{M+4}\cdot 3^i\cdot 7^{\nu+1})\,O L q\,(2^{M+5}\cdot 3^i\cdot 7^{\nu+1}),\quad \nu=0,1,\ldots,n-1,n,n+1$

$q\,(2^{M+5}\cdot 3^i\cdot 7^{\nu+1})\,O S_\nu q\,(2^{M+5}\cdot 3^i\cdot 7^{n+3}),\quad \nu=0,1,\ldots n-1,n,n+1$

$q\,(2^{M+5}\cdot 3^i\cdot 7^{n+3})\,S_\nu R q\,(2^{M+3}\cdot 3^i),\quad \nu=0,1,\ldots,n-1,n,n+1$

$q\,(2^{M+4}\cdot 3^i)\,\eta\, O q\,(2^{M+6}\cdot 3^i)$

$q\,(2^{M+6}\cdot 3^i)\,O L q\,(2^{M+7}\cdot 3^i)$

$q\,(2^{M+7}\cdot 3^i)\,O\tau q\,(2^{M+8}\cdot 3^i)$

$q\,(2^{M+8}\cdot 3^i)\,\tau R q_i.$

The effect of $\mathfrak{L}$ is to eliminate the role of the τ and the part on the left of it in the work of Z'.

Let now $Z'''=Z''\cup\mathfrak{L}$. We have

$$Z'':P\tau O q_{N_{i_\nu}+1}\,Y*X_1*,\ \ldots\ ,*X_m*Z$$

$$\models P\tau O q_{N_{i_{\nu+1}}}\,\Psi_{Z_{i_\nu}:\mathfrak{A}}\ \overline{(Y,X_1,\ \ldots\ ,X_m,Z)},$$

and now beginns the work of $\mathfrak{T}$.

We state that the algorithm

$$Z=\mathfrak{D}\cup Z'\cup\mathfrak{T}\cup\mathfrak{L}\cup\{q_1\,O\,O\,q_N\}$$

is such that

(1) $$\Psi_{Z:\mathfrak{A}}(Y,X_1,\ \ldots\ ,X_m)=f\,(Y,X_1,\ \ldots\ ,X_m)$$

We prove first two lemmas.

L e m m a 13. 1. Let P be any expression. Then

$$Z:P\tau q_1\,Y*X_1*,\ \ldots\ ,*X_m$$
$$\models_{\mathfrak{A}} P q_{T+2n+10}*f\,(Y,X_1,\ \ldots\ ,X_m).$$

Proof. (Note that the instantaneous description after $\models_{\mathfrak{A}}$ is not meant to be final!)

For $Y=O$ we have:

$$Z: P\tau q_1 O * X_1 * X_2 * \cdots * X_m$$

$$\vdash P\tau q_N O * X_1 * X_2 * \cdots * X_m$$

$$\underset{\mathfrak{A}}{\models} P\tau Q q_{N_0} a(X_1, \ldots, X_m)$$

$$\models P\tau O q_{T+1} a(X_1, \ldots, X_m)$$

$$\vdash P\tau O a(X_1, \ldots, X_m) q_{T+2}\sigma$$

$$\models P q_{T+4}\tau O a(X_1, \ldots, X_m)\sigma$$

$$\models P\tau O q_{T+5} a(X_1, \ldots, X_m)\sigma$$

$$\models P q_{T+2n+9}\tau a(X_1, \ldots, X_m)$$

$$\vdash P q_{T+2n+10} * f(O, X_1, \ldots, X_m).$$

Let now the statement of the lemma be valid for Y. Then, for every $i=0, 1, \ldots, n-1$ we have

$$Z: P\tau q_1 S_i Y * X_1 * \cdots * X_m$$

$$\models P\tau\tau_i Y * X_1 * \cdots * X_m \tau q_1 Y * X_1 * \cdots * X_m$$

and, by induction hypothesis,

$$\underset{\mathfrak{A}}{\models} P\tau\tau_i Y * X_1 * \cdots * X_m q_{T+2n+10} * f(Y, X_1, \ldots, X_m)$$

$$\models P\tau O q_{N_i+1} Y * X_1 * \cdots * X_m * f(Y, X_1, \ldots, X_m)$$

$$\underset{\mathfrak{A}}{\models} P q_{T+2n+10} * \Psi_{Z_i:\mathfrak{A}}(Y, X_1, \ldots, X_m, f(Y, X_1, \ldots, X_m))$$

$$= P q_{T+2n+10} * b_i(Y, X_1, \ldots, X_m, f(Y, X_1, \ldots, X_m))$$

$$= P q_{T+2n+10} * f(S_i Y, X_1, \ldots, X_m)$$

and this proves the lemma.

Remark. In the proof of the foregoing lemma we did employ the induction axiom for the set $\Omega(\mathfrak{S})$ in the following form: If $f(O)$ is true and if from the truth of $f(X)$ follows the truth of all $f(S_i X)$, for $i=0, 1, 2, \ldots, n-1$, then $f(X)$ is true for every word $X \in \Omega(\mathfrak{S})$. We call this form of the induction-axiom the axiom of the stage induction. In [3] we have shown that this axiom can be proved if the uniqueness of the recursive definition is assumed.

We prove now

Lemma 13. 2. For $i=0, 1, \ldots, n-1$

$$Z: q_1 S_i Y * X_1 * \cdots * X_m$$

$$\underset{\mathfrak{A}}{\models} \cdot q_{T+4} OO f(S_i Y, X_1, \ldots, X_m)$$

Proof. (Note that the instantaneous description after $\underset{\mathfrak{A}}{\models}\cdot$ is now final). We have

$$Z\colon q_1 S_i Y * X_1 * \cdots * X_m$$

$$\models \tau_i Y * X_1 * \cdots * X_m \tau q_1 Y * X_1 * \cdots * X_m$$

$$\underset{\mathfrak{A}}{\models} \tau_i Y * X_1 * \cdots * X_m q_{T+2n+10} * f(Y, X_1, \ldots, X_m) \text{—(by Lemma 13. 1)}$$

$$\models O q_{N_{i+1}} Y * X_1 * \cdots * X_m * f(Y, X_1, \ldots, X_m)$$

$$\underset{\mathfrak{A}}{\models} O q_{N_{i+1}} b_i (Y, X_1, \ldots, X_m, f(Y, X_1, \ldots, X_m))$$

$$\vdash O q_{T+1} f(S_i Y, X_1, \ldots, X_m)$$

$$\models q_{T+4} O O f(S_i Y, X_1, \ldots, X_m).$$

This proves the lemma.

Now, by direct computation, we have

$$\Psi_{Z:\mathfrak{A}} (O, X_1, \ldots, X_m) = f(O, X_1, \ldots, X_m)$$

and by lemma 13. 2 we have

$$\Psi_{Z:\mathfrak{A}} (S_i Y, X_1, \ldots, X_m) = f(S_i Y, X_1, \ldots, X_m), \qquad i = 0, 1, \ldots, n-1.$$

By the axiom of stage induction follows (1), and the theorem 13. 1. is proved.

In [1], def. 6. 1, we defined the $\mathfrak{A}$-primitive recursive functions of words. As we have shown that all the functions appearing there are $\mathfrak{A}$-algorithmic and that the operations of composition and of primitive recursion are $\mathfrak{A}$-algorithmic also, we have

Theorem 13. 2. *Every ($\mathfrak{A}$-) primitive recursive function is ($\mathfrak{A}$-) algorithmic.*

(In [1] this theorem was tacitely assumed).

14. Relation to Markov's normal algorithms. In this section we prove that for every normal algorithm there is an equivalent *Turing* algorithm. To shorten the proof we shall employ some results of [3] and the theorems of the foregoing sections.

As always we regard an alphabet $\mathfrak{S} = \{S_0, S_1, \ldots, S_{n-1}\}$, over which is given some *Markov*'s normal algorithm $\mathfrak{M}$

$$(14.\ 1) \qquad P_i \to (\cdot)\, Q_i, \qquad i = 1, 2, \ldots, r,$$

where P_i and Q_i are the words of the alphabet

$$\mathfrak{S}' = \{S_0, S_1, \ldots, S_{n-1}, S_{n+1}, S_{n+2}, \ldots, S_{n+k}\}$$

of *Markov's* algorithm $\mathfrak{M}$ (We excluded the letter S_n, as it will represent the empty square in the corresponding *Turing* algorithm, — and the empty word in *Markov*'s algorithm $\mathfrak{M}$). We take into account that the *Markov* algorithm $\mathfrak{M}$ mapps the words of $\Omega(\mathfrak{S})$ into the words of the same set. $S_{n+1}, S_{n+2}, \ldots, S_{n+k}$ are also auxiliar letters in $\mathfrak{M}$.

We now regard in the alphabet $\mathfrak{S}'$ the function $\sigma(P_i, Q_i, X)$, whose value is equal X if X does not contain P_i, and whose value is equal to the word obtained by substitution of the first appearance of the word P_i in X by the word Q_i — if X contains P_i. As easily seen from [3]

$$\sigma(P_i, Q_i, X)=\left\{\left[X\sim\left(\underset{Z=O}{\overset{s_0(X)\sim s_0(P_i)}{L}}\left\{\prod_{\mu=O}^{Z}\alpha\,[P_i\dot{-}(X\sim\mu)]=O\right\}+P_i\right)\right]+Q_i\right\}$$

$$+R\left\{R(X)\sim\left[X\sim\underset{Z=O}{\overset{s_0(X)\sim s_0(P_i)}{L}}\left(\prod_{\mu=O}^{Z}\alpha\,[P_i\dot{-}(X\sim\mu)]=O\right]\right\}.$$

So $\sigma(P_i, Q_i, X)$ is a primitive recursive word-function in $\Omega(\mathfrak{S}')$. (The same was also shown in [5]). Therefore there exists a *Turing* algorithm Z_i, with the printing alphabet $\mathfrak{S}'$, such that

$$\Psi_{Z_i}(X)=\sigma(P_i, Q_i, X) \quad (i=1, 2, \dots, \tau).$$

By theorems of section 8. there exists a *Turing* algorithm $\overline{Z_i}$, with the same printing alphabet, such that

$$(14.\ 2) \qquad \mathrm{Res}_{\overline{Z}_i}\, q_1 X = q_{\Theta(\bar{Z}_i)}\, \sigma(P_i, Q_i, X), \qquad i=1, 2, \dots, r.$$

Now we shall construct algorithms $\mathfrak{P}_i$ by which we shall examine if the words on the tape contain P_i or not. This is necessary as to take into account the terminating and the non-terminating rules of $\mathfrak{M}$.

Let $P_i=S_{i_1}S_{i_2}, \dots, S_{i_k}$, where $S_{i_\nu}\in\mathfrak{S}'$ for $\nu=1, 2, \dots, k$. $\mathfrak{P}_i$ will consist of the quadruples

$q_1 S_\nu R q_1$ for all $S_\nu\neq S_{i_1}$ and $S_\nu\in\mathfrak{S}'$

$q_\nu S_{i_\nu} R q_{\nu+1}$, $\nu=1, 2, \dots, k$

$q_\nu S_\tau\, S_\tau\, q_1$ for $\tau\neq i_\nu, (\tau=0, 1, \dots, n-1, n+1, \dots, n+k, \nu=1, 2, \dots, k)$

$q_{k+1} S_\nu S_\nu q_{k+2}$ $S_\nu\in\mathfrak{S}'$ or $S_\nu=O$

$q_1\, OO\, q_{k+3}$

$q_{k+2} S_\nu L q_{k+4}$ $S_\nu\in\mathfrak{S}'$ or $S_\nu=O$

$q_{k+4} S_\nu L q_{k+4}$ $S_\nu\in\mathfrak{S}'$

$q_{k+4}\, O R q_{k+8}$ (if X contains P_i finish with $Oq_{k+8}X$)

$q_{k+3}\, O L q_{k+6}$

$q_{k+6} S_\nu L q_{k+6}$, $S_\nu\in\mathfrak{S}'$

$q_{k+6}\, O R q_{k+7}$. (If X does not contain P_i finish with $Oq_{k+7}X$)

Let $X=S_a S_b S_c S_d S_{i_1} S_{i_2} S_e$, where all a, b, c, d, e are different from $i_1, i_2, \dots, i_k$; so X does not contain P_i. We have

$\mathfrak{P}_i: q_1 X \vdash S_a q_1 S_b S_c S_d S_{i_1} S_{i_2} S_e$

$\models S_a S_b S_c S_d q_1 S_{i_1} S_{i_2} S_e$

$\vdash S_a S_b S_c S_d S_{i_1} q_2 S_{i_2} S_e$

$\vdash S_a S_b S_c S_d S_{i_1} S_{i_2} q_3 S_e$

$\vdash S_a S_b S_c S_d S_{i_1} S_{i_2} q_1 S_e$

$\vdash S_a S_b S_c S_d S_{i_1} S_{i_2} S_e q_1 O$

$\vdash S_a S_b S_c S_d S_{i_1} S_{i_2} S_e q_{k+3} O$

$\models \cdot O q_{k+7} X.$

Similarily, if X contains P_i we get

$\mathfrak{P}_i : q_1 X \models \cdot O q_{k+8} X$

(If $P_i = O$ the modifications are obvious).

Now the algorithm

$$P_i \cup \overline{Z}_i^{(k+7)}$$

is such that

$$\mathrm{Res}_{P_i \cup \overline{Z_i}^{(k+7)}} (q_1 X) = \begin{cases} O q_{k+7} X, \text{ if } X \text{ does not contain } P_i, \\ O q_{\Theta(\overline{Z}_i)+k+7}\, \sigma(P_i, Q_i, X), \text{ if } X \text{ contains } P_i. \end{cases}$$

Let now N_i be some natural number. By $\mathfrak{M}_i$ we denote the *Turing* algorithm $(P_i \cup \overline{Z}_i^{(k+Z)})^{(N_i - 1)}$.
So, we have

$$(14.\ 3) \qquad \mathrm{Res}_{\mathfrak{M}_i} (q_{N_i} X) = \begin{cases} O q_{N_i+k+6} X, & \text{if } X \text{ does not contain } P_i \\ O q_{\Theta(\mathfrak{M}_i)}\, \sigma(P_i, Q_i, X), & \text{if } X \text{ contains } P_i \end{cases}.$$

We regard now the given *Markov's* algorithm (14. 1). Let the length of the word P_i be k_i, $i = 1, 2, \ldots, r$.

We construct first the *Turing* algorithm $\mathfrak{P}_1 \cup \overline{Z}_1^{(k_1+7)}$; if the first rule in (14. 1) is non-terminating, i. e. if it is of the form $P_1 \to Q_1$, we let $\mathfrak{M}_1$ be

$$\mathfrak{M}_1 = \mathfrak{P}_1 \cup \overline{Z}_1^{(k_1+7)} \cup \mathfrak{F}_1,$$

where $\mathfrak{F}_1$ is the algorithm

$q_{k+7} S_\nu S_\nu q_{N_2}, \quad S_\nu \in \mathfrak{S}'$ or $= O$, and $N_2 = \Theta(\overline{Z}_1) + k + 8,$

$q_{N_2-1} S_\nu S_\nu q_1, \quad S_\nu \in \mathfrak{S}'$ or $= O.$

The effect of $\mathfrak{F}_1$ is: if the first rule is not applied pass to the second; if the first rule is applied, apply it anew.

If the first rule of (14. 1) is terminating, i. e. of the form $P_1 \to \cdot Q_1$, we eliminate from $\mathfrak{F}_1$ the quadruples of the second row. So, if the finst rule is applied, the process will finish. (Then, for the next rule we shall employ q_{N_2}).

Let now the *Turing* lagorithms $\mathfrak{M}_1, \mathfrak{M}_2, \ldots, \mathfrak{M}_{k-1}$ for $k - 1 < r$ be constructed.

We construct the *Turing* lagorithm $\mathfrak{M}_k$ corresponding to the k-th rule $P_k \to (\cdot)\, Q_k$. Let N_k be $\Theta(\mathfrak{M}_{k-1})$ (which is in the quadruples of te first row of $\mathfrak{F}_{k-1}$).

If the k-th rule is non-terminating, i. e. if it is of the form $P_k \to Q_k$, we let $\mathfrak{M}_k$ be

$$\mathfrak{M}_k = (\mathfrak{P}_k \cup \overline{Z}_k^{(K_k+7)})^{(N_k - 1)} \cup \mathfrak{F}_k$$

where $\mathfrak{F}_k$ is

$q_{N_k + K_k + G} S_\nu S_\nu q_{N_{k+1}}$ *for* $S_\nu \in \mathfrak{S}'$ *or* $= 0$ and $N_{k+1} = \Theta(\mathfrak{P}_k \cup \overline{Z}^{(K_k+7)})^{N_k - 1}) + 1$

$$q_{N_{k+1} - 1} S_\nu S_\nu q_1 \text{ for } S_\nu \in \mathfrak{S}' \text{ or} = O.$$

If the k-th rule is terminating, i. e. of the form $P_k \rightarrow \cdot Q_k$, we delete the quadruples in the second row of $\mathfrak{F}_k$.

From the construction it is obvious that the *Turing* algorithm

$$Z = \mathfrak{M}_1 \cup \mathfrak{M}_2 \cup, \ldots, \cup \mathfrak{M}_r$$

works exactly as the *Markov* algorithm $\mathfrak{M}$. So

$$\Psi_Z(X) \simeq \mathfrak{M}(X),$$

for every word $X \in \Omega(\mathfrak{S}')$, especially for every word $X \in \Omega(\mathfrak{S})$.

Therefore we have

Theorem 14. 1. *Let $\mathfrak{M}$ be a Markov normal algorithm over $\mathfrak{S}$, and let $\mathfrak{S}' \supset \mathfrak{S}$ be its alphabet. Then, there exist a Turing algorithm Z with $\mathfrak{S}'$ as the printing alphabet, such that*

$$\mathfrak{M}(X) \simeq \Psi_Z(X)$$

for every word $X \in \Omega(\mathfrak{S})$.

As a special case we have

Theorem 14. 2. *Let $\mathfrak{M}$ be a Markov normal algorithm in $\mathfrak{S}$. Then, there exists a Turing algorithm Z, with the same printing alphabet, such that*

$$\mathfrak{M}(X) \simeq \Psi_Z(X)$$

for every word $X \in \Omega(\mathfrak{S})$.

From [4] it is easy to infer the inverse conclusion. Therefore *Turing* algorithms and *Markov*'s normal algorithms are completely equivalent.

LITERATURE

[1] V. Vučković: *Turing algorithms.* Zeitschr. f. math. Logik und Grundlagen d. Math., Bd. 7, (1961), 106—116.

[2] M. Davis: *Computability and Unsolvability.* McGRAW—HILL, New York, 1958.

[3] V. Vučković: *Rekursive Wortarithmetik.* These Publ. T. XIV, (1960), 9—60.

[4] В. С. Чернявский: *Об одном классе нормальных алгорифмов Маркова. Логические исследования.* Академия СССР, Москва (1959), 263—299.

[5] G. Asser und V. Vučković: *Funktionen-Algorithmen.* Zeitschr. f. math. Logik und Grundlagen d. Math. Bd., 7, (1961), 1—8.

[6] G. Asser: *Turing-Maschinen und Markowsche Algorithmen.* Zeitschr. f. math. Logik und Grundlagen der Math., Bd. 5 (1959), 346—365.