ソフトウェア基礎論　37－12
プログラミング言語　27－12
（1990. 12. 13）

RS型ベクトル機械

岩本 宙造　岩間 一雄

九州大学工学部

RS型ベクトル機械(RS-VM)は、ベクトル機械の一標準形の試みとして導入された。それは、*repeat, stretch*の2演算を基本とするベクトル機械である。拡大率$d(m) = 2$のとき、*repeat*はベクトル$(a_1, a_2, \cdots, a_m)$を$(a_1, a_2, \cdots, a_m, a_1, a_2, \cdots, a_m)$に、*stretch*は$(a_1, a_1, a_2, a_2, \cdots, a_m, a_m)$に拡大する演算である。この機械は、$d(m) = 2$のとき多項式領域を多項式時間量に、$d(m) = m$のとき指数時間、多項式交代数を多項式時間量に加速する能力があることが知られている。

本稿では次のことを示す。(i)$d(m) = k$(定数)のRS-VMは、$d(m) = 2$と同じ能力である。(ii)様々な拡大率、例えば$d(m) = m, m^2, \cdots, cm^k, 2^m/m, \cdots, \overbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}^{m}/m, \cdots$は、少なくとも$d(m) = m$と同じ能力を持つ。(iii)$d(m)$が$m$の多項式の範囲では、$d(m) = m$の能力を越えない。また、指数関数（または、それより大きい）$d(m)$のRS-VMは、多項式$d(m)$のそれと能力の差があるかどうかを考察する。

RS-Vector Machine

Chuzo Iwamoto and Kazuo Iwama

Department of Information Science and Communication Engineering
Kyushu University

Fukuoka 812, Japan

RS-vector machines(RS-VMs) were introduced as a canonical form of vector machines. They based on the vector operations called *repeat* and *stretch*. *Repeat* enlarges a vector $(a_1, a_2, \cdots, a_m)$ to $(a_1, a_2, \cdots, a_m, a_1, a_2, \cdots, a_m)$ and *stretch* enlarges to $(a_1, a_1, a_2, a_2, \cdots, a_m, a_m)$, when the operation factor $d(m) = 2$. It is already known that when $d(m) = 2$, RS-VMs can simulate sequential poly-space by poly-time, and when $d(m) = m$, they can simulate exp-time poly-alternation TMs by poly-time.

In this paper, we show that (i)RS-VMs of $d(m) = k$(k is constant) have the same power as those of $d(m) = 2$, (ii)RS-VMs of wide variety of $d(m)$ like $d(m) = m, m^2, \cdots, cm^k, 2^m/m, \cdots, \overbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}^{m}/m, \cdots$ have at least the same power as $d(m) = m$, and (iii)VMs of $d(m) =$ polynomial of m cannot surpass the power when $d(m) = m$. We shall also give an informal observation on if VMs of exponential (or even faster growing) $d(m)$ are strictly more powerful than those of polynomial $d(m)$.

1 Introduction

Vector machines are one of the simplest and the most abstracted parallel computation models. They are completely uniform, their local computation and communication between processors are clearly distinguished, and their communication facilities are usually much simpler (and weaker) than more algorithm-oriented models like PRAMs. Vector machines could therefore be the best for discussing theoretical or structural aspects of parallel computation.

Vector machines(VMs) are first introduced in [6], which shows that the instruction set of shift operations(communication) and bit-wise Boolean operations(local computation) is as powerful as parallel computers of the second class(which can simulate sequential polynomial space by polynomial time). Since then, we have a fairly large literature discussing what kind of operation set is how powerful. For example, each of multiplication(by regarding a vector as an integer), concatenation and so on can replace the shift operations above[3]. Multiplication and integer division without bit-wise operations(we do not need local computation!), are also equivalent to the second class[1]. Shift operations, division and boolean operations[7] are much more powerful; they can speed up $2^{\overbrace{2^{\cdot^{\cdot^{\cdot^{2}}}}}^{n}}$ sequential time to polynomial time. Thus this line of research gives us a lot of interesting and surprising knowledge on the true power of several operations on the vector machines. Unfortunately, however, that knowledge was often given in ad hoc manners and with tricky proofs, which less describes more elementary structure on what kind of (single-step) vector operations are how powerful and on how related they are to the parallel architecture of physical machines.

[4] is the first paper which tried to introduce a "canonical form" to those vector machines. Also it tries to separate, more clearly, the operations of parallel model, into local operations and communications. As local operations it has the element-wise addition and subtraction. As vector operations corresponding to communications, it includes four operations called *repeat, stretch, fold* and *contract*. We call this vector machine *RS-Vector Machine*(*RS-VM* in short). *Repeat* enlarges a vector $(a_1, a_2, \cdots, a_m)$ to $(a_1, a_2, \cdots, a_m, a_1, a_2, \cdots, a_m)$, *stretch* enlarges to $(a_1, a_1, a_2, a_2, \cdots, a_m, a_m)$. *Fold* and *contract* are the inverse operations of *repeat* and *stretch*, respectively. Those operations are defined with a parameter $d(m)$, called the *expansion factor*. Above description of *repeat* and *stretch* is for $d(m) = 2$. If we take $d(m) = m$, then *repeat* and *stretch* generate vectors of length m^2 in the similar manner.

In [4], it is shown that if $d(m) = 2$ then RS-VMs are as powerful as the second class and if $d(m) = m$, then it is much higher, as powerful as exponential-time polynomial-alternation TMs. Thus the instruction set exhibits different powers by only changing the expansion factor. That is the reason why [4] claims that RS-VMs can be a canonical form of VMs.

It would be true that [4] made a significant step toward our goal, i.e.,investigating the fundamental structure of parallelism through generalized vector machines. It is also true, however, that [4] investigated only two particular cases that $d(m) = 2$ and $d(m) = m$. There is seemingly a large gap between $d(m) = 2$ and $d(m) = m$, and also a large parameter space after $d(m) = m$. In this paper, we show that (i)RS-VMs of $d(m) = k (k \geq 2$ is constant) have the same power as those of $d(m) = 2$, (ii)RS-VMs of wide variety of $d(m)$ like $d(m) = m, m^2, \cdots, cm^k, 2^m/m, \cdots, 2^{\overbrace{2^{\cdot^{\cdot^{\cdot^{2}}}}}^{m}}/m, \cdots$ have at least the same power as $d(m) = m$, and (iii)VMs of $d(m) =$ polynomial of m cannot surpass the power when $d(m) = m$. We shall also give an informal observation on if VM's of exponential (or even faster growing) $d(m)$ are strictly more powerful than those of polynomial $d(m)$.

2 Models, Results and Future researches

A *vector of length* m is denoted by $A = (a_1, a_2, \cdots, a_m)$. Each a_i(denoted by $A[i]$) is a nonnegative integer called a *scalar*. Let $A = (a_1, a_2, \cdots, a_i)$, $B = (b_1, b_2, \cdots, b_j)$ and let $\circ$ be a binary operator for scalars. Then $A \circ B = (a_1 \circ b_1, a_2 \circ b_2, \cdots, a_k \circ b_k)$ where $k = \min(i, j)$. The *concatenation* of vectors A and B, denoted by $A \cdot B$ or simply AB, is $(a_1, a_2, \cdots, a_i, b_1, b_2, \cdots, b_j)$. A^0 is (), i.e.,the *empty vector*. For $l \geq 1, A^l = A^{l-1}A$. In this paper, we commonly use upper-case letters, $A, B, \cdots$, for vectors and lower-case $a, b, \cdots$, for scalars.

The vector operations are called *repeat, stretch, fold* and *contract*, and are denoted by $\downarrow, \rightarrow, \uparrow, \leftarrow$, respectively. $d(m)$ always denotes the expansion factor. Let f be a function from vectors to scalars defined by the following (exactly speaking, $f(a_1, a_2, \cdots, a_m)$ should be written as $f((a_1, a_2, \cdots, a_m))$):

$$f(a_1, a_2, \cdots, a_m) = \begin{cases} 0 & \text{if } a_1 = a_2 = \cdots = a_m = 0 \\ a & \text{if all } a_i\text{'s are 0 or } a \text{ and at least one of } a_i\text{'s is } a \\ \text{undefined} & \text{if } a_i\text{'s includes two or more different positives.} \end{cases}$$

For $A = (a_1, a_2, \cdots, a_m)$ and $B = (a_1, a_2, \cdots, a_{d(m)\cdot m})$, $\downarrow, \rightarrow, \uparrow$ and $\leftarrow$ are defined as follows:

$$\begin{aligned} \downarrow A &= A^{d(m)} \\ \rightarrow A &= (a_1^{d(m)}, a_2^{d(m)}, \cdots, a_m^{d(m)}) \\ \uparrow B &= (f(a_1, a_{m+1}, a_{2m+1}, \cdots, a_{(d(m)-1)m+1}), \\ &\quad f(a_2, a_{m+2}, a_{2m+2}, \cdots, a_{(d(m)-1)m+2}), \\ &\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &\quad f(a_m, a_{2m}, a_{3m}, \cdots, a_{d(m)m})) \\ \leftarrow B &= (f(a_1, a_2, , \cdots, a_{d(m)}), \\ &\quad f(a_{d(m)+1}, a_{d(m)+2}, \cdots, a_{2d(m)}), \\ &\quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &\quad f(a_{(m-1)d(m)+1}, \cdots, a_{md(m)})) \end{aligned}$$

Now we are ready to introduce an RS-VM formally. It is defined as a program M accepting a language over $\{0,1\}^*$. M can use:

1. A finite number of scalar variables, $x, y, z, \cdots$.
2. A finite number of scalar constants, $a, b, c, \cdots$.
3. Scalar instructions:

   $$x := y + z,\ x := y - z (= 0 \text{ if } z > y),$$

   if $(x > 0)$ **goto** label, **accept**, **reject**.
4. A finite number of vector variables, $X, Y, Z, \cdots$
5. A finite number of vector constants, $A, B, C, \cdots$
6. A special constant vector $\Omega = (0, 1, 2, 3, \cdots)$. (None of $\downarrow, \rightarrow, \uparrow$ and $\leftarrow$ can be applied to Ω and Ω is assumed to have infinite number of elements.) A special vector valuable IN. When the input string is $i_1 i_2 \cdots i_n \in \{0,1\}^*$, $IN[1]$ holds i_1, $IN[2]$ holds i_2, and so on. We assume $IN[n+1], IN[n+2], \cdots$ holds a large number (≥ 2).
7. Vector instructions:

   $$X := Y+Z, X := Y-Z, X := \downarrow Y, X := \rightarrow Y, X := \uparrow Y, X := \leftarrow Y$$

8. $x := X[1]$

Let VM(d(m)) denote an RS-VM whose expansion factor is $d(m)$. VM($d(m), T(n)$) denote the class of languages that are accepted by VM($d(m)$)s within $T(n)$ steps and ATIMALT($T(n), A(n)$) by alternating TMs within $T(n)$ steps and $A(n)$ changes between the universal and existential states. We introduce $r(m)$ as $m \cdot d(m)$. One can see that $r(m)$ denotes the length of the vector obtained by *repeating*(*stretching*) a vector of length m.

Here are our main results:

Theorem 1 VM($k, poly$)=DSPACE($poly$).

Theorem 2 *For an expansion factor* $d(m)$, *let* $d^{<0>} = 2$ *and* $d^{<n>} = r(d^{<n-1>})$. *Suppose that* $d_s(m)$ *denotes any expansion factor that* $(d_s^{<i>})^2$ *divides* $d_s^{<i+1>}$. *Then*

$$\text{VM}(d_s(m), poly) \supseteq \text{ATIMALT}(2^{poly}, poly).$$

Theorem 3 VM(polynomial of m, $poly$)$\subseteq$ATIMALT($2^{poly}, poly$).

Corollary 1 VM($cm^k, poly$)=ATIMALT($2^{poly}, poly$).

[4] only showed the case for $k = 2$ of Theorem 1 and the case for $c = 1$, $k = 1$ of Corollary 1. As the function $d_s(m)$ described in Theorem 2, there are, for example, $d_s(m) = m, m^2, \cdots, cm^k, 2^m/m, \cdots, 2^{\overbrace{2^{\cdot^{\cdot^{\cdot^{2}}}}}^{m}}/m, \cdots$. Theorem 3 shows that the power of RS-VMs does not increase if the expansion factor stays within polynomial. One would say that RS-VMs of larger expansion factors can simulate obviously those of smaller expansion factors. This is not true at all. What is obvious is that VM(2)(smaller expansion factors) can

simulate VM(4)(larger expansion factors) with a little sacrificer of time. To prove Theorem 2 is far more difficult than the case for $d_s(m) = m$.

Let us observe intuitively why RS-VMs have such computation powers: If a vector, say V, of length m is *stretched(repeated)*, then it becomes of length m^2. We can regard the vector of length m^2 as a square matrix of m columns and m rows. Note that each column is equal to the original V in $\rightarrow V$ and each row the same as V in $\downarrow V$. Then it is easy to take a direct product of $\rightarrow V$ and $\downarrow V$ by, intuitively, placing one over the other. Thus we can increase the "complexity" of vectors from m to m^2. To construct a more powerful vector machine, we could set $d(m) = m^2$. When a vector of length m is *stretched(repeated)*, then it becomes of length m^3. We regard this vector as a cube vector. What about extending naturally the direct product mentioned above to get a vector of complexity m^3? The answer is negative. Because RS-VMs have only two vector operations, they can still make a direct-product of only two objects. Thus the similar direct product of $\downarrow V$ and $\rightarrow V$ on a VM(m^2) gives us only m^2 different elements, which means that the complexity of the vector increases from m to m^2, the same as before. Of course this direct product generates much more elements than m^2. The same element appears repeatedly in a complicated form, which is very undesirable for proceeding computation. This fact makes the proof of Theorem 2 much harder than it looks. We have a strong conjecture that the power of Theorem 2 is the best possible for RS-VMs of any expansion factor.

Future researches would be as follows:

- Is the conjecture above true? A significant difficulty for proving it is the existence of the special constant vector $\Omega = (0, 1, 2, \cdots)$. In this paper, we use, in a sense, it for picking out of diagonal elements. We never use "actual" values of this vector. If we could remove this Ω from the program, it would be a nice improvement by itself and would contribute a lot to the goal.
- Supposing that the conjecture is correct, is there any (different kind of) vector operation set whose power increases unlimitedly with the expansion factor?
- Is there any expansion factor characterizing a natural complexity class between those of Theorem 1 and Corollary 1.

In what follows, we can only give the proof of Theorem 2, for the space reason.

3 Proof of Theorem 2

3.1 An important idea

Before describing a formal proof for the general $d_s(m)$, we shall sketch how the proof proceeds using a simplest particular case, i.e.,the case for $d_s(m) = m$. In the rest of the paper, we simply use $d(m)$ for the expansion factor $d_s(m)$. Recall that this expansion factor enlarges a vector of length m to a vector of length m^2 by *repeat* or *stretch*. Among several techniques used in simulating alternating TMs by VMs, the most fundamental one could be to construct a vector of length $(2^{2^n})^2$ quickly, in which all the different 0/1 subvectors of length 2^n appear as follows.

$$C(n) = (\overbrace{\underbrace{00\cdots00}_{2^n}\,22\cdots2\ \underbrace{00\cdots01}_{2^n}\,22\cdots2\ \cdots\cdots\ \underbrace{11\cdots11}_{2^n}\,22\cdots2}^{2^{2^n}\text{sections(length }(2^{2^n})^2)})^{\overbrace{\ }^{2^{2^n}}} \quad (1)$$

(section of length 2^{2^n}; 2^{2^n}; 2^{2^n})

Thus the vector consists of 2^{2^n} *sections*. Each section begins with a 0/1 vector of length 2^n(one of the different 2^{2^n} patterns) followed by $2^{2^n} - 2^n$ 2's.

$C(n)$ is constructed step by step. To do so, we introduce five vectors, $A(h)$, $B(h)$, $C(h)$, D and $E(h)$, where the parameter h means that the vector contains all the 0/1 patterns of length 2^h. Note that the length of all the five vectors is $(2^{2^n})^2$ that does not depend on h. $C(h)$ plays a key role. Fig.1 shows how to construct $C(h+1)$ from $C(h)$. Like $C(n)$ above, $C(h)$ also consists of 2^{2^n} sections of length 2^{2^n}:

$$C(h) = (\overbrace{(\underbrace{0\cdots00}_{2^h}\cdots\underbrace{0\cdots00}_{2^h}2\cdots2)\,(\underbrace{0\cdots01}_{2^h}\cdots\underbrace{0\cdots01}_{2^h}2\cdots2)\cdots\cdots}^{\text{division of }2^{2^n}\text{ sections}})^{\frac{2^{2^n}}{2^{2h}}} \quad (2)$$

(section of length 2^{2^n}; section of length 2^{2^n}; 2^{2^n}; 2^{2^n})

As one can see, we have already constructed all different vectors of length $2^h (h \leq n)$, which means that the number of all the different sections is 2^{2^h}. We call such a sequence of the 2^{2^h} sections *division*.

Before describing how to construct $C(h+1)$, we introduce, for easier understanding, a two-dimensional way of representing vectors. For example, let $X = (a, b, c, d)$. We can write $\downarrow X(h)$ and $\rightarrow X(h)$ as

$$\downarrow X = (a, b, c, d,\ a, b, c, d,\ a, b, c, d,\ a, b, c, d) \quad (3)$$

and

$$\rightarrow X = (a, a, a, a,\ b, b, b, b,\ c, c, c, c,\ d, d, d, d). \quad (4)$$

We will often regard those as matrices, where all the elements in a single column of $\downarrow X$ are the same and all the elements in a single row of $\rightarrow X$ are the same, or the original X corresponds with each column of $\rightarrow X$ and with each row of $\downarrow X$. Now we construct $C(h+1)$ from $\rightarrow C(h)$ and $\downarrow C(h)$. Fig.1 shows left upper part of the two vectors $\rightarrow C(h)$ and $\downarrow C(h)$ assuming that one is placed over the other. $s_h(i)$ denotes the ith section of (2), for example, $s_h(1) = (\overbrace{\underbrace{0\cdots01}_{2^h}\cdots\underbrace{0\cdots01}_{2^h}2\cdots22}^{\text{section of length }2^{2^n}})$. We also introduce a new vector denoted by Z of the same length as $\downarrow C(h)$, which is also regarded as being placed at the same position in Fig.1. Z has some *boxes* and *diagonal elements*. Each shaded rectangle in Fig.1 of 2^{2^n} rows $\times\ 2^{2^n} \cdot 2^{2^h}$ columns is called a *box*. Each box is divided into 2^{2^h} *squares* consisting of 2^{2^n} rows $\times\ 2^{2^n}$ columns. Each square contains 2^{2^n} diagonal elements (shown by a broken line in Fig.1). Only those diagonal elements are significant in Z. As shown in Fig.2, they have 2^n 1/2 elements followed by $2^{2^n} - 2^n$ 2's. The 1/2 portion begins with, from left-top to right-bottom, 2^h 1's, then 2^h 2's, again 2^h 1's, 2^h 2's and so on. All the other elements of Z are 0. Fig.2 is the case $h = 2$. Using this vector Z for a masking purpose, we can get $C(h+1)$ as a direct product of all the different patterns of $\rightarrow C(h)$ and $\downarrow C(h)$. One can see that a single box of Fig.1 corresponds to all the 2^{2^h} different patterns of $\downarrow C(h)$ and a single pattern of $\rightarrow C(h)$. We now construct a vector of length $(2^{2^n})^4$(=the length of $\rightarrow C(h)$), say $\tilde{C}(h+1)$. $\tilde{C}(h+1)[i]$ is the same as $\rightarrow C(h)[i]$ if $Z[i] = 1$ and the same as $\downarrow C(h)[i]$ if $Z[i] = 2$. All the other (nondiagonal) positions are set to 0. By *folding* $\tilde{C}(h+1)$, each column is compressed to a single element, which is $C(h+1)$. $C(h+1)$ contains all the different 0/1 subvectors of length 2^{h+1}, i.e.,the length of each 0/1 subvectors are *doubled*.

Now the problem is reduced to how to construct the vector Z, for which we need the vectors A, B, D and E. Recall that all of them are of the same length as C.

$$A(h) = (\underbrace{00\cdots0}_{2^{2^n}}\underbrace{11\cdots1}_{2^{2^n}}\underbrace{22\cdots2}_{2^{2^n}}\cdots\cdots\underbrace{(2^{2^h}-1)\cdots(2^{2^h}-1)}_{2^{2^n}})^{\frac{2^{2^n}}{2^{2^h}}} \quad (5)$$

$$B(h) = (\underbrace{00\cdots\cdots0}_{2^{2^n}\cdot2^{2^h}}\underbrace{11\cdots\cdots1}_{2^{2^n}\cdot2^{2^h}}\cdots\cdots\underbrace{(2^{2^h}-1)\cdots\cdots(2^{2^h}-1)}_{2^{2^n}\cdot2^{2^h}})^{\frac{2^{2^n}}{(2^{2^h})^2}} \quad (6)$$

$$D = (0, 1, 2, \cdots, (2^{2^n}-1))^{2^{2^n}} \quad (7)$$

$$E(h) = (\overbrace{\underbrace{11\cdots1}_{2^h}\underbrace{22\cdots2}_{2^h}\cdots\cdots\underbrace{11\cdots1}_{2^h}\underbrace{22\cdots2}_{2^h}}^{2^{2^n}}2\cdots\cdots2)^{2^{2^n}} \quad (8)$$

Note that the length of $(\underbrace{00\cdots\cdots0}_{2^{2^n}\cdot2^{2^h}})$ in $B(h)$ coincides with the length of $(\underbrace{00\cdots0}_{2^{2^n}}\underbrace{11\cdots1}_{2^{2^n}}\cdots\cdots\underbrace{(2^{2^h}-1)\cdots(2^{2^h}-1)}_{2^{2^n}})$ in $A(h)$. We first make a matrix called Z_{BOX} which has 1's inside the boxes of Fig.1 and 0's outside. To do so, we construct $\rightarrow A(h)$ and $\downarrow B(h)$. Z_{BOX} is obtained as a vector which has value 1 in the position where $\rightarrow A(h)$ and $\downarrow B(h)$ coincide. We next make a matrix called Z_{DIA} which has 1's on the diagonal lines(not only inside the boxes but

every where) in Fig.1. To do so, we construct $\rightarrow D$ and $\downarrow D$. Z_{DIA} is a vector which has value 1 in the position where $\rightarrow D$ and $\downarrow D$ coincide. We compute $\tilde{Z} := Z_{BOX} \wedge Z_{DIA}$, where $\wedge$ is element-wise *and* operation, and this operation is easily realized by $+$ and $-$. We need $\downarrow E(h)$ to place the correct values of the diagonal elements of Z. One can see that Z is finally obtained by changing the elements of $\downarrow E(h)$ into 0 in the positions where Z has 0. This kind of "masking" operation is frequently used in this proof and will be described in detail later.

3.2 Sketch of the proof

Now we begin the proof for the general case, i.e.,the case for an arbitrary $d(m)$ such that $(d^{<i>})^2$ divides $d^{<i+1>}$. The basic idea is similar as above, but the entire structure is much more complicated. For simplicity, we first prove $\mathrm{VM}(d_s(m), poly) \supseteq \mathrm{DTIME}(2^{p^{oly}})$ and extend it to $\mathrm{VM}(d_s(m), poly) \supseteq \mathrm{ATIMALT}(2^{p^{oly}}, poly)$ later. Let M be a $2^{t(n)}$ time-bounded, deterministic, single read-write tape TM with only 0 and 1 as its tape symbols. We construct a vector machine V of $O((t(n))^k)$ steps (constant k is not large) which simulates M of $2^{t(n)}$ steps. Since M finishes its operation within $2^{t(n)}$ steps, it uses at most $2^{t(n)}$ tape cells. Generally speaking, we can determine whether M accepts its input by creating M's computation sequence S of length $2^{t(n)}$. However, V does not try to make the particular sequence S but makes all possible sequences first and then find if S exists in the set, which is the standard approach in the parallel environment. The vector machine V first constructs a vector representing all possible tapes and then a vector representing all states and head-positions of TM M. All possible sequences of those configurations are now obtained by combining both vectors. We represent a single configuration by $2^{t(n)}$ elements and a single sequence consists of $2^{t(n)}$ such configurations. Hence the sequence of configurations is of length $2^{t(n)} \times 2^{t(n)}$.

Roughly speaking, the vector machine V operates as follows: (i)To generate all possible tapes of length $2^{t(n)}$, V makes a "very long" vector in which all different 0/1 vectors of length $2^{t(n)}$ appear as subvectors. (ii)Let k be the number of TM M's states. We use a single integer from 1 to $k \cdot 2^{t(n)}$ to denote both M's state and head position at some moment. For example, value $(j-1) \cdot 2^{t(n)} + i$ represents the jth state and head-position i(M is now in state j and places its head on the ith cell from the left end of the tape). Vector machine V makes a vector which contains all numbers from 1 to $k \cdot 2^{t(n)}$. (iii) V then generate all the configurations of TM M by taking, in a sense, a direct product of all the tapes (i) above and all the state/head-positions (ii). (iv)Now V has all the configurations of M. By concatenating two of them, we can generate all the sequences of M's configurations for two steps, then all the sequences for four steps, and so on, up to for $2^{t(n)}$ steps. (v)V then "shifts" each sequence of configurations $2^{t(n)}$ bits(corresponding to a single step) to the right. We can check whether the sequence of configurations is proper or not by comparing that sequence with its shifted counterpart.

We should mention what $d^{<i>}$ comes from and what it means. As one can see later, all constant vectors we use in this proof are of length 2. When we *stretch(repeat)* a vector of length m, we get a vector of length $r(m) = m \cdot d(m)$. Then one can see that $d^{<i>}$ shows the length of the vector which is *stretched(repeated)* i times beginning with a constant vector of length 2. We use several vectors as before(Section 3.1); including $A(h)$, $B(h)$, $C_T(h)$, $C_{TS}(h)$, $C_H(h)$, $C_{HS}(h)$, D, $E(h)$. Let $2t(n) = T(n)$, namely $2^{T(n)} = 2^{t(n)} \times 2^{t(n)}$. Then all vectors are of length $d^{<T(n)+2>}$, which is independent of h. T, H and S represents Tape, state/Head-position and Shift, respectively. Parameter h also plays the same role as before.

3.3 Constructing all tapes

3.3.1 Structure of $C_T(h)$

We first construct $C_T(0)$, and then $C_T(1)$ from $C_T(0)$, $C_T(2)$ from $C_T(1)$ and so on, until $C_T(t(n))$. $C_T(h)$ looks like:

$$C_T(h) = (\underbrace{S_h(0), S_h(1), \cdots\cdots\cdots\cdots}_{\text{subdivision of } d^{<h>} \text{ sections}})^{\frac{d^{<T(n)+1>}}{d^{<h>}}}, \tag{9}$$

where $S_h(i)$ denotes $(\overbrace{\underbrace{(BN_{2^h}(i))^{2^{T(n)-h}}, 2, 2, \cdots, 2}_{\text{subsection of length } d^{<T(n)+1>}}}^{\text{section of length } \frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}})^{\frac{d^{<T(n)+2>}}{(d^{<T(n)+1>})^2}}$ and

$BN_j(i)$ denotes the binary representation of the integer i using j bits. Although $C_T(h)$ seems much more complicated, it essentially has the same structure as $C(h)$ in Section 3.1. Recall that the length of $C_T(h)$ is $d^{<T(n)+2>}$, not depending on h. Each $C_T(h)$ is divided into $d^{<T(n)+1>}$ *sections* of length $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}$. Each section is divided into $\frac{d^{<T(n)+2>}}{(d^{<T(n)+1>})^2}$ *subsections* of length $d^{<T(n)+1>}$. All subsections in a particular section has the same bit-pattern. $d^{<h>}$ consecutive sections are called a *subdivision* which contains all the different 0/1 subvectors of length 2^h. $\frac{d^{<h+1>}}{d^{<h>}} (> d^{<h>})$ consecutive sections are called a *division*. First $2^{T(n)}$ bits of each subsection contains $2^{T(n)-h}$ repetitions of the same subvectors of length 2^h which is one of $0 \sim 2^{2^h} - 1$ if it is regarded as a binary number, and the rest of the subsection hold 2's everywhere. This value "2" is not important, sometimes called a *dummy element*. Note that a single subdivision of (9) has $d^{<h>}$ sections and that we need only $2^{2^h} (\ll d^{<h>})$ sections to cover all the different bit-patterns of length 2^h. In reality, a subdivision contains a lot of the same sections that appear in a complicated fashion. For example, when $d(m) = m^2$,

$$C_T(h) = (S_h(0), S_h(1), S_h(0), S_h(1), S_h(2), S_h(3), S_h(2), S_h(3), \cdots)^{\frac{d^{<T(n)+1>}}{d^{<h>}}}. \tag{10}$$

Structure of such a repetition of $S_h(i)$ depends on $d(m)$.

3.3.2 Constructing $C_T(t(n))$

Constructing $C_T(0)$: We first construct $C_T(0)$. We will not mention so frequently on the time complexity, but one can verify easily that we do not spend more than a polynomial time supposing that $t(n)$ (and $T(n)$) is a polynomial. A constant vector (0,1) is *repeated* $T(n)+1$ times, which gives

$$(\underbrace{0, 1, 0, 1, \cdots, 0, 1}_{d^{<T(n)+1>}}). \tag{11}$$

Stretching this vector, we get

$$(\underbrace{0, 0, \cdots\cdots\cdots, 0}_{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}, \underbrace{1, 1, \cdots\cdots\cdots, 1}_{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}})^{\frac{d^{<T(n)+1>}}{2}}. \tag{12}$$

By *repeating* constant vector (0,0) $T(n)+1$ times, we get

$$(\underbrace{0, 0, \cdots, 0}_{d^{<T(n)+1>}}). \tag{13}$$

By adding Ω to (13), we get

$$(0, 1, 2, 3, \cdots, (d^{<T(n)+1>} - 1)). \tag{14}$$

(Recall that Ω is a vector of infinite length. Vector (13) is used only for "cutting" Ω at the desired position.)

Beginning with constant vector (1,1), we get (2,2) by adding two (1,1)'s, (4,4) by adding two (2,2)'s, finally $(2^{T(n)}, 2^{T(n)})$ in roughly $T(n)$ steps. Subtracting (1,1) from $(2^{T(n)}, 2^{T(n)})$, we get $(2^{T(n)} - 1, 2^{T(n)} - 1)$. *Repeating* this vector $T(n)+1$ times, we get

$$(\underbrace{(2^{T(n)} - 1), (2^{T(n)} - 1), \cdots, (2^{T(n)} - 1)}_{d^{<T(n)+1>}}). \tag{15}$$

Subtracting (15) from (14) implies

$$(\overbrace{\underbrace{0, 0, \cdots, 0}_{2^{T(n)}}, 1, 2, 3, 4, \cdots\cdots}^{d^{<T(n)+1>}}). \tag{16}$$

By *repeating* constant vector (1,1) $T(n)+1$ times, we get

$$(\underbrace{1, 1, \cdots, 1}_{d^{<T(n)+1>}}). \tag{17}$$

(17) $-$ (16) gives us

$$(\overbrace{\underbrace{1, 1, \cdots, 1}_{2^{T(n)}}, 0, 0, 0, 0, \cdots\cdots}^{d^{<T(n)+1>}}). \tag{18}$$

(17) − (18) switches 0's and 1's of (18) like

$$(\overbrace{\underbrace{0,0,\cdots,0}_{2^{T(n)}},1,1,1,1,\cdots\cdots}^{d^{<T(n)+1>}}). \tag{19}$$

Adding (19) twice, we get

$$(\overbrace{\underbrace{0,0,\cdots,0}_{2^{T(n)}},2,2,\cdots\cdots\cdots,2}^{d^{<T(n)+1>}}). \tag{20}$$

By *Repeating* this vector, we obtain

$$\underbrace{(\overbrace{\underbrace{0,0,\cdots,0}_{2^{T(n)}},2,2,\cdots\cdots\cdots,2}^{d^{<T(n)+1>}})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}}_{d^{<T(n)+2>}}. \tag{21}$$

Note that vector (12) also can be written as follows:

$$\underbrace{((\overbrace{\underbrace{00\cdots0}_{2^{T(n)}}00\cdots0}^{d^{<T(n)+1>}})^{\frac{d^{<T(n)+2>}}{(d^{<T(n)+1>})^2}}(\overbrace{\underbrace{11\cdots1}_{2^{T(n)}}11\cdots1}^{d^{<T(n)+1>}})^{\frac{d^{<T(n)+2>}}{(d^{<T(n)+1>})^2}})^{\frac{d^{<T(n)+1>}}{2}}}_{d^{<T(n)+2>}} \tag{22}$$

By $\{(22)-(21)\}+(21)$, we get $C_T(0)$ as follows:

$$C_T(0) = (S_0(0), S_0(1))^{\frac{d^{<T(n)+1>}}{2}} \tag{23}$$

$$= ((\overbrace{\underbrace{00\cdots02\cdots2}_{2^{T(n)}}}^{\text{subsection}})^{\frac{d^{<T(n)+2>}}{(d^{<T(n)+1>})^2}}(\overbrace{\underbrace{11\cdots12\cdots2}_{2^{T(n)}}}^{\text{subsection}})^{\frac{d^{<T(n)+2>}}{(d^{<T(n)+1>})^2}})^{\frac{d^{<T(n)+1>}}{2}} \tag{24}$$

(section of length $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}$; section of length $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}$; subsection: $d^{<T(n)+1>}$; subsection: $d^{<T(n)+1>}$)

Constructing $C_T(t(n))$: Suppose that we have constructed $C_T(h)$ up to some $h < t(n)$:

$$C_T(h) = (\underbrace{S_h(0), S_h(1), \cdots\cdots\cdots\cdots}_{\text{subsection of } d^{<h>} \text{ sections}})^{\frac{d^{<T(n)+1>}}{d^{<h>}}} \tag{25}$$

We construct $C_T(h+1)$ with keeping the following feature of $C_T(h)$ in our mind: When we divide $C_T(h)$ into subdivisions, one subdivision contains $d^{<h>}$ sections and every subdivision has all the different 0/1 subvectors of length 2^h. Since a division is a collection of subdivisions, each division contains all different 0/1 subvectors of length 2^h also(one division contains $\frac{d^{<h+1>}}{d^{<h>}}$ sections). Intuitively we make a direct product of $d^{<h>}$ sections(=a subdivision) and $\frac{d^{<h+1>}}{d^{<h>}}$ sections(=a division) of $C_T(h)$. (This is much easier in our current environment than taking a direct product of two subdivisions. One can see the reason in a moment: A key point is that our current matrix is not square as in 3.1 but very narrow.) As a result, $d^{<h>} \times \frac{d^{<h+1>}}{d^{<h>}} = d^{<h+1>}$ sections will become a new subdivision, which contains all different 0/1 subvectors of length 2^{h+1}.

We first make $\rightarrow C_T(h)$ and $\downarrow C_T(h)$. Fig.3 shows left upper part of the two vectors assuming that one placed over the other. Note that the length of both $\rightarrow C_T(h)$ and $\downarrow C_T(h)$ is $d^{<T(n)+3>}$. We regard each of them as a matrix composed of $\frac{d^{<T(n)+3>}}{d^{<T(n)+2>}}$ rows and $d^{<T(n)+2>}$ columns($\frac{d^{<T(n)+3>}}{d^{<T(n)+2>}} \gg d^{<T(n)+2>}$). Namely, when we *repeat* $C_T(h)$, a single element of $C_T(h)$ naturally corresponds to a single column of $\downarrow C_T(h)$ in its matrix representation. If $C_T(h)$ is *stretched*, however, a single element of $C_T(h)$ is "stretched" to $\frac{d^{<T(n)+3>}}{d^{<T(n)+2>}}$ elements which occupy not a single row but $\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2}$ rows. Thus all the elements of consecutive $\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2}$ rows are completely the same. Since the length of one section of $C_T(h)$ is $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}$, $\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2} \times \frac{d^{<T(n)+2>}}{d^{<T(n)+1>}} = \frac{d^{<T(n)+3>}}{d^{<T(n)+2>}d^{<T(n)+1>}}$ rows of $\rightarrow C_T(h)$ correspond to a single section of $C_T(h)$. As for $\downarrow C_T(h)$, the entire $\frac{d^{<T(n)+3>}}{d^{<T(n)+2>}}$ elements on a single column are the same and $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}$ columns constitute to one section. We again introduce a vector Z of the same length as $\downarrow C_T(h)$, which is also regarded as the same-styled matrix as in Fig.3. Z has *boxes* and *diagonal sticks*. Each shaded portion in Fig.3 consists of $\frac{d^{<T(n)+3>}}{d^{<T(n)+2>}d^{<T(n)+1>}}$ rows and $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}\frac{d^{<h+1>}}{d^{<h>}}$ columns which we again call a *box*. The $\frac{d^{<T(n)+3>}}{d^{<T(n)+2>}d^{<T(n)+1>}}$ rows correspond to a section of $C_T(h)$ as mentioned above, and the $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}\frac{d^{<h+1>}}{d^{<h>}}$ columns to a division$(=\frac{d^{<h+1>}}{d^{<h>}}$ sections). As a result, each box includes a single section of $\rightarrow C_T(h)$ and $\frac{d^{<h+1>}}{d^{<h>}}$ sections of $\downarrow C_T(h)$.

Each box is divided into $\frac{d^{<h+1>}}{d^{<h>}}$ *smallboxes* which are composed of $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}$ columns and the $\frac{d^{<T(n)+3>}}{d^{<T(n)+2>}d^{<T(n)+1>}}$ rows(the same as the box). Thus a smallbox corresponds to one section of $\downarrow C_T(h)$. Fig.4 shows magnification of this smallbox. It is further divided into *tinyboxes* each of which consists of $\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2} \cdot d^{<T(n)+1>}$ rows $\times$ $d^{<T(n)+1>}$ columns. Each tinybox corresponds to a subsection of both $\downarrow C_T(h)$ and $\rightarrow C_T(h)$ and contains a repetition of a single bit-pattern among $\underbrace{00\cdots0}_{2^h}$ to $\underbrace{11\cdots1}_{2^h}$.

Fig.5 shows magnification of the tinybox. All elements except the left upper part of $\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2} \cdot 2^{T(n)}$ rows $\times$ $2^{T(n)}$ columns are 2's. Recall that only first $2^{T(n)}$ bits of each subsection are important for us. In this figure are also shown $2^{T(n)}$ "sticks" of $\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2}$ rows $\times$ 1 column. We shall call them "diagonal elements" or "diagonal sticks". The length $\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2}$(= the height of a stick) should be associated with the fact that a single element of $C_T(h)$ is "stretched" to $\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2}$ rows. Only those diagonal sticks are significant in Z. Similarly as in Fig.2 there are $2^{T(n)}$ sticks whose elements are 1 or 2 followed by $d^{<T(n)+1>} - 2^{T(n)}$ sticks whose elements are all 2's. In the left upper part($\frac{d^{<T(n)+3>}}{(d^{<T(n)+2>})^2} \cdot 2^{T(n)}$ rows $\times$ $2^{T(n)}$ columns), first 2^h sticks are value 1's, next 2^h are value 2's, next 2^h are value 1's, next 2^h are value 2's, and so on. We use each 2^h sticks whose values are 1's(2's, respectively) for picking out 0/1 subvectors of length 2^h in $\rightarrow C_T(h)$($\downarrow C_T(h)$, respectively). More precisely, we construct a vector of length $d^{<T(n)+3>}$ (=the length of $\rightarrow C_T(h)$), say $\tilde{C}_T(h+1)$. $\tilde{C}_T(h+1)[i]$ is the same as $\rightarrow C_T(h)[i]$ if $Z[i] = 1$ and the same as $\downarrow C_T(h)[i]$ if $Z[i] = 2$. All the other positions are set to 0. Note that all non-zero elements in each column are the same. (Those correspond to a single "element" of $\tilde{C}_T(h)$.) By *folding* $\tilde{C}_T(h+1)$, all elements of a single column of Fig.3 compressed to a single element, which is $C_T(h+1)$. $C_T(h+1)$ contains all the different 0/1 subvectors of length 2^{h+1} in a new single subdivision of $d^{<h>} \times \frac{d^{<h+1>}}{d^{<h>}} = d^{<h+1>}$ sections. The new subsection contains $2^{T(n)-(h+1)}$ subvectors of length 2^{h+1}, i.e.,the length of each 0/1 subvectors are *doubled*.

Now the problem is reduced to how to construct Z in polynomial time. We again introduce four vectors A, B, D, E. Recall that all of them are of the same length as $C_T(h)$.

$$A(h) = \underbrace{(0^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}, 1^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}, \cdots, (d^{<h>}-1)^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}})^{\frac{d^{<T(n)+1>}}{d^{<h>}}}}_{d^{<T(n)+2>}} \tag{26}$$

$$B(h) = \underbrace{(0^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}\frac{d^{<h+1>}}{d^{<h>}}}, \cdots, (d^{<h>}-1)^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}\frac{d^{<h+1>}}{d^{<h>}}})^{\frac{d^{<T(n)+1>}}{d^{<h+1>}}}}_{d^{<T(n)+2>}} \tag{27}$$

$$D = \underbrace{(0,1,2,\ldots,(d^{<T(n)+1>}-1))^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}}_{d^{<T(n)+2>}} \tag{28}$$

$$E(h) = \begin{cases} \underbrace{(\overbrace{(1^{2^h}, 2^{2^h})^{2^{T(n)-(h+1)}}}^{d^{<T(n)+1>}}, 2, 2, \cdots, 2)^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}}_{d^{<T(n)+2>}}, & \text{for } h < t(n) \\ \underbrace{(1^{2^h}, 1^{2^h})^{2^{T(n)-(h+1)}}, 2, 2, \cdots, 2)^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}}_{d^{<T(n)+2>}}, & \text{for } h = t(n) \\ \underbrace{(\underbrace{(1^{2^{h-1}}, 2^{2^{h-1}})^{2^{T(n)-h}}}_{2^{T(n)}}, 2, 2, \cdots, 2)^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}}_{d^{<T(n)+2>}}, & \text{for } h > t(n) \end{cases} \tag{29}$$

Note that $E(h)$ makes a slightly incontinuous change at $h = t(n)$. The purpose of $E(t(n))$ is different from the others. $E(t(n))$ is used when we construct all configurations by combining all the tape vectors with all the state/head-position vectors. These (26), (27), (28)

and (29) are based on the same idea as (5), (6), (7) and (8), respectively.

We again make a matrix called Z_{BOX} which has 1's inside the boxes of Fig.2 and 0's outside. To do so, we construct $\rightarrow A(h)$ and $\downarrow B(h)$, and make a vector which has value 1 in the position where $\rightarrow A(h)$ and $\downarrow B(h)$ coincide. We next construct a matrix called Z_{DIA} which has 1's on the diagonal sticks(not only inside the boxes but every where) in Fig.3. We make a vector which has value 1 in the position where $\rightarrow D$ and $\downarrow D$ coincide. We compute $\tilde{Z} := Z_{BOX} \wedge Z_{DIA}$. We need $\downarrow E(h)$ to place the correct value(1 or 2) on diagonal sticks of Z. Note that Z has non-zero elements in the positions where $\tilde{Z}$ has 1, which we can get by changing the elements of $\downarrow E(h)$ into 0's where $\tilde{Z}$ has 0.

Now we construct $A(h)$, $B(h)$, D, $E(h)$. We first make the following vector just as (14)

$$(0, 1, 2, 3, \cdots, (d^{<h>} - 1)) \tag{30}$$

By *repeating* this vector $(T(n) - h + 1)$ times, we get

$$\underbrace{(0, 1, 2, 3, \cdots, (d^{<h>} - 1))^{\frac{d^{<T(n)+1>}}{d^{<h>}}}}_{d^{<T(n)+1>}} \tag{31}$$

$A(h)$ is obtained by *stretching* this vector. By *stretching* vector (30), we get

$$\underbrace{(0^{\frac{d^{<h+1>}}{d^{<h>}}}, 1^{\frac{d^{<h+1>}}{d^{<h>}}}, 2^{\frac{d^{<h+1>}}{d^{<h>}}}, \cdots, (d^{<h>} - 1)^{\frac{d^{<h+1>}}{d^{<h>}}})}_{d^{<h+1>}}. \tag{32}$$

By *repeating* this vector $(T(n) - h)$ times, we get

$$\underbrace{(0^{\frac{d^{<h+1>}}{d^{<h>}}}, 1^{\frac{d^{<h+1>}}{d^{<h>}}}, 2^{\frac{d^{<h+1>}}{d^{<h>}}}, \cdots, (d^{<h>} - 1)^{\frac{d^{<h+1>}}{d^{<h>}}})^{\frac{d^{<T(n)+1>}}{d^{<h+1>}}}}_{d^{<T(n)+1>}}. \tag{33}$$

$B(h)$ is obtained by *stretching* this vector. We obtain D simply by *repeating* (14).

Before describing how to construct $E(h)$, we introduce another subtraction operation $-^*$ as follows:

$$x -^* y = \begin{cases} x - y, & \text{for } x \geq y, \\ x, & \text{for } x < y. \end{cases}$$

It is not hard to see that $X -^* Y$ can be computed using the normal subtraction as follows. (i)$Z := Y - X$. (ii)$Z := (1, 1, \cdots, 1) - Z$.(Namely, $Z[i] = 1$ iff $X[i] \geq Y[i]$.) (iii)Repeat[†] $\{Z := Z + Z\}$ until each element of Z become sufficiently large. (iv)$(X - Y) + (X - Z)$ is the answer.

Now we will show how to make the vector $E(h)(h < t(n))$: Let X denote (20). We repeat $\{X := X + X\}$ $T(n)$ times. Then $\{(14) - (16) - X\}$ implies

$$\underbrace{(0, 1, 2, 3, \cdots, (2^{T(n)} - 1), 0, 0, \cdots, 0)}_{d^{<T(n)+1>}}. \tag{34}$$

In a similar way as (15), we can make $\underbrace{(2^{T(n)-1}, 2^{T(n)-1}, \cdots, 2^{T(n)-1})}_{d^{<T(n)+1>}}$.

Subtracting this vector from (34) by $-^*$ operation, we obtain

$$\underbrace{(0, 1, \cdots, (2^{T(n)-1} - 1), 0, 1, \cdots, (2^{T(n)-1} - 1), 0, 0, \cdots, 0)}_{d^{<T(n)+1>}}, \tag{35}$$

We again subtract $\underbrace{(2^{T(n)-2}, 2^{T(n)-2}, \cdots, 2^{T(n)-2})}_{d^{<T(n)+1>}}$ by $-^*$ operation, $\cdots$, and finally subtract $\underbrace{(2^{h+1}, 2^{h+1}, \cdots, 2^{h+1})}_{d^{<T(n)+1>}}$ by $-^*$ operation.

The result is

$$\underbrace{(\overbrace{(0, 1, 2, 3, \cdots, (2^{h+1} - 1)}^{2^{T(n)}})^{2^{T(n)-(h+1)}}, 0, 0, \cdots, 0)}_{\text{subsection of length } d^{<T(n)+1>}}. \tag{36}$$

[†]Do not confuse with the vector operation *repeat*.

We subtract $\underbrace{(2^h - 1, 2^h - 1, \cdots, 2^h - 1)}_{d^{<T(n)+1>}}$ from (36) and *repeat* it once more. Then we get

$$\underbrace{(\overbrace{(\underbrace{0, 0, 0, \cdots, 0}_{2^h}, \underbrace{1, 2, 3, \cdots, 2^h - 1}_{2^h})}^{2^{T(n)}})^{2^{T(n)-(h+1)}}, 0, 0, \cdots, 0)}_{\text{subsection of length } d^{<T(n)+1>}}{}^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}. \tag{37}$$

By *stretching* (17), we call the resulting vector as Y. Executing $Y - \{Y - (37)\}$, we can change all non-zero elements to 1. By adding this vector to (21), we get $E(h)$ for $h < t(n)$. $E(h)$ for $h \geq t(n)$ is similar.

3.4 Constructing all state/head-positions

Next we construct all state/head-position vector. Suppose that the given TM M has states $s_1, s_2, \cdots s_k$. Then a pair (s_j, i) of state and head-position$(1 \leq i \leq 2^{t(n)})$ is represented by a single integer $(j-1) \times 2^{t(n)} + i$, which is between 1 and $k \cdot 2^{t(n)}$. In a similar way as with (34), we construct

$$\underbrace{(1, 2, 3, \cdots, k \cdot 2^{t(n)}, 0, 0, \cdots, 0)}_{d^{<t(n)>}}{}^{\frac{d^{<T(n)+1>}}{d^{<t(n)>}}} \tag{38}$$

Value 0's in this vector are just for adjusting the length. By *stretching* this vector, we get

$$\underbrace{(1^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}, \cdots, (k \cdot 2^{t(n)})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}, 0^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}, \cdots)}_{\text{subdivision of } d^{<t(n)>} \text{ sections}}{}^{\frac{d^{<T(n)+1>}}{d^{<t(n)>}}} \tag{39}$$

We can enlarge each element of vector (21) at least $k \cdot 2^{t(n)}$ by the repeated addition previously mentioned. We then subtract this vector from vector (39), which implies $C_H(t(n))$ as follows.

$$C_H(t(n)) = \underbrace{(H_{t(n)}(1), \cdots, H_{t(n)}(k \cdot 2^{t(n)}), H_{t(n)}(0), \cdots)}_{\text{subsection of length } d^{<T(n)+1>}}{}^{\frac{d^{<T(n)+1>}}{d^{<t(n)>}}} \tag{40}$$

In (40), $H_{t(n)}(i)$ denotes $(\overbrace{\underbrace{ii\cdots i}_{2^{t(n)}}\underbrace{ii\cdots i}_{2^{t(n)}}\cdots\underbrace{ii\cdots i}_{2^{t(n)}}}^{2^{T(n)}}00\cdots0)^{\frac{d^{<T(n)+2>}}{(d^{<T(n)+1>})^2}}$ and $H_{t(n)}(0)$ is called a *dummy section*.

More natural way of expressing the head position is, as in [6], to use a 0/1 vector with holding exactly one 1 at the position the head is placed. Our current approach is quite different; we explicitly write the head position in integer and that integer is *repeated* a lot of times to be adjusted to tape vectors. The major reason is that our vector machine lacks the ability of "shifting operation". One can feel that if we use the conventional approach then we have to "shift" the position of 1 to the left or to the right according to the motion of M's head. However, it is quite unlikely for our RS-VM to be able to "shift" something, even by one position.

3.5 Constructing all configurations

We have already completed the construction of $C_H(t(n))$ which have all state/head-positions and $C_T(t(n))$ which have all tapes of length $2^{t(n)}$. At stage $t(n)$ we combine those two vectors to get all configurations. It should be noted that both are of very similar forms. For example, all divisions in each vector are completely the same. So, the idea is to make a direct product of $d^{<t(n)>}$ sections of $C_H(t(n))$ and $\frac{d^{<t(n)+1>}}{d^{<t(n)>}}$ sections of $C_T(t(n))$ to generate all combinations of tapes and state/head-positions. In more detail, we first make the following vector of length $d^{<T(n)+2>}$.

$$C_H(t(n)+1) = (H_{t(n)}(1)^{\frac{d^{<t(n)+1>}}{d^{<t(n)>}}}, H_{t(n)}(2)^{\frac{d^{<t(n)+1>}}{d^{<t(n)>}}}, \cdots)^{\frac{d^{<T(n)+2>}}{(d^{<T(n)+1>})^2}} \tag{41}$$

Now let us take a look at $C_H(t(n)+1)$ and $C_T(t(n)+1) = C_T(t(n))$ (thus the tape vector does not change from the proceeding stage) written in line-up as follows.

$$\begin{aligned} C_H(t(n)+1) &= H_{t(n)}(1)H_{t(n)}(1) \cdots \quad H_{t(n)}(1) \quad H_{t(n)}(2) \cdots\cdots \\ C_T(t(n)+1) &= S_{t(n)}(0)\, S_{t(n)}(1) \cdots S_{t(n)}(2^{2^{t(n)}} - 1) S_{t(n)}(0) \cdots\cdots \end{aligned} \tag{42}$$

One can see all the configurations appearing in these two vectors.

To construct vector (41), we can make use of the technique described in 3.3.2. We make $\rightarrow C_H(t(n))$ and $\downarrow C_H(t(n))$ and we regard them as matrix of $d^{<T(n)+2>}$ columns $\times$ $\frac{d^{<T(n)+3>}}{d^{<T(n)+2>}}$ rows. We again make the vector Z having non-zero value on the diagonal sticks. Note that $E(t(n))$ in (29) has only 1's in its important portion. Exactly as we did in 3.3.2, we make $\tilde{C}_H(t(n)+1)$ of length $d^{<T(n)+3>}$(=length of $\rightarrow C_T(t(n))$. One can verify that $C_H(t(n)+1)$ is obtained by *folding* $\tilde{C}_H(t(n)+1)$. Again note that sequences of configurations are represented not by a single vector but by two vectors $C_T(h)$ and $C_H(h)$ for $h > t(n)$ with the assumption that those two vectors have a natural one-to-one correspondence between their sections(the first section of $C_T(h)$ and the first of $C_H(h)$, the second one of $C_T(h)$ and the second one of $C_H(h)$, and so on).

3.6 Constructing sequences of configurations

Beginning with the vector which have all the configurations of TM M (actually the "two vectors", but we sometimes regard them as if a single vector), we construct all the sequences of configurations for 2steps, 4steps, and so on, until $2^{t(n)}$ steps. Suppose that stage $h(> t(n))$ has been just finished. The vectors $C_T(h)$ and $C_H(h)$ for $h > t(n)$ have exactly the same structure as $C_T(h)$ and $C_H(h)$ for $h < t(n)$. Hence we do not have to change our procedure given in 3.3.2 to construct $C_T(t(n)+2), C_H(t(n)+2), C_T(t(n)+3), C_H(t(n)+3), \cdots, C_T(2t(n)), C_H(2t(n))$ which contain all the desired sequences of configurations.

3.7 Constructing shifted sequences

Recall that what the vector machine V is now doing is to find (if exists) an accepting configuration sequence of TM M. In order for the sequence $c_0c_1\cdots c_{2^{t(n)}}$ of configurations to be an accepting sequence, first of all, c_i must be a proper successor of c_{i-1} for all i. To make this test possible, we introduce vectors TS and HS those are the same as T and H, respectively, but the first $2^{T(n)}$ bits in each subsection are shifted $2^{t(n)}$ bits(the length of a single configuration) cyclically to the right. We start with $C_{TS}(t(n)+1) := C_T(t(n)+1)$ and $C_{HS}(t(n)+1) := C_H(t(n)+1)$. Then, we construct $C_{TS}(t(n)+2), C_{HS}(t(n)+2)$, $\cdots$,$C_{TS}(2t(n)), C_{HS}(2t(n))$ using essentially the same technique previously mentioned. We again need four vectors A, B, D and E. A, B and D are completely the same as before. E is slightly different as follows, which we denote by E_S.

$$E_S(h) = (\overbrace{\underbrace{2^{2^{t(n)}}, 1^{2^{h-1}}, 2^{2^{h-1}}, \cdots, 1^{2^{h-1}}, 2^{2^{h-1}-2^{t(n)}}}_{2^{T(n)}}, 2, 2, \cdots, 2}^{d^{<T(n)+1>}})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}} \tag{43}$$

$E_S(h)$ is $E(h)$ whose first $2^{T(n)}$ bits in each subsection is shifted $2^{t(n)}$ bits cyclically to the right. One can see that $E_S(t(n)+1)$ is obtained from $E(t(n)+1)$ by switching 0's and 1's in first $2^{T(n)}$ bits of each subsection. In general, $E_S(h)$ for $h > t(n)+1$ is obtained as follows. Like (36), we obtain

$$(\underbrace{\overbrace{(1, 2, 3, \cdots, 2^{t(n)}, \cdots, 2^h)^{2^{T(n)-h}}}^{2^{T(n)}}, 0, 0, \cdots, 0}_{d^{<T(n)+1>}})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}. \tag{44}$$

As in (18), we change all elements larger than $2^{t(n)}$ into 0. Then, by adding (21), we get the following vector, which we call $F(h)$.

$$F(h) = (\underbrace{(\overbrace{\underbrace{1, 1, \cdots, 1}_{2^{t(n)}}, 0, 0, \cdots, 0}^{2^h})^{2^{T(n)-h}}, 2, 2, \cdots, 2}_{2^{T(n)}})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}} \tag{45}$$

We want to change $E(h)$, 1 into 2 and 2 into 1 only where the element of $F(h)$ is 1. This can be done as follows: Let $2F(h) := F(h)+F(h)$, $G_1(h) := 2F(h) - E(h)$ and $G_2(h) := F(h) - G_1(h)$,

$$E_S(h) := E(h) + G_1(h) - G_2(h) \tag{46}$$

is the answer. As an elementary example, if $2^{t(n)} = 2$ and $2^h = 8$ then those vectors look like:

$$E(h) = \cdots 11111111222222221111111122222222\cdots \tag{47}$$
$$F(h) = \cdots 11000000110000001100000011000000\cdots \tag{48}$$
$$E_S(h) = \cdots 22111111112222222211111111222222\cdots \tag{49}$$

3.8 Choosing proper sequences

Now we have all sequences of configurations $C_T(2t(n))$ and $C_H(2t(n))$ and their shifted version $C_{TS}(2t(n))$ and $C_{HS}(2t(n))$, which are in short denoted by T, H, TS and HS, respectively. What we are going to do is to test whether or not an accepting sequence exists. The accepting sequence requires, for example, (i)that its first configuration correctly contains M's input, (ii)that all changes of configurations in the sequence are proper and (iii)that the final configuration of sequence includes an accepting state. We introduce vectors $INIT$, SUC, FIN of length $d^{<T(n)+2>}$. In the rest of the proof, we regard a vectors of length $d^{<T(n)+2>}$ as a matrix of $d^{<T(n)+1>}$ rows $\times$ $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}$ columns, i.e.,horizontally long matrix; instead of the vertically long ones so far. A single row corresponds to a single section. Note that if we *contract* this matrix, then a single row, composed of $\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}$ elements, is compressed to a single element. Thus we get a column vector of length $d^{<T(n)+1>}$ whose element has a non-zero element of each row of the original matrix.

INIT: First we construct a (column) vector $INIT$ of length $d^{<T(n)+1>}$. This vector represents whether each row of T has a proper input string. As with (34), we get

$$(\underbrace{1, 2, 3, \cdots, 2^{t(n)}, 0, 0, \cdots, 0}_{d^{<T(n)+1>}})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}} \tag{50}$$

By turning all non-zero elements to 1 (see (19) for how to do so), we get vector

$$(\underbrace{\overbrace{1, 1, \cdots, 1}^{d^{<T(n)+1>}}, 0, 0, \cdots, 0}_{2^{t(n)}})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}}. \tag{51}$$

We regard this vector, say T_{INIT}, as having the following structure:

$$T_{INIT} = (\overbrace{\underbrace{11\cdots1}_{2^{t(n)}}\underbrace{00\cdots0}_{2^{t(n)}}\cdots\underbrace{00\cdots0}_{2^{t(n)}}\ 00\cdots0}^{\text{subsection of length } d^{<T(n)+1>}})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}} \tag{52}$$

By placing T_{INIT} over T and H, one can see that the portion of T_{INIT} having value 1 corresponds to the initial configuration of T and H. Recall that we have a special vector IN. When the input string is $i_1i_2\cdots i_n \in \{0,1\}^*$, $IN[1]$ holds i_1, $IN[2]$ holds i_2, and so on. $IN[n+1], IN[n+2], \cdots$ hold a large number (≥ 2). We add (13) to IN to cut it at the $(d^{<T(n)+1>})$th element. Changing all elements larger than 1 into 0, we obtain

$$(\underbrace{\overbrace{i_1i_2\cdots i_n00\cdots0}^{2^{t(n)}}00\cdots0}_{d^{<T(n)+1>}}) \tag{53}$$

Repeating this vector implies

$$(\overbrace{\underbrace{i_1i_2\cdots i_n}_{2^{t(n)}}0\cdots0\underbrace{00\cdots0}_{2^{t(n)}}\cdots\underbrace{00\cdots0}_{2^{t(n)}}\ 00\cdots0}^{\text{subsection of length } d^{<T(n)+1>}})^{\frac{d^{<T(n)+2>}}{d^{<T(n)+1>}}} \tag{54}$$

We make the vector which has value 1 at the position where (i)(52) has value 1 and (ii)T and (54) does not coincide. $INIT$ is obtained by *contracting* it. Verify that $INIT[i] = 1$ iff the sequence of T's ith row contains incorrect input symbols.

FIN: Next we construct a column vector FIN of length $d^{<T(n)+1>}$. This vector represents which sequence ends in an accepting configuration. We first make ($\underbrace{2^{T(n)}-1, 2^{T(n)}-1, \cdots, 2^{T(n)}-1}_{d^{<T(n)+2>}}$) and ($\underbrace{2^{t(n)}, 2^{t(n)}, \cdots, 2^{t(n)}}_{d^{<T(n)+2>}}$) like (15), and subtract the latter from the former. The result is

$$(\underbrace{(2^{T(n)}-2^{t(n)}-1), (2^{T(n)}-2^{t(n)}-1), \cdots, (2^{T(n)}-2^{t(n)}-1)}_{d^{<T(n)+2>}}). \tag{55}$$

Subtracting (55) from (34), and turning all non-zero elements to 1, we get

$$(\overbrace{\underbrace{0,0,\cdots,0}_{2^{T(n)}},\underbrace{1,1,\cdots,1}_{2^{t(n)}},0,0,\cdots,0}^{d^{<T(n)+1>}})^{d^{<T(n)+2>}_{d^{<T(n)+1>}}}. \quad (56)$$

One can see that the portion having value 1 corresponds to the final ($2^{t(n)}$th) configuration of both T and H. We test if elements of H in the above portion represent an accepting state. It may be omitted how to construct the vector of the same length as (56) which has value 1 only where this test fails. *Contracting* this vector we get FIN.

SUC: Now we check if each sequence of configurations by T and H is proper, i.e.,if all changes from every configurations to their right-door ones in the sequence follow the state transition function of TM M. To do so, we change every configuration sequence $c_{2^{t(n)}-1}\, c_0\, c_1 \cdots c_{2^{t(n)}-2}$ in TS and HS into $c'_{2^{t(n)}-1}\, c'_0\, c'_1 \cdots c'_{2^{t(n)}-2}$, where c'_i is the one-step-after configuration of c_i determined by the transition function. We compare $c'_{2^{t(n)}-1}\, c'_0\, c'_1 \cdots c'_{2^{t(n)}-2}$ with $c_0 c_1 c_2 \cdots c_{2^{t(n)}-1}$. If $c'_0 = c_1$, $c'_1 = c_2$, $\cdots$, $c'_{2^{t(n)}-2} = c_{2^{t(n)}-1}$ then we can decide that the sequence $c_0\ c_1\ c_2\ \cdots\ c_{2^{t(n)}-1}$ of configurations is proper. It is not trivial, however, how to carry out this modification in parallel. The idea is this: We first extract all c_i's that are associated with state 1 and symbol 0 under the head and modify them in parallel according to the state transition function. Then the same procedure is repeated for state 1 and symbol 1, state 2 and symbol 0 and so on.

The following procedure is repeated for state 1, then for state 2 and so on. Suppose that we are now carrying out the procedure for state j. Recall that state j is represented by one of the values from $(j-1)\cdot 2^{t(n)}+1$ to $j\cdot 2^{t(n)}$. We first change all elements having values outside this range into 0. Then the same value $(j-1)\cdot 2^{t(n)}$ is subtracted from each element of this vector. We call the resulting vector $HS(j)$. Note that $HS(j)$ now contains only the information of head-positions (a value from 1 to $2^{t(n)}$) only where M's state is j. Now we want to compute the next head-position and the next state, for which this $HS(j)$ still does not seem enough. Note that the head position is represented by like $\cdots(33333333)\cdots$ in $HS(j)$ (the head is on the third cell), but what we really want is $\cdots(00100000)\cdots$, merely by which we can, for example, pick out a tape symbol under the head. Let $P(j)$ be a vector of the latter style, which is obtained as follows. We first make the vector like (44).

$$(\overbrace{\underbrace{1,2,3,4,\cdots,2^{t(n)})2^{t(n)}}_{\text{subsection of } d^{<T(n)+1>}},1,1,\cdots,1}^{d^{<T(n)+1>}})^{d^{<T(n)+2>}_{d^{<T(n)+1>}}} \quad (57)$$

One can see that $P(i)$ is the vector which has value 1 in the positions where the value of vector (57) and that of vector $HS(j)$ coincide, i.e.,in the positions where the head exists. We also need 0/1 vectors $P(j,0)$ and $P(j,1)$ such that $P(j,0)[i] = 1(P(j,1)[i] = 1$, respectively) iff (i)$P(j)[i] = 1$, i.e.,the head exists at that position and (ii)$TS(j)[i] = 1$, i.e.,the tape symbol under the head is 0(1, respectively). Clearly $P(j) = P(j,0) + P(j,1)$.

We then introduce $NTS(j)$. $NTS(j)$ is exactly as $TS(j)$ but is constructed for the "one-step-after" sequences of configurations. All we have to do is to change only one position of $TS(j)$ where M's head exists($P(i) = 1$). For example, if some portion of $P(j)$ is $\cdots(00100000)\cdots$ and the same portion of $TS(j)$ is $\cdots(10010101)\cdots$, then $NTS(j)$ should be $\cdots(10x_010101)\cdots$, where x_0 is the symbol written by M on state j reading 0. Also let x_1 be the symbol written by M reading 1. Then

$$NTS(j)[i] = \begin{cases} TS(j)[i] & \text{if } P(j)[i] = 0 \\ x_0 & \text{if } P(j,0)[i] = 1 \\ x_1 & \text{if } P(j,1)[i] = 1. \end{cases}$$

How to construct it may be omitted.

We need one more vector $NHS(j)$. $NHS(j)$ is again similar to $HS(j)$. Suppose that M moves its head by $y_0(y_1$, respectively) to the right and M's next state is m when it reads tape symbol 0(1,respectively). (y_0 and y_1 are $+1$ or -1).

$$NHS(j)[i] = \begin{cases} HS(j)[i] + y_0 + m\cdot 2^{t(n)} & \text{if } P(j,0)[i] = 1 \\ HS(j)[i] + y_1 + m\cdot 2^{t(n)} & \text{if } P(j,1)[i] = 1 \\ 0 & \text{otherwise} \end{cases}$$

(Note that $NHS(j)$ has a non-zero value in a single position of each configuration, i.e., where the head exists. That is different from $HS(j)$ but does not cause any problem as shown in a moment.)

After completing above procedure for all k states, we sum the vectors up as follows.

$$NTS = NTS(1) + NTS(2) + \cdots + NTS(k). \quad (58)$$

$$NHS = NHS(1) + NHS(2) + \cdots + NHS(k). \quad (59)$$

Now we have NTS and NHS as the vectors containing the configuration of "one-step-after". They are compared with T and H, respectively. We make a 0/1 vector, SUC_T, of length $d^{<T(n)+2>}$. SUC_T has value 1 in the positions where NTS and T does not coincide except the first $2^{t(n)}$ bits of each sequence.(Recall that these positions have been already checked by $INIT$.) Formally, $SUC_T[i]$ is set to 1 iff (i)$NTS[i] \neq T[i]$ and (ii) $T_{INIT}[i] = 0$(see (52)) and (iii)$T[i] \neq 2$(dummy elements). Similarly, we also make a 0/1 vector, SUC_H, of length $d^{<T(n)+2>}$ such that $SUC_H[i] = 1$ iff (i)$NHS[i] \neq 0$ (we should pay attention only to the positions the head exists) and (ii)$NHS[i] \neq H[i]$ and (iii)$T_{INIT}[i] = 0$. Let $S\tilde{U}C := SUC_T + SUC_H$. If $S\tilde{U}C$ holds a 1 in some row, then the row does not proper as a sequence of configurations. *Contracting* $S\tilde{U}C$, we get the column vector SUC of length $d^{<T(n)+1>}$.

We add $INIT$, SUC, FIN into a single vector and change all 0 into 1 and all 1 into 0. We call this vector ACC. The element of ACC is 1 iff (i)the beginning of the sequence coincides with the input string ($INIT$) and (ii)changes of configurations are proper(SUC) and (iii)the sequence ends with an accepting state(FIN). Thus M accepts the input string if this ACC contains at least one 1. To figure this out, we *contract* this vector until it become of length 2. We finally get one of (0,0), (0,1), (1,0), (1,1). The last three vectors can be modified to (1,1) by *repeating* followed by *contracting*. (The first one is still (0,0).) Thus VM knows whether M accept the input string by checking if the first element of the vector is 1.

3.9 Extension to alternating TMs

So far we have only shown that $\mathrm{VM}(d_s(m), poly) \supseteq \mathrm{DTIME}(2^{poly})$. Our final goal is to show that $\mathrm{VM}(d_s(m), poly) \supseteq \mathrm{ATIMALT}(2^{poly}, poly)$. We have constructed the configuration sequences of $2^{t(n)}$ steps, which are extend to of $2^{t(n)+A(n)}$ steps, where $A(n)$ is the number of alternations. Namely, we make all configuration sequences of $2^{t(n)+A(n)}$ steps and each configuration is represented by $2^{t(n)}$ bits as before. (The total number of elements for such a sequence is $2^{t(n)} \times 2^{t(n)+A(n)} = 2^{T(n)+A(n)}$.) We make four vectors T, TS, H, HS of length $d^{<T(n)+A(n)+2>}$ based on the same idea as before. Note that $2^{T(n)+A(n)}$ can be written as

$$2^{T(n)}\cdot 2^{A(n)} = 2^{T(n)} + 2^{T(n)} + 2^{T(n)}\cdot 2 + 2^{T(n)}\cdot 4 + \cdots + 2^{T(n)}\cdot 2^{A(n)-1}. \quad (60)$$

It should be noted that we are still assuming, as before, that the alternating TM M simulated by the VM stops within $t(n)$ steps. Such a TM begins its operation in, without loss of generality, an $\exists$-state, then switches its state from an $\exists$-state to a $\forall$-state, then from a $\forall$-state to an $\exists$-state and so on. M can, of course, spend different number of steps(probably much less than $2^{t(n)}$) in the first $\exists$-states, in the second $\forall$-states and so on. However, when we consider the sequence of configurations for that kind of M's operation, it is enough to consider such sequences that the state changes from $\exists$-states to $\forall$-states (or vice versa) only at fixed points. Above (60) shows those fixed points, i.e.,they are at $(2^{T(n)})$th step, $(2^{T(n)} + 2^{T(n)} = 2^{T(n)+1})$th step, $(2^{T(n)} + 2^{T(n)} + 2^{T(n)}\cdot 2 = 2^{T(n)+2})$th step, and so on. (The reason: We allow each configuration to appear repeatedly. Namely, the "proper change" now includes "does not change".) One can see later how those fixed points are adopted.

Similarly as before, we construct $C_T(T(n)+A(n))$, $C_H(T(n)+A(n))$, $C_{TS}(T(n)+A(n))$, $C_{HS}(T(n)+A(n))$ of length $d^{<T(n)+A(n)+2>}$, which are denoted by T, H, TS and HS, respectively. We again construct three vectors $INIT$, SUC and FIN of length $d^{<T(n)+A(n)+1>}$. $INIT$ is a 0/2 vector; $INIT[i] = 2$ shows that the first configuration of the ith row (of T and H) does not coincide with the input string. SUC is a 0/2 vector; $SUC[i] = 2$ shows that the configuration sequence of the ith row include improper changes. FIN is a 0/1 vector and $FIN[i] = 1$ shows that the final configuration of the sequence is accepting. (One can see later why we introduce the new value 2 other than 0 and 1.)

How to contract $INIT$ and FIN may be omitted since they are almost the same as in 3.8. As for SUC, we should be careful since the TM M is now nondeterministic. Suppose that l is the maximum number of nondeterministic choices for each combination of state and tape symbol. Instead of the single $S\tilde{U}C$ in 3.8, we construct $S\tilde{U}C_1$, $S\tilde{U}C_2$, $\cdots$, $S\tilde{U}C_l$, where $S\tilde{U}C_i$ is obtained exactly as $S\tilde{U}C$ by assuming that M "always" choose the ith move among its nondeterministic choices. Each vector has value 1 in the position where changes of configuration is not proper. We also need

$S\tilde{U}C_E$ that corresponds to the repetition of the same configuration mentioned above. Furthermore we have to consider whether the sequence changes the tape of its state at the fixed points. Although details are omitted, it is not hard to construct another vector $S\tilde{U}C_A$ which has value 1 in the position where H holds a different type of state from what is supposed to be. Now we compute $S\tilde{U}C'$ as

$$S\tilde{U}C' = (S\tilde{U}C_1 \wedge \cdots \wedge S\tilde{U}C_l \wedge S\tilde{U}C_E) \vee S\tilde{U}C_A, \qquad (61)$$

where $\wedge(\vee)$ is an element-wise logical-and(logical-or) operation. $S\tilde{U}C'$ holds value 1 iff the sequence contains improper changes or improper states. Changing all 1 into 2 of $S\tilde{U}C'$, we get $S\tilde{U}C$.

We compute $(FIN-INIT-SUC)+INIT+SUC$, denoted by $ACC(d^{<T(n)+A(n)+1>})$, which is regarded as a column vector of length $d^{<T(n)+A(n)+1>}$. $ACC(d^{<T(n)+A(n)+1>})[i] = 1$ means that the sequence in the ith row starts with the correct input string and that all the changes are proper and that the sequence is ended in some accepting state. $ACC(d^{<T(n)+A(n)+1>})[i] = 0$ means that the sequence in the ith row starts with the given input string and that all the changes are proper but that the sequence is ended in some rejecting state. $ACC(d^{<T(n)+A(n)+1>})[i] = 2$ shows the other cases, i.e.,changes of configurations in the sequence do not follow the transition function or the input is not correct or the type of state changes at other than the fixed points.

We again change the form of matrix; we regard vectors of length $d^{<T(n)+A(n)+2>}$ (T, H etc.) as a matrix of $d^{<T(n)+A(n)+1>}$ rows and $\frac{d^{<T(n)+A(n)+2>}}{d^{<T(n)+A(n)+1>}}$ columns. It will be beneficial to make the following observation on this matrix: Recall that a single row of this matrix corresponds to a single sequence of configurations, which appears repeatedly along the row. Fig.6 shows the left most occurrence (the left most $2^{T(n)+A(n)}$ columns) of this sequence. These $2^{T(n)+A(n)}$ columns are divided into $A(n)+1$ portions following (60), i.e.,into portions of $2^{T(n)}$, $2^{T(n)}$, $2^{T(n)+1}$, $2^{T(n)+2}$, $\cdots$ and $2^{T(n)+A(n)-1}$ columns. Those are denoted by $PT(0)$, $PT(1)$, $PT(2)$, $\cdots$, $PT(A(n))$ in Fig.5. Recall that TM M is in $\exists$-states in $PT(0)$, $\forall$-states in $PT(1)$, $\cdots$, $\forall$-states in $PT(A(n))$. One can see that our construction of T and H guarantees the following structure on this matrix.

1. In $PT(A(n))$(consisting of $2^{T(n)+A(n)-1}$ columns), consecutive $\frac{d^{<T(n)+A(n)+1>}}{d^{<T(n)+A(n)>}}$ rows contains all the different configuration sequences represented by $2^{T(n)+A(n)-1}$ bits (with a repetition).

2. In $PT(A(n)-1)$, consecutive $\frac{d^{<T(n)+A(n)+1>}}{d^{<T(n)+A(n)>}}$ rows are completely the same, which we put together as a ***group***. Consecutive $\frac{d^{<T(n)+A(n)>}}{d^{<T(n)+A(n)-1>}}$ groups cover all the sequences represented by $2^{T(n)+A(n)-2}$ bits.

3. In $PT(A(n)-2)$, consecutive $\frac{d^{<T(n)+A(n)>}}{d^{<T(n)+A(n)-1>}}$ ***groups***(defined above) are completely the same, which we put together as a *new group*. Again consecutive $\frac{d^{<T(n)+A(n)-1>}}{d^{<T(n)+A(n)-2>}}$ new groups cover all the sequences represented by $2^{T(n)+A(n)-3}$ bits. Note that single new group contains all the different (old) groups. All other portions are similar.

We can associate this matrix with a computation tree of alternating TMs by regarding each group as a node. One should recall the following basic rules on the computation tree of alternating TMs. For M to accept the input, "all" $\forall$-branches from a particular node must lead to accepting nodes, and "at least one" of $\exists$-branches from a particular node must lead to an accepting node. We can decide if M actually does so by decreasing the height of the tree one by one. First, all leaves emitted from each single node, say v, on the next level from the bottom are put together and we can decide if v is accepting or not by the rule above. Note that the leaves themselves are accepting iff the corresponding sequence of configurations is accepting.

A crucial point is that this process of lowering the tree one by one can be simulated simply by contracting vector ACC. We first consider $ACC(d^{<T(n)+A(n)+1>})$ which corresponds to, in a sense, all leaves mentioned above. If we *contract* it, then we can put together all the consecutive $\frac{d^{<T(n)+A(n)+1>}}{d^{<T(n)+A(n)>}}$ rows(leaves of $PT(A(n))$) into one. Then if we *contract* the resulting vector then we can put together all the consecutive $d^{<T(n)+A(n)>}$ groups of $PT(A(n)-1)$, and so on. One can now see how we determined the fixed points only where TM M can change $\exists$-states from $\forall$-states(or vice versa). More formally, we get $ACC(d^{<T(n)+j-1>})$ from $ACC(d^{<T(n)+j>})$ as follows. Suppose that $PT(j-1)$ consists of $\exists$-states. As mentioned above, we put every $\frac{d^{<T(n)+j>}}{d^{<T(n)+j-1>}}$ consecutive elements of $ACC(d^{<T(n)+j-1>})$ together into a single element. Let G denotes a single set of these $\frac{d^{<T(n)+j>}}{d^{<T(n)+j-1>}}$ consecutive elements. Following the rule of alternating TMs:

1. If all elements of G are 2 then we compress G to a single element 2(the sequence is not proper).

2. If G includes at least one 1 then we compress G to 1(accepting sequence).

3. If all elements of G are 2 or 0 and at least one element is 0 then we compress G to 0(rejecting sequence),

We call above operation $\exists$-*contract*, which is realized as follows.

$\exists$-*contract*: We change all 2 of $ACC(d^{<T(n)+j>})$ into 0, which gives, say, $A\tilde{C}C_1$. *Contracting* $A\tilde{C}C_1$, we get $ACC_1(d^{<T(n)+j-1>})$. We change all 0/1 elements of $ACC(d^{<T(n)+j>})$ into 0, which gives us, say, $A\tilde{C}C_2$. We then switch 0 and 2 of $A\tilde{C}C_2$, and *contract* the resulting vector, which gives $A\tilde{C}C_2'$. We finally switch 2 and 0 of $A\tilde{C}C_2'$, Which implies $ACC_2(d^{<T(n)+j-1>})$. Now

$$ACC(d^{<T(n)+j>}) := ACC_1(d^{<T(n)+j>}) + ACC_2(d^{<T(n)+j>}) \qquad (62)$$

We should mention why ACC includes value 2 other than 0 and 1. We regard the matrix as a computation tree but the matrix contains many improper sequences. We should carry out the contraction above so that such improper sequences will not give any effect on the decision of accepting or rejecting. It should be noticed that how to maintain the value 2 is different between the $\exists$-contract and the following $\forall$-contract.

$\forall$-contract, is given as follows:

1. If all elements of G are 2 then we compress G to a single element 2(the sequence is not proper).

2. If all elements of G are 2 or 1 and at least one element is 1 then we compress G to 1(accepting sequence).

3. If G includes at least one 0, then we compress G to 0(rejecting sequence),

We omit its realization because it is similar to $\exists$-contract.

Beginning with $ACC(d^{<T(n)+A(n)+1>})$, we apply $\forall$-*contract*, $\exists$-*contract*, $\forall$-*contract*, $\exists$-*contract*, and so on ,until $ACC(d^{<T(n)+1>})$. Then we change all elements of value 2 in $ACC(d^{<T(n)+1>})$ into 0. Our procedure after that is completely the same as 3.8. ∎

References

[1] A. Bertoni, G. Mauri, and N. Sabadini,"A characterization of the class of functions computable in polynomial time on random access machines," *Proc. 13th ACM Symp. on Theory of Computing*,pp.168-176, 1981.

[2] A. Chandra, D. Kozen, and L. Stockmeyer,"Alternation," *J. Assoc. Comput. Mach.*, vol.28, pp.114-133, 1981.

[3] J. Hartmanis and J. Simon, "On the power of multiplication in random access machines," *Proc. IEEE Symp. on Switching and Automata Theory*, pp.13-23, 1974.

[4] K. Iwama, "A canonical form of vector machines," *Technical Report COMP88-22,Institute of Electronics and Communication Engineers of Japan*, 1988

[5] K. Iwama, "Exponential Speedup by Vector Operations," *Bulletin of the Institute of Computer Sciences, Kyoto Sangyou Univ.*, vol. 5, 1988.

[6] V. Pratt and L. Stockmeyer, "A characterization of the power of vector machines," *J. Comput. Syst. Sci.*, vol. 12, pp. 198-221, 1976.

[7] J. Simon, "Division is good," *Proc. 20th IEEE Symp. on Foundations of Computer Science*, pp. 411-420, 1979.

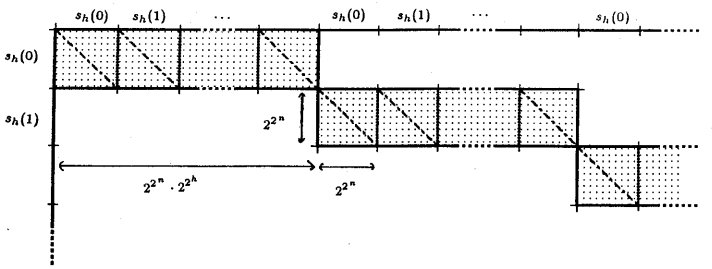

Fig.1

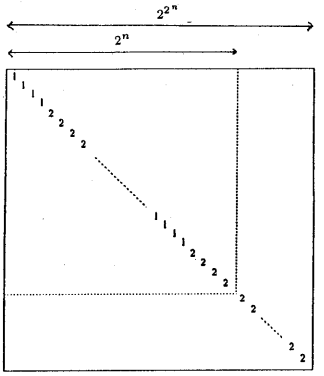

Fig.2

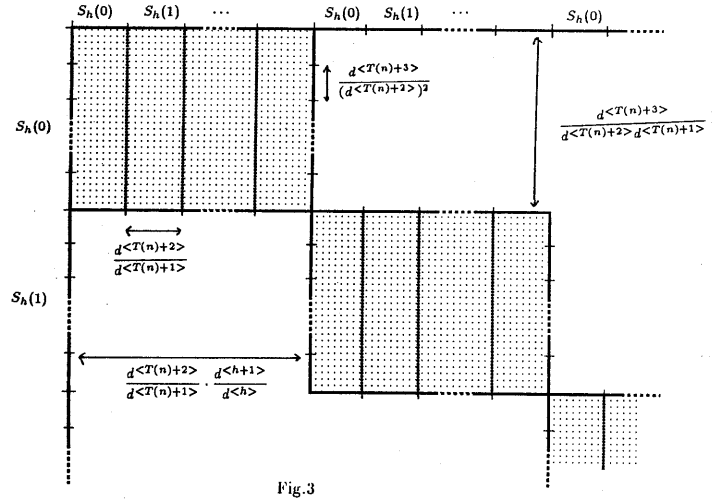

Fig.3

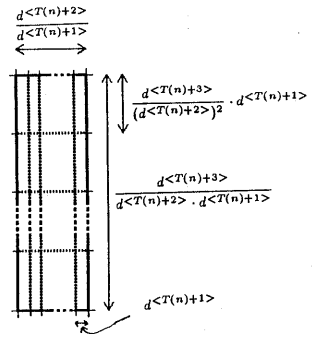

Fig.4

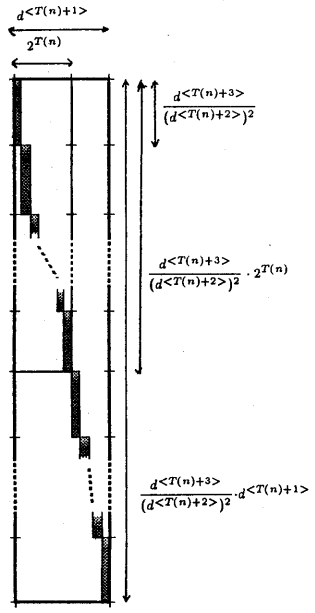

Fig.5

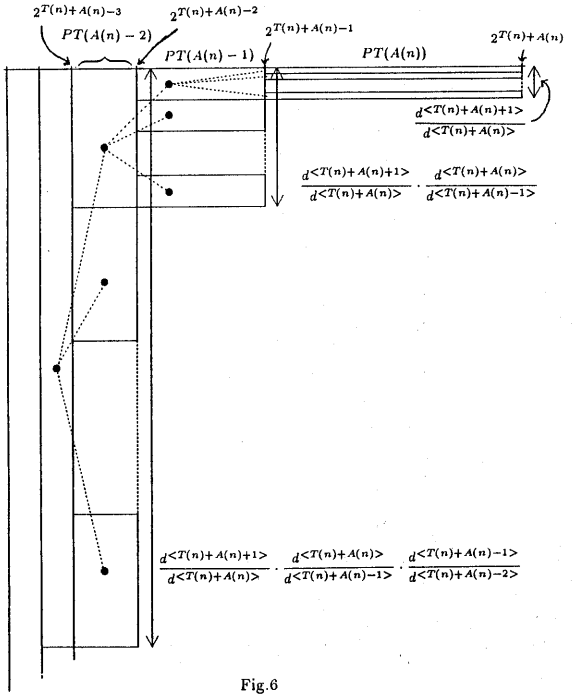

Fig.6