International Mathematical Forum, 1, 2006, no. 26, 1273 - 1284

Plemwanal: A Communicating Computing Mathematics Generator Type

Fevzi Ünlü

Department of Mathematics, Yasar University,
Bornova, Izmir, Turkey

e-mail: fevzi.unlu@yasar.edu.tr and fevziunlu@ttnet.net.tr

Abstract

[1] Has introduced the notion of *I@I*. [2-10] have developed, studied and introduced the fundamentals of computing mathematics on 1) Prüfer's type *I@I* clusters; 2) the notion of knowledge based object - *KBO*; 3) integration and derivation of *KBO*; 4) us-crop; 5) us-culture; 6) tuze-channeled KBO for DCP; 7) linear-circular convoluted us-crop *I@I KBO* cluster generating us-crop; 8) us-crop based compact plasma memory; and 9) us-crop based plemal. Discrete mathematical backgrounds and point set topology backgrounds can be found in [11, 12]. Combinatorial mathematical backgrounds can be found in [13]. This paper develops, studies and introduces some fundamentals of a formal language type that generates a communicating computer mathematics type virtual machine KBO and its remote programming style on us-crop based compact plasma memory by ten definitions, two corollary and one theorem. It is called as *plemwanal* in this paper. Author claims that plemwanal is a very special communicating computur mathematics type virtual machine KBO and its remote programming style generating language type. It is produced as an extension of *plemal* for obtaining an artificial intelligently communicating numerical words with antennae. These numerical words with antennae are communicating themselves by broadcasting and intercepting information in the execution time of its substructures. Eeach token numerical word with antennae in each substructure has wireless communication property with each other and with its environment. It is designed for generating remote, parallel, sequential, mobile, distributed, etc. type us-crop based compact I@I KBO cluster plasma memory programming in classical or quantum information processing environment. After plemwanal there is going to be a new

computing mathematics type. The author calls it as "Communicating Computer Mathematics."

Mathematics Subject Classification: 00A05, 00A08, 00A35, 00A71, 03F50, 44A55, 47A65, 68P20, 68Q10, 68Q65, 68Q85, 94A05, 94A16.

Keywords: KBO, us-crop I@I KBO cluster, us-crop I@I cluster memory type, formal grammar, formal language, remote programming style, virtual machine KBO, communicating computer mathematics.

1 Introduction

[1] has introduced the notion of *I@I*. [2-10] have developed, studied and introduced the fundamentals of computer mathematics on 1) Prüfer's type *I@I* clusters; 2) the notion of knowledge based object -*KBO*; 3) integration and derivation of *KBO*; 4) us-crop; 5) us-culture; 6) tuze-channeled KBO for DCP; 7) linear-circular convoluted us-crop *I@I KBO* cluster generating us-crop; 8) us-crop based compact plem memory; 9) us-crop based plemal. Discrete mathematical backgrounds and point set topology backgrounds can be found in [11, 12]. Combinatorial mathematical backgrounds can be found in [13]. This paper develops, studies and introduces some fundamentals of a formal language type that generates a communicating computer mathematics type virtual machine KBO and its remote programming style on us-crop based compact plem memory by ten definitions, two corollary and one theorem. It is called as *plemwanal* in this paper. Author claims that plemwanal is a very special communicating computer mathematics type virtual machine KBO and its remote programming style generating language type. It is produced as an extension of *plemal* for obtaining an artificial intelligently communicating numerical words with antennae. These numerical words with antennae are communicating themselves by broadcasting and intercepting information in the execution time of its substructures. Each token numerical word with antennae in each substructure has wireless communication property with each other and with its environment. It is designed for generating remote, parallel, sequential, mobile, distributed, etc. type us-crop based compact I@I KBO cluster plem memory programming in classical or quantum information processing environment. After plemwanal there is going to be a new computing mathematics type. The author calls it as "Communicating Computer Mathematics."

2 Preliminary Notes

[1-10] introduce the fundamentals of computing mathematics of data structures in/on us-crop and syntax, semantics and pragmatics of *plemal.* Those structures are currently adapted into *plemwanal* for current use and making it isomorph to *plemal.* In this section, therefore, the preliminary notions and tools which are necessary for generating *plemwanal* will be introduced. The needed discrete mathematical backgrounds and point set topology backgrounds can be found in [11, 12]. Combinatorial mathematical backgrounds can be found in [13].

2.1 Plem and Data Structures on Plem

Definition 1 Let 1) "*plem*" be a brief vocabulary for *programmable linear expending-contracting memory;* 2) "$\circlearrowleft$ " and "$\circlearrowright$ " be descriptions for *plem icons for representing u-crop and s-crop;* 3) "KBO" be a brief vocabulary for *Knowledge Based Object.* 4) "$\langle$" be a *broadcasting antenna icon*; 5) "$\rangle$" be an *intercepting antenna icon*; and 6) "ζ" be a formal KBO type; 7) "Z^+" be the set of *positive integers*; 8) "$\leftarrowtail$" be a *generalized assignment operator*; 9) "|" be a *logical OR operator* in the descriptive language(*in an α-language type*); 10) r_i be a *proper KBO register type that it is indexed by* $i \in Z^+$; 11) *plemwan* be a brief vocabulary for "*programmable linear expanding-contracting memory with antennae network.*" Then 1) $\langle\ \zeta\ \rangle$ is called a *KBO ζ with a pair of broadcasting and intercepting antennae;* 2) r_u is called an *u-plem register, if* $r_u = \circlearrowleft$*;* 3) r_s is called an *s-plem register, if* $r_s = \circlearrowright$*;* 4) r is called a *plem register*; if r takes the value of $r_u \mid r_s$. 5) $\langle r_1^{\diamond}, r_2^{\diamond}, r_3^{\diamond} : r_u \rangle = \langle r_3, r_2, r_1 : r_u \rangle$ is called an *u-plemwan token, if* $\langle r_3, r_2, r_1 : r_u \rangle = \langle\ \begin{matrix} r_3 & r_u & r_1 \\ & r_2 & \end{matrix}\ \rangle = \langle\ \begin{matrix} r_3 & \circlearrowleft & r_1 \\ & r_2 & \end{matrix}\ \rangle$. Where (i) $r_1 \leftarrowtail v_1$, $v_1 \in [p + 1] = \{ 0, 1, ..., p\}$, $p \in Z^+$; (ii) $r_2 \leftarrowtail v_2$, $v_2 \in [x + 1] = \{ 0, 1, ..., x\}$, $x \in \mathring{A}$, where Å is a letteral alphabet; (iii) $r_3 \leftarrowtail v_3$, $v_3 \in [m + 1] = \{ 0, 1, ..., m\}$, $m \in Z^+$. 6) $\langle r_1, r_2, r_3 : r_s \rangle$ is called an *s-plemwan token, if* $\langle r_1, r_2, r_3 : r_s \rangle = \langle\ \begin{matrix} r_1 & r_s & r_3 \\ & r_2 & \end{matrix}\ \rangle = \langle\ \begin{matrix} r_1 & \circlearrowright & r_3 \\ & r_2 & \end{matrix}\ \rangle$. Where (i) $r_1 \leftarrowtail v_1$, $v_1 \in [q + 1] = \{ 0, 1, ..., q\}$, $q \in Z^+$; (ii) $r_2 \leftarrowtail v_2$, $v_2 \in [y + 1] = \{ 0, 1, ..., y\}$, $y \in \ddot{A}$, where Ä is a letteral alphabet; (iii) $r_3 \leftarrowtail v_3$, $v_3 \in [n + 1] = \{ 0, 1, ..., n\}$, $n \in Z^+$;

Corollary 1 $r \leftarrowtail r_u \mid r_s$ implies $\langle r_1, r_2, r_3 : r\rangle \leftarrowtail \langle r_3, r_2, r_1 : r_u\rangle \mid \langle r_1, r_2, r_3 : r_s\rangle$.

Proof Let 1) r_u be an *u-plem register;* 2) r_s be an *s-plem register.* Then $r \leftarrowtail r_u \mid r_s$ implies that $\langle r_1, r_2, r_3 : r\rangle \leftarrowtail \langle r_3, r_2, r_1 : r_u\rangle \mid \langle r_1, r_2, r_3 : r_s\rangle$.

2.2 Plemwana: Alphabet of Plemwanal

Definition 2 Let "*FTD*" be a brief vocabulary for "*formal technology dependent*". 1) A *mod* p enforced(registered by an available formal technology), and m-ary negative u-crop I@I KBO cluster memory with a pair of broadcasting and intercepting antennae is called as *an m-ary negative FTD plem memory type with a pair of broadcasting and intercepting antennae.* It is represented by $\langle X_- \rangle = \langle p \circlearrowleft m \rangle \iff$ [-m; -0]$\langle X_- \rangle @ \langle X_+ \rangle$ $[+\phi; +\phi]$ *mod* p = array[-m ... -0] of ($\langle X_- \rangle$ *mod* p, $\langle X_+ \rangle$ *mod* ϕ). 2) A *mod* q enforced, and n-ary positive s-crop I@I KBO cluster memory is called *an m-ary positive plem memory* type *with a pair of broadcasting and intercepting antennae.* It is represented by $X_+ = \langle n \circlearrowleft q \rangle \iff [-\phi; -\phi]\langle X_- \rangle @ \langle X_- \rangle [+0; +n]$ *mod* q = array[+0 .. +n] of ($\langle X_- \rangle$ *mod* ϕ, $\langle X_+ \rangle$ *mod* q). 3) $\Omega = \{\langle p \circlearrowleft m \rangle, \langle n \circlearrowleft q \rangle\}$ is called a *plemwana* type KBO for p, q ≥ 2 and p, m, n, q $\in$ N. Where *plemwana* is a brief vocabulary for "*alphabet of programmable linearly-extending-contracting-plema memory with antenna networks.*"

2.3 Plemwanal

Definition 3 Let p, q ≥ 2; m, n, p and q $\in$ N and N be the set of natural numbers. Let $\Omega = \{ \langle m \circlearrowleft p \rangle, \langle q \circlearrowleft n \rangle \}$ be a *plemwana.* 1) $\hat{G} = \{ R_0 = \Omega^0 = \{ \varepsilon \}, R_1 : \Omega^1 = \Omega, R_2 = \Omega^t = \{ x \circ y = xy : x \in \Omega, y \in \Omega^{t-1}, t \geq 2, t \in N \} \}$ is called a recursive *formal grammar* o*f plemwanal* on the *plemwana* Ω under concatenation operator $\circ$. 2) $\Omega^+ = \cup_{t=1}^{\infty} \Omega^t = \cup_{t \in Z^+}^{\infty} \Omega^t$ is called *plemwana*$^+$. 3) $\Omega^* = \cup_{t=0}^{\infty} \Omega^t = \cup_{t \in N}^{\infty} \Omega^t$ is called *plemwana** 4) Each string element in *plemwana** is called a *plemwanal word.* 5) Any subset of a *plemwana** is called a *plemwanal. Plemwanal* is a new constructed vocabulary. It stands for the family of languages generated on a *plemwana* Ω by the recursive *plemwana grammar* $\hat{G}$. It includes empty subset ϕ, which is *empty plemwanal.*

Corollary 2 (a) Plemwanal is a family of formal language *on plemwana* $\Omega = \{ \langle m \circlearrowleft p \rangle, \langle q \circlearrowleft n \rangle \}$. (b) Each *plemwanal* is isomorph to *plemal on plema* $\Sigma = \{m \circlearrowleft p, q \circlearrowleft n \}$.

Proof (a) Plemwanal is a family of formal language due to its definition. (b) Each plemwanal is isomorph to a plemal because only the terminal symbols in their alphabet *plemwana* and *plema* are different from each other.

3 Plemwanal KBO Configurations

Definition 4 (a) Let x be a plemwan. $x \leftarrowtail \langle m \circlearrowleft p \rangle \mid \langle n \circlearrowleft q \rangle$; p, $p \geq 2$, p, q, n, m $\in$ N*;* Each element of x with different combinatorial values assigned for p, q, n, and m is called a *plemwan configuration.* (b) Let y be a plemwanal word. Each element of y $\in \Omega^*$ with different combinatorial values assigned

for p, q, n, and m is called a *plemwanal word configuration.* (c) Let z be a plemwanal L. Each element of $z \subseteq \Omega^*$ with different values assigned for p, q, n, and m is called a *plemwanal configuration.*

3.1 General Plemwanal KBO Structure Types

Definition 5 The followings are called general configuration KBO types:

01) $[-u; -0] = [-u, ..., -0]$, u ∈ N.

02) $[+0; +s] = [+0, ..., +s]$, s ∈ N.

03) $[\phi_{-u}; \phi_{-0}] = [\phi_{-u}, ..., \phi_{-0}] = -\Phi$, where -Φ is called KBO of negative emptiness, if $\Phi_{-i} \longleftarrow \Phi$ for i ∈ [-u, -0].

04) $[\phi_{+0}; \phi_{+s}] = [\phi_{+0}, ..., \phi_{+s}] = +\Phi$, where +Φ is called KBO of positive emptiness, if $\Phi_{+i} \longleftarrow \Phi$ for i ∈ [+0, +s].

05) u-crop = a set of spikes indexed by $[-u; -0] = [-u, ..., -0]$, u ∈ N.

06) s-crop = a set of spikes indexed by $[+0; +s] = [+0, ..., +s]$, s ∈ N.

07) $m\circlearrowleft p$ = u-crop token of the spike-values that they take their values in mod p, where $p \geq 2$, p, u ∈ N.

08) $q\circlearrowleft n$ = s-crop token of the spike-values that they take their values in mod q, where $q \geq 2$, q, u ∈ N.

09) us-crop = a set of spikes that they are indexed by $[-u; -0] \cup [+0; +s]$, where u, s ∈ N.

10) u-culture = a set of u-crops that they are indexed by $[-u_i; -0]$, where $u_i, _i \in$ N.

11) s-culture = a set of s-crop that they are indexed by $[+0; +s_j]$, where $s_j, _j \in$ N.

12) us-culture = a set of us-crops that they are indexed by $[-u_i; -0] \cup [+0; +s_j]$, where $u_i, s_j, _i, _j \in$ N.

13) plem $\leftharpoonup m\circlearrowleft p \mid q\circlearrowleft n$; *where p, q* ≥ 2, *p, q, n, m* ∈ N.

14) plema = $\{m\circlearrowleft p\ ,\ q\circlearrowleft n\}$; where *p, q* ≥ 2, *p, q, n, m* ∈ N.

15) plemal = a subset of plema*.

16) plemwan $\leftharpoonup \langle m \circlearrowleft p\rangle \mid \langle n\circlearrowleft q\rangle$; where *p, p* ≥ *2, p, q, n, m* ∈ N.

17) plemwana $\leftharpoonup \{\ \langle m \circlearrowleft p\rangle, \langle n\circlearrowleft q\rangle\}$; where *p, p* ≥ *2, p, q, n, m* ∈ N.

18) plemwanal = a subset of plemwana*.

Where * is *Kleene's Closure operator.* It produces all words(or strings) from a given alphabet.

3.2 Variable KBO Structure types

Definition 6 The followings are called variable-configuration KBO structure types.

1) δ = a character type of a | b | ... | x | y| z .

2) Δ = a character type of A | B | ... | X | Y| Z.
3) Σ : Plema.
4) Ω : Plemwana.
5) δ : the set of plemwanal-word variable.
6) Δ : the set of plemwanal variable.
7) L : formal language.

3.3 Constant KBO Structure Types

Definition 7 The followings are called constant-configuration KBO types.
1) Z^+ : The set of positive numbers.
2) N : The set of natural numbers.
3) [m] : The set of mod m numerals. That is [m] = {0, 1, 2, ..., m-1}.

3.4 Structured Construction KBO Types

Definition 8 The followings are called construction-configuration KBO structure types:
01) $\Omega^1 = \Omega$.
02) $\Omega^{n+1} = \{ x \circ y = xy : x \in \Omega, y \in \Omega^n \}$, where $xy = x \circ y$ denotes the concatenation of x and y.
03) Ω^+ = Plus closure of Ω .
04) Ω^* = Kleen's closure of Ω.
05) $\Omega^* = \{ \varepsilon \} \cup \Omega^+$.
06) $L \subseteq \Omega^*$.
07) $L^0 = \{ \varepsilon \}$.
08) $L^1 = L$.
09) $L^{n+1} = \{ x \circ y = xy : x \in L, y \in L^n \}$.
10) L^+ = Plus closure of L.
11) L^* = Kleen's closure of L.
12) $L^* = \{ \varepsilon \} \cup L^+$.

3.4.1 Function KBO Types

Definition 9 Let K be a parameteric list of KBO's or KBO clusters, and M be a KBO-plem cluster memory in a given universal KBO cluster memory Ü. Let viros be a brief vocabulary for "**vi**rtual **r**obotic **o**perating **s**ystem." The followings are called viros-function configurational KBO types:

01 **Begin** is a delimeter for the begining of a new action in the current agorithm.

02 **Reserve(K: M)** is a viros-function type that it reserves a KBO K in the M^{th} partition of Ü.

03 **Inject(K: M)** is a viros-function type that it injects a KBO K into M^{th} partition of Ü.

04 **Point(K: M; x)** is a viros-function type that it points a KBO K in the M^{th} partition of Ü by a beam type x.

05 **ShowKonfigurationof(K: M)** is a viros-function type that it shows the configuration of KBO **K** in the M^{th} partition of Ü.

06 **ShowDesignof(K: M)** is a viros-function type that it shows design of KBO K in the M^{th} partition of Ü.

07 **ShowBeamof(K: M)** is a viros-function type that it shows beam of KBO K in the M^{th} partition of Ü.

08 **ShowusCultureof(K: M)** is a viros-function type that it shows the us-culture of KBO K in the M^{th} partition of Ü.

09 **ShowLoadof(K: M)** is a viros-function type that it shows the loads of KBO K in the M^{th} partition of Ü.

10 **FixLoad(K: M)** is a viros-function type that it fixes the load of KBO K in M.

11 **RealizeLoad(K: M)** is a viros-function type that it realizes the fixed load of KBO K in M.

12 **ShowFixedLoad(K: M)** is a viros-function type that it showes the fixed loads of KBO K in the M^{th} partition of Ü .

13 **BeamFor Load(A, B:M; x, y)** is a viros-function type that it loads KBO A and KBO B in M by the beams x and y.

14 **Stop** stops the action of a current algorithm for a while.

15 **End** is delimeter for ending the action of the current algorithm.

4 Remote Programming Style

Definition 10 Let 1) *emw* be a brief vocabulary for "**e**lectoral **m**agnetic **w**ave." 2) $\langle$ x $\rangle$ be a remote KBO x with a pair of broadcasting and intercepting antennae network that they are sensitive to *emw*. Any programing style that programs x from a remote distance by using some coded(or codable) information into beam of *emw*(or light) by a FTD technology is called a *remote programming style.*

Theorem Let 1) M be a reserved KBO plem memory cluster in KBO Q^{th} partition of KBO Ü; 2) "**viros**" be a brief vocabulary for describing the notion of "**vi**rtual **r**obotic **o**perating **s**ystem;" 3) A, B, C and Û be KBO-clusters can be generated in M by injection such a way that A and B becomes to be generated in C and C becomes to be generated in Û; 4) F be a viros-function type (or class) KBO-cluster that it contains the viros-functions given in Definition 9. There is at least one remote programming style to program: (a) A and B in C in Û by a *plemwan x;* (b) A and B in C in Û by a *plemwanal word y*; (c) A and B in C in Û by a *plemwanal z*.

Proof (a) The following **Algorithm R1** is a remote programming style to program A and B in C in $\hat{U}$ by a *plemwan*. (b) Algorithm R2 is a remote programming style to program A and B in C in $\hat{U}$ by a *plemwanal word*. (c) Algorithm R3 is a remote programming style to program A and B in C in $\hat{U}$ by a plemwanal.

(a) **Algorithm R1**

Let x and y be two variables on the plemwan KBO's. Let the conditions in the hypothesis exist for the current use.

S00 **Begin**;

S01 **Reserve(M: Ü)** reserves the M^{th} partion of Ü. Hence, one has $\ddot{U} \leftharpoonup M$;

S02 **Inject($\hat{U}$: M)** injects $\hat{U}$ into M^{th} partition of Ü. Hence, one has $\ddot{U} \leftharpoonup M \leftharpoonup \hat{U}$;

S03 **Inject**(C: $\hat{U}$) injects C into C^{th} partition of M. Hence, one has $\ddot{U} \leftharpoonup M \leftharpoonup \hat{U} \leftharpoonup C$;

S04 **Inject**(A, B: C) injects A and B into C^{th} partion of $\hat{U}$. Hence, one has $\ddot{U} \leftharpoonup M \leftharpoonup \hat{U} \leftharpoonup C \leftharpoonup A$ and|or B;

S05 **Point** ($A \leftharpoonup x = \langle\, r_3, r_2, r_1 : r_u \,\rangle$: C) points A by a ray of light called x that does carry some loaded information $\langle\, r_3, r_2, r_1 : r_u \,\rangle$. Hence, one has $\ddot{U} \leftharpoonup M \leftharpoonup \hat{U} \leftharpoonup C \leftharpoonup (A \leftharpoonup x = \langle\, r_3, r_2, r_1 : r_u \,\rangle, B)$;

S06 **Point**($B \leftharpoonup y = \langle\, r_1, r_2, r_3 : r_s \,\rangle$: C) points B by a ray of light called y that does carry some loaded information $\langle\, r_1, r_2, r_3 : r_s \,\rangle$. Hence, one has $\ddot{U} \leftharpoonup M \leftharpoonup \hat{U} \leftharpoonup C \leftharpoonup (A \leftharpoonup x = \langle\, r_3, r_2, r_1 : r_u \,\rangle, B \leftharpoonup y = \langle\, r_1, r_2, r_3 : r_s \,\rangle : C)$;

S07 **ShowConfigurationof**(A, B: C) shows the configuration of (A, B) in C. Hence, one has $C \leftharpoonup (A \leftharpoonup x = \langle\, r_3, r_2, r_1 : r_u \,\rangle, B \leftharpoonup y = \langle\, r_1, r_2, r_3 : r_s \,\rangle)$ implies that $C \leftharpoonup (A \leftharpoonup \langle r_3, r_2, r_1 : r_u \rangle = \langle\, {}^{u}r_3 \;\; r_u \;\; {}^{u}r_1 \,\rangle = \langle\, {}^{u}r_2 \,\rangle, B \leftharpoonup \langle r_1, r_2, r_3 : r_s \rangle = \langle\, {}^{s}r_1 \;\; r_s \;\; {}^{s}r_3 \,\rangle = \langle\, {}^{s}r_2 \,\rangle : C)$.

Where (i) ${}^{u}r_1 \leftharpoonup v_1$, $v_1 \in [p + 1] = \{ 0, 1, ..., p\}$, $p \in Z^+$; (ii) ${}^{u}r_2 \leftharpoonup v_2$, $v_2 \in [x + 1] = \{ 0, 1, ..., x\}$, $x \in \mathring{A}$, where Å is a letteral alphabet; (iii) ${}^{u}r_3 \leftharpoonup v_3$, $v_3 \in [m + 1] = \{ 0, 1, ..., m\}$, $m \in Z^+$ (iv) ${}^{s}r_1 \leftharpoonup v_1$, $v_1 \in [q + 1] = \{ 0, 1, ..., q\}$, $q \in Z^+$; (v) ${}^{s}r_2 \leftharpoonup v_2$, $v_2 \in [y + 1] = \{ 0, 1, ..., y\}$, $y \in \ddot{A}$, where Ä is a letteral alphabet; (vi) ${}^{s}r_3 \leftharpoonup v_3$, $v_3 \in [n + 1] = \{ 0, 1, ..., n\}$, $n \in Z^+$ }.

S08 **Point**($A \leftharpoonup x = \langle\, r_3 \leftharpoonup \nu \bmod (m + 1), r_2 \leftharpoonup A, r_1 \leftharpoonup v \bmod (q + 1) : r_u \,\rangle$: C) points A by a ray of light called x that does carry the loaded information $\langle\, r_3 \leftharpoonup \nu \bmod 6, r_2 \leftharpoonup A, r_1 \leftharpoonup \nu \bmod 4 : r_u \,\rangle$. Hence, one has $C \leftharpoonup (A \leftharpoonup x = \langle\, r_3, r_2, r_1 : r_u \,\rangle, B \leftharpoonup y = \langle\, r_1, r_2, r_3 : r_s \,\rangle)$ implies that $C \leftharpoonup (A \leftharpoonup \langle r_3, r_2, r_1 : r_u \rangle = \langle\, {}^{u}r_3 \;\; r_u \;\; {}^{u}r_1 \,\rangle = \langle\, {}^{u}r_2 \,\rangle = \langle\, \nu \bmod 6 \;\; \circlearrowleft \;\; \nu \bmod 4 \;\; A \,\rangle$,

$\mathrm{B} \leftarrowtail \langle \mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3 : \mathrm{r}_s \rangle = \langle \begin{smallmatrix} {}^{s}r_1 & r_s & {}^{s}r_3 \\ & {}_{s}r_2 & \end{smallmatrix} \rangle = \langle \begin{smallmatrix} {}^{\circlearrowleft}r_1 & \circlearrowleft & {}^{\circlearrowleft}r_3 \\ & {}_{\circlearrowleft}r_2 & \end{smallmatrix} \rangle : \mathrm{C})$.

S09 **Point**(B $\leftarrowtail$ y = $\langle$ $\mathrm{r}_1 \leftarrowtail \nu \bmod 5, \mathrm{r}_2 \leftarrowtail$ B, $\mathrm{r}_3 \leftarrowtail \nu \bmod 8 : \mathrm{r}_s \rangle$: C) points B by a ray of light called y that does carry the loaded information $\langle$ $\mathrm{r}_1 \leftarrowtail \nu \bmod 6, \mathrm{r}_2 \leftarrowtail$ B, $\mathrm{r}_3 \leftarrowtail \nu \bmod 8 : \mathrm{r}_u \rangle$. Hence, one has C $\leftarrowtail$ (A $\leftarrowtail$ x = $\langle$ r_3, $\mathrm{r}_2, \mathrm{r}_1 : \mathrm{r}_u \rangle$,B$\leftarrowtail$ y = $\langle$ $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3 : \mathrm{r}_s \rangle$) implies that C $\leftarrowtail$ (A $\leftarrowtail \langle \mathrm{r}_3, \mathrm{r}_2, \mathrm{r}_1 : \mathrm{r}_u \rangle = \langle \begin{smallmatrix} {}^{u}r_3 & r_u & {}^{u}r_1 \\ & {}_{u}r_2 & \end{smallmatrix} \rangle = \langle \begin{smallmatrix} {}^{\circlearrowleft}r_3 & \circlearrowleft & {}^{\circlearrowleft}r_1 \\ & {}_{\circlearrowleft}r_2 & \end{smallmatrix} \rangle = \langle \begin{smallmatrix} \upsilon \bmod 6 & \circlearrowleft & \nu \bmod 4 \\ & \mathrm{A} & \end{smallmatrix} \rangle$, B $\leftarrowtail \langle \mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3 : \mathrm{r}_s \rangle = \langle \begin{smallmatrix} {}^{s}r_1 & r_s & {}^{s}r_3 \\ & {}_{s}r_2 & \end{smallmatrix} \rangle = \langle \begin{smallmatrix} {}^{\circlearrowleft}r_1 & \circlearrowleft & {}^{\circlearrowleft}r_3 \\ & {}_{\circlearrowleft}r_2 & \end{smallmatrix} \rangle = \langle \begin{smallmatrix} \nu \bmod 6 & \circlearrowleft & \upsilon \bmod 8 \\ & \mathrm{B} & \end{smallmatrix} \rangle$: C).

S10 **MakeDesign**(A, B: C) makes design of (A, B) in C. Hence one has C $\leftarrowtail$ (A $\leftarrowtail$ x = $\langle$ $\mathrm{r}_3, \mathrm{r}_2, \mathrm{r}_1 : \mathrm{r}_u \rangle$,B$\leftarrowtail$ y = $\langle$ $\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3 : \mathrm{r}_s \rangle$) implies that C $\leftarrowtail$ (A $\leftarrowtail \langle \mathrm{r}_3, \mathrm{r}_2, \mathrm{r}_1 : \mathrm{r}_u \rangle = \langle \begin{smallmatrix} {}^{u}r_3 & r_u & {}^{u}r_1 \\ & {}_{u}r_2 & \end{smallmatrix} \rangle = \langle \begin{smallmatrix} {}^{\circlearrowleft}r_3 & \circlearrowleft & {}^{\circlearrowleft}r_1 \\ & {}_{\circlearrowleft}r_2 & \end{smallmatrix} \rangle =$

$\langle \begin{smallmatrix} 5 \bmod 6 & \circlearrowleft & 3 \bmod 4 \\ & \mathrm{A} & \end{smallmatrix} \rangle$, B $\leftarrowtail \langle \mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3 : \mathrm{r}_s \rangle = \langle \begin{smallmatrix} {}^{s}r_1 & r_s & {}^{s}r_3 \\ & {}_{s}r_2 & \end{smallmatrix} \rangle$

$= \langle \begin{smallmatrix} {}^{\circlearrowleft}r_1 & \circlearrowleft & {}^{\circlearrowleft}r_3 \\ & {}_{\circlearrowleft}r_2 & \end{smallmatrix} \rangle = \langle \begin{smallmatrix} \nu \bmod 6 & \circlearrowleft & \nu \bmod 8 \\ & \mathrm{B} & \end{smallmatrix} \rangle$: C)

$= (\langle \begin{smallmatrix} \nu \bmod 6 & \circlearrowleft & \nu \bmod 4 \\ & \mathrm{A} & \end{smallmatrix} \rangle , \langle \begin{smallmatrix} \nu \bmod 6 & \circlearrowleft & \upsilon \bmod 8 \\ & \mathrm{B} & \end{smallmatrix} \rangle : C)$.

S11 **ShowDesignof**(A, B: M; x, y) gives the designs of A and B like

$$(\mathrm{A} \longleftarrow \langle \begin{smallmatrix} 4 & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc \\ & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ \hookleftarrow & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft \\ 6 & A_{-5} & A_{-4} & A_{-3} & A_{-2} & A_{-1} & A_{-0} \end{smallmatrix} \rangle$$

,

$$\mathrm{B} \longleftarrow \langle \begin{smallmatrix} \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & \bigcirc & 6 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \\ \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \hookrightarrow \\ B_{+0} & B_{+1} & B_{+2} & B_{+3} & B_{+4} & B_{+5} & B_{+6} & B_{+7} & 8 \end{smallmatrix} \rangle : C) .$$

S12 **Point**(A: M) shows A in M as in S11.

S13 **BeamForLoad**(A: M , x $\longleftarrow$ <133102> mod 4) registers (or stores) x into A in M. Hence A becomes like

$$\mathrm{A} \longleftarrow \langle \begin{smallmatrix} 4 & 1 & 3 & 3 & 1 & 0 & 2 \\ & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ \hookleftarrow & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft \\ 6 & A_{-5} & A_{-4} & A_{-3} & A_{-2} & A_{-1} & A_{-0} \end{smallmatrix} \rangle$$

S14 **Point**(B: M) shows B in M as in S11.

S15 **BeamForLoad**(B: M; y $\longleftarrow$ <52022534> mod 6) registers (or stores) y into B in M. Hence B becomes like

$$\mathrm{B} \longleftarrow \langle \begin{array}{ccccccccc} 5 & 2 & 0 & 2 & 2 & 5 & 3 & 4 & 6 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \\ \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \hookrightarrow \\ B_{+0} & B_{+1} & B_{+2} & B_{+3} & B_{+4} & B_{+5} & B_{+6} & B_{+7} & 8 \end{array} \rangle$$

S16 **ShowBeamof**(A, B: C) Shows <133102> mod 4 in A and <52022534> mod 6 in B. Hence we have:

$$(\mathrm{A, B: C}) = (\mathrm{A} \longleftarrow \langle \begin{array}{ccccccc} 4 & 1 & 3 & 3 & 1 & 0 & 2 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \\ \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \hookleftarrow \\ 6 & A_{-5} & A_{-4} & A_{-3} & A_{-2} & A_{-1} & A_{-0} \end{array} \rangle ,$$

$$\mathrm{B} \longleftarrow \langle \begin{array}{ccccccccc} 5 & 2 & 0 & 2 & 2 & 5 & 3 & 4 & 6 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \\ \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \circlearrowleft & \hookrightarrow \\ B_{+0} & B_{+1} & B_{+2} & B_{+3} & B_{+4} & B_{+5} & B_{+6} & B_{+7} & 8 \end{array} \rangle : \mathrm{C}).$$

S17 **Stop** ;
S18 **End**.

(b) **Algorithm R2** Let x and y be plemwanal words. Let the conditions in the hypothesis exist for the current use. S00 **Begin**; S01 **Reserve**(M: a KBO-plem memory cluster); S02 **Inject**($\hat{\mathrm{U}}$) into M; **Inject**(C) into $\hat{\mathrm{U}}$; **Inject**(A, B) into C; S03 **Point** (A); A $\leftharpoondown$ x = $\langle$ r_3, r_2, r_1 : r_u $\rangle\langle$ r_1, r_2, r_3 : r_s $\rangle\langle$ r_3, r_2, r_1 : r_u $\rangle$; S04 **Point**(B); B $\leftharpoondown$ y = $\langle$ r_1, r_2, r_3 : r_s $\rangle\langle$ r_1, r_2, r_3 : r_s $\rangle\langle$ r_3, r_2, r_1 : r_u $\rangle\langle$ r_1, r_2, r_3 : r_s $\rangle\langle$ r_3, r_2, r_1 : r_u $\rangle$; S05 **ShowConfigurationof**(A, B); S06 **Point**(A) ; A $\leftharpoondown$ x = $\langle$ r_3 $\leftharpoondown$ 5, r_2 $\leftharpoondown$ A, r_1 $\leftharpoondown$ 3 : r_u $\rangle\langle$ r_1 $\leftharpoondown$ 5, r_2 $\leftharpoondown$ A, r_3 $\leftharpoondown$ 2 : r_s $\rangle\langle$ r_3 $\leftharpoondown$ 5, r_2 $\leftharpoondown$ A, r_1 $\leftharpoondown$ 4 : r_u $\rangle$; **Point**(B); B $\leftharpoondown$ y = $\langle$ r_1 $\leftharpoondown$ 5, r_2 $\leftharpoondown$ B, r_3 $\leftharpoondown$ 2 : r_s $\rangle\langle$ r_1 $\leftharpoondown$ 3, r_2 $\leftharpoondown$ B, r_3 $\leftharpoondown$ 2: r_s $\rangle\langle$ r_3 $\leftharpoondown$ 2, r_2 $\leftharpoondown$ B, r_1 $\leftharpoondown$ 2: r_u $\rangle\langle$ r_1 $\leftharpoondown$ 2, r_2 $\leftharpoondown$B, r_3 $\leftharpoondown$ 2: r_s $\rangle$; **Design**(A, B) ; S07 **ShowDesignof**(A, B) ; S08 **Point**(A) ; **Beam**(A); **Point**(B); **Beam**(B) ; S09 **ShowBeamof**(A, B) ; S10 **Point**(A, B) ; **Load** B) ; S11 **ShowLoadedof**(A, B) ; S12 **Stop** ; S13 **End**.

(c) **Algorithm R3** Let 1) w_1 = $\langle$ r_3, r_2, r_1 : r_u $\rangle$ $\langle$ r_1, r_2, r_3 : r_s $\rangle$ $\langle$ r_3, r_2, r_1 : r_u $\rangle$ and w_2 = $\langle$ r_1, r_2, r_3 : r_s $\rangle$ $\langle$ r_1, r_2, r_3 : r_s $\rangle$ $\langle$ r_3, r_2, r_1 : r_u $\rangle$ $\langle$ r_1, r_2, r_3 : r_s $\rangle$ be two plemwanal words; 2) L_1 = {w_1} and L_2 = { w_1, w_2} be two plemwanals. S00 **Begin**; S01 **Reserve**(M: a KBO-plem memory cluster); S02 **Inject**($\hat{\mathrm{U}}$) into M; **Inject**(C) into $\hat{\mathrm{U}}$; **Inject**(A, B) into C; S03 **Point** (A); A $\leftharpoondown$ w_1 = $\langle$ r_3, r_2, r_1 : r_u $\rangle\langle$ r_1, r_2, r_3 : r_s $\rangle\langle$ r_3, r_2, r_1 : r_u $\rangle$; S04 **Point**(B); B $\leftharpoondown$ w_2 = $\langle$ r_1, r_2, r_3 : r_s $\rangle\langle$ r_1, r_2, r_3 : r_s $\rangle\langle$ r_3, r_2, r_1 : r_u $\rangle\langle$ r_1, r_2, r_3 : r_s $\rangle\langle$ r_3, r_2, r_1 : r_u $\rangle$; L_1 $\leftharpoondown$ {A}; L_2 $\leftharpoondown$ {A, B}; S05 **ShowConfigurationof**(L_1, L_2); S06 **Point**(A) ; A $\leftharpoondown$ $\langle$ r_3 $\leftharpoondown$ 5, r_2 $\leftharpoondown$ A, r_1 $\leftharpoondown$ 3 : r_u $\rangle\langle$ r_1 $\leftharpoondown$ 5, r_2 $\leftharpoondown$ A, r_3 $\leftharpoondown$ 2 : r_s $\rangle\langle$ r_3 $\leftharpoondown$ 5, r_2 $\leftharpoondown$ A, r_1 $\leftharpoondown$ 4 : r_u $\rangle$; **Point**(B); B $\leftharpoondown$ $\langle$ r_1 $\leftharpoondown$ 5, r_2 $\leftharpoondown$ B, r_3 $\leftharpoondown$ 2 : r_s $\rangle\langle$ r_1 $\leftharpoondown$ 3, r_2 $\leftharpoondown$ B, r_3 $\leftharpoondown$ 2: r_s $\rangle\langle$ r_3 $\leftharpoondown$ 2, r_2 $\leftharpoondown$ B, r_1 $\leftharpoondown$ 2: r_u $\rangle\langle$ r_1 $\leftharpoondown$ 2, r_2 $\leftharpoondown$B, r_3 $\leftharpoondown$ 2: r_s $\rangle$; **Design**(A, B); L_1 $\leftharpoondown$ {A}; L_2 $\leftharpoondown$ {A, B}; S07

ShowDesignof(L_1, L_2); S08 **Point**(A) ; **Beam**(A); **Point**(B); **Beam**(B) ; $L_1 \leftarrow\!\!\prec \{A\}$; $L_2 \leftarrow\!\!\prec \{A, B\}$; S09 **ShowBeamof**($L_1$, L_2) ; S10 **Point**(A, B) ; **Load**(A**,** B) ; $L_1 \leftarrow\!\!\prec \{A\}$; $L_2 \leftarrow\!\!\prec \{A, B\}$; S11 **ShowLoaded**($L_1$, L_2) ; S12 **Point**(A, B) ; **Fix**(A**,** B) ; $L_1 \leftarrow\!\!\prec \{A\}$; $L_2 \leftarrow\!\!\prec \{A, B\}$; S13 **ShowFixed**($L_1$, L_2) ; S14 **Stop** ; S15 **End**.

5 Main Results

This paper develops, studies and introduces us-crop based *plemwanal. It is a formal language type in which numerical words with communicating antennae produce a special computer mathematics generator type of machine KBO. It has a self communication process in the overall process of organizing token classical or token quantum substructures. Plemal in this paper has been introduced by 10 definitions, 2 corollaries and one theorem* for covering the ideas of configurational KBO types and remote programming style. Author claims that *plemwanal* is a very special communicating computer mathematics generating language type. It is produced as an extension of *plemal* for obtaining an artificial intelligently communicating numerical words with antennae. These numerical words with antennae are communicating themselves by broadcasting and intercepting information in the execution time of its substructures. Each token numerical word with antennae in each substructure has wireless communication property with each other and with its environment. It is designed for generating remote, parallel, sequential, mobile, distributed, etc. type us-crop based compact I@I KBO cluster plem memory programming in a classical or quantum information processing environment. After plemwanal there is going to be a new computing mathematics type. The author calls it as "Communicating Computer Mathematics."

References

[1] F. Ünlü and S. Sönmez, *I@I : Tıkız Internet Tabanlı Tıkız Internet*, II. Proceeding of Information Technology Congress, pp. 246-248, 1-May 2003, Pamukkale University-Denizli, Turkey.

[2] F. Ünlü and S. Sönmez, *Convoluted Prüfer's Type I@I,* International Mathematical Journal, Vol. 4, no. 6, pp. 539 - 547, 2003.

[3] F. Ünlü, *An Intuitive Differential Equation Model for Knowledge Based Objects(KBO) Representation of Science*, ISCISXIV, Proceeding of The Fourteenth International Symposium on Computer and Information Science, pp1066-1068, October, 18-20, Kuşadası, Aydın, Turkey, 1999.

[4] F. Ünlü, *FTD Grammar Graphs*, International Journal of Computer Mathematics Vol. 80, no. 1, pp1-9, January 2003.

[5] F. Ünlü, *Chance Constrained Threshold KBO System Design,* International Mathematical Journal, Vol. 5, no. 4, pp 321 - 328, 2004.

[6] F. Ünlü, A Generalized us-Culture Job Scheduling for Forecasting Problems, International Mathematical Journal, Vol. 4, no. 4, pp 313-320, 2003.

[7] F. Ünlü, S. Sönmez, and Z. I. Ünlü, *Linear Circular Convoluted I@I KBO Cluster Generating us-Crop,* International Mathematical Journal, Vol. 5, no. 4, pp 329 - 338, 2004.

[8] F. Ünlü, *tuze-Channeled KBO for DCP,* International Mathematical Journal, Vol. 5. no. 4, pp 339 - 346, 2004.

[9] F. Ünlü, The *Fundamental Mathematics of us-Crop Based Compact plem Memory, (unpublished).*

[10] F. Ünlü, The *Fundamental Mathematics of Plemal, (unpublished).*

[11] S. Washburn, T. Marlowe and C. T. Ryan, *Discrete Mathematics, Addson-Wesley, New York, 2000.*

[12] J. D. Baum, Elements of Point Set Topology, Prentice-Hall, Inc., Engelwood Cliffs, N.J., 1964.

[13] R. P. Grimaldı, *Discrete and Combinatorial Mathematics, Addison-Wesley, New York, 1999.*

Received: October 25, 2005