Genesis of the Rationality from 'Chaos'

A. G. Grappone
Editor-in-Chief of "Metalogicon"

Ἤτοι μὲν πρώτιστα Χάος ....
Ἐκ Χάεος δ' Ἔρεβός τε μέλαινά τε Νὺξ ἐγένοντο·
Νυκτὸς δ' αὖτ' Αἰθήρ τε καὶ Ἡμέρη ἐξεγένοντο ...
(HESIODUS, *Θεογονία*, 114-124)

Abstract

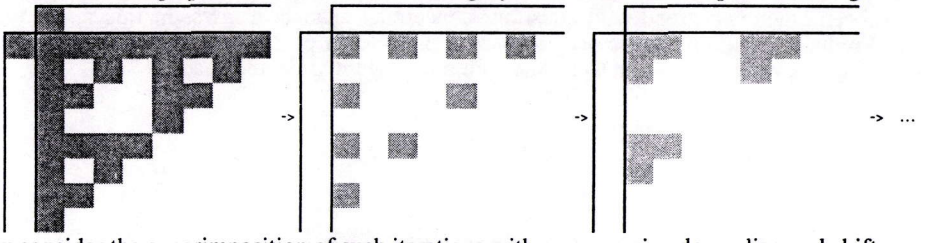
This paper proves that the Boolean connectives and the recursive functions —i. e. the rational order— are generable by hyperincursive algorythms —i. e. 'chaotic' phenomena— as the above Hesiodus' verse affirms. But, for the archaic ancient Greeks, *chaos* —the space that is enclosed between sky and earth [χάος = χά(ϝ)ος =*cavus* = cave?]— does not mean *disorder*. Were they right?

Keywords

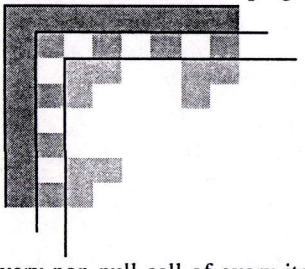
Hyperincursive function; Sierpinski's triangle; Recursive function; Sheffer's connective;

1. Dubois' Algorythm for Sierpinski's Triangle and Dubois' Succession

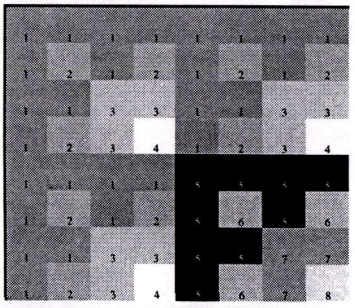
Consider the graphic iterations of Dubois' algorythm to calculate Sierpinski's triangle:



Now consider the superimposition of such iterations with a progressive down diagonal shift:

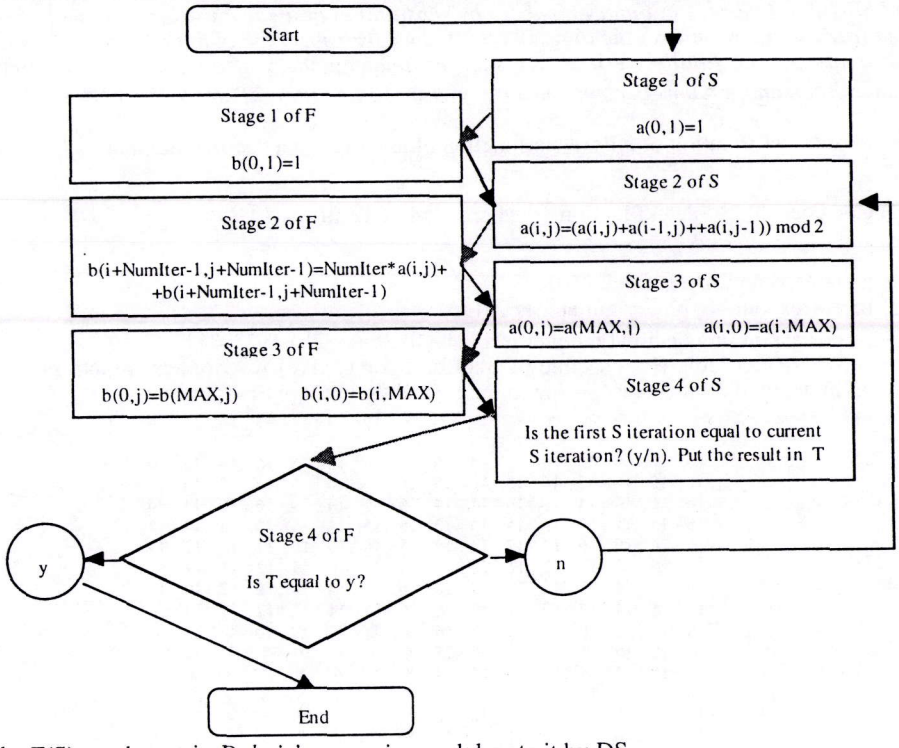


Given the last superimposition, every non null cell of every iteration superimposes itself to null cells of other iterations only. Assign the number n to non null cells of the n-th iteration. So we can conserve the entire information of all the iterations in a sole table. We have:



International Journal of Computing Anticipatory Systems, Volume 1, 1998
Ed. by D. M. Dubois, Publ. by CHAOS, Liège, Belgium. ISSN 1373-5411 ISBN 2-9600179-1-9

The tables that represent such superimposition of a number at will of the iterations of Dubois' algorythm for Sierpinski's triangle can be obtained by a composition of two hyperincursive algorythms, stage by stage, F(S) where S denotes the known Dubois' algorithm for Sierpinski's triangle and F another opportune hyperincursive algorythm. Suppose that F is implemented on a matrix 'b' and S on a matrix 'a'. F(S) is represented by this flowchart:



Call the F(S) result matrix *Dubois' succession* and denote it by DS.

2. Dubois' Succession as Fundament of Numeric Calculus and Boole's Algebra

Assign to every cell of DS a couple of non negative integer indexes. The highest cell most to the left has indexes 0, 0 and the indexes increase towards the right and in a downwards direction. So, denote every cell of DS with indexes i, j with the symbol DS(i,j). Consider the set $\mathbb{N}$ $=\{1,2,3,\ldots\}$ We can define two $\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ applications: $\mathfrak{N}(n,m)=p$ by putting $\mathfrak{N}(n,m)=p$ iff DS(n , m)=p and $\mathfrak{B}(m,n)=p$ by putting $\mathfrak{B}(m,n)=p$ iff DS(n -1, m -1)+1=p

Theorem 2.1: *Every recursive function on* $\mathbb{N}$ *derives from* $\mathfrak{N}(m,n)$ *by applications. of* substitution $f(n)=g(h(n))$, recursion $f(n)=\mathrm{rec}\langle f(1)=g(n)|f(N(n))=h(f(n))\rangle$, μ-operator $f(n)=\mu k(g(n,k))$ (we assume 1 as minime number obviously).

Proof:

a) function *minimal number* $Z(n)=1$: $Z(n)=\mathfrak{N}\left(n,\mu k\left(\mathfrak{N}(n,k)\right)\right)$;

b) function *number successor* $N(n)=n+1$: $N(n)=\mathfrak{N}(n,n)$;

c) function *number predecessor* $\partial(n)=n-1$: $\partial(n)=\mathrm{rec}\langle\partial(1)=1|\partial(N(n))=n\rangle$;

d) function *number sum* $n+m$: $n+m=\mathrm{rec}\langle n+1=N(n)|n+N(m)=N(n+m)\rangle$;

e) function *positive difference* $n-m$: $n-m=\mathrm{rec}\langle n-1=\partial(n)|n-N(m)=\partial(n-m)\rangle$;

f) function *left projection* $U_1^2(n,m)=n$: $U_1^2(n,m)=n+m-m$;

g) function *right projection* $U_2^2(n,m)=m$: $U_2^2(n,m)=n+m-n$;

h) function *projection* $U_i^k(n_1,\ldots,n_i,\ldots,n_k)=n_i: U_i^k(n_1,\ldots,n_i,\ldots,n_k)=U_2^2\left(n_1,U_2^2\left(\ldots U_1^2\left(n_i,U_1^2(\ldots)\right)\ldots\right)\right)$

i) It is a known result that *minimal number, number successor* and *projection* are a complete system of primitive recursive function, i. e. we can obtain any recursive function from them by substitution, recursion and μ-operator. This last fact proves the theorem.

Theorem 2.2: *If we transpose a set of Boolean functions of length* l, *and then, we transform the obtained horizontal vectors in binary numbers by substitution of the truth value* true *with the digit* 1 *and the truth value* false *with the digit* 0, *and that we transform the obtained binary numbers in decimal numbers, and finally we increase such decimal numbers of one, then the connectives among such Boolean functions become numeric operations and they all derive from* $\mathfrak{B}(m,n)$.

Proof

a) Example of transformation of a Boolean function of length l in a decimal number.

$$\begin{matrix} l \\ 0 \\ l \\ l \end{matrix}$$

a0) Consider the Boolean function where l =4 evidently;

a1) its matricial transposition is *1011*;

a2) its corresponding binary number is 1011;

a3) its corresponding decimal number is $1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0=11$;

b) Example of NAND table (i. e. a table of Sheffer's connective) for Boolean functions that are transformed in decimal numbers according to the procedure in a): put l =4, we have:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
2	16	15	16	15	16	15	16	15	16	15	16	15	16	15	16	15
3	16	16	14	14	16	16	14	14	16	16	14	14	16	16	14	14
4	16	15	14	13	16	15	14	13	16	15	14	13	16	15	14	13
5	16	16	16	16	12	12	12	12	16	16	16	16	12	12	12	12
6	16	15	16	15	12	11	12	11	16	15	16	15	12	11	12	11
7	16	16	14	14	12	12	10	10	16	16	14	14	12	12	10	10
8	16	15	14	13	12	11	10	9	16	15	14	13	12	11	10	9
9	16	16	16	16	16	16	16	16	8	8	8	8	8	8	8	8
10	16	15	16	15	16	15	16	15	8	7	8	7	8	7	8	7
11	16	16	14	14	16	16	14	14	8	8	6	6	8	8	6	6
12	16	15	14	13	16	15	14	13	8	7	6	5	8	7	6	5
13	16	16	16	16	12	12	12	12	8	8	8	8	4	4	4	4
14	16	15	16	15	12	11	12	11	8	7	8	7	4	3	4	3
15	16	16	14	14	12	12	10	10	8	8	6	6	4	4	2	2
16	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

c) Define NAND(l, m, n)=p the numeric function such that m and n are decimal natural numbers that correspond to two Boolean functions of length l according to the procedure in a) and p is a decimal natural number that corresponds to the NAND of two such Boolean functions.

d) define m^n as $\mathrm{rec}\left\langle m^0=1\middle| m^k=m\cdot m^{\partial(k)}\right\rangle$.

e) Given the table in b) and its generalization, we deduce NAND(l, m, n)= $\left(2^l+1\right)-\mathfrak{B}(m+1,n+1)$.

f) All the Boolean connectives are equivalent to compositions of Sheffer's connectives (connectives NAND). This last fact proves the theorem.

3. An Example of Boolean Calculus by $\mathfrak{B}(n,m)=DS(n-1,m-1)+1$

Consider the wff *Cpq*. To calculate its truth functions, transform *Cpq* in its equivalent wff *DpDqq* where *D* is the Sheffer's connective, i. e. the NAND connective.

Build an opportune calculus table. Observe that the lenght of the truth functions l is 4.

D	*p*	*D*	*q*	*q*
	1		*1*	*1*
	1		*0*	*0*
	0		*1*	*1*
	0		*0*	*0*

Now, transform the truth functions in decimal numbers. We have:

$1100=1100_2=\left(1\cdot 2^3+1\cdot 2^2+0\cdot 2^1+0\cdot 2^0\right)_{10}=12_{10}=12$

$1010 = 1010_2 = \left(1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0\right)_{10} = 10_{10} = 10$

So, the calculus table becomes:

D	p	D	q	q
	12		10	10

by developing the calculus we have:

D	p	D	q	q
	12	$NAND(4,10,10)$	10	10

D	p	D	q	q
	12	$(2^4+1)-\mathfrak{B}(10+1,10+1)$	10	10

D	p	D	q	q
	12	$17-\mathfrak{B}(11,11)$	10	10

D	p	D	q	q
	12	17-12	10	10

D	p	D	q	q
	12	5	10	10

D	p	D	q	q
$NAND(4,12,15)$	12	5	10	10

D	p	D	q	q
$(2^4+1)-\mathfrak{B}(12+1,5+1)$	12	5	10	10

D	p	D	q	q
$17-\mathfrak{B}(13,6)$	12	5	10	10

D	p	D	q	q
17-6	12	5	10	10

D	p	D	q	q
11	12	5	10	10

Now, transform the truth functions in decimal numbers. We have:

$11 = 11_{10} = 1011_2 = 1011$

$12 = 12_{10} = 1100_2 = 1100$

$5 = 5_{10} = 0101_2 = 0101$

$10 = 10_{10} = 1010_2 = 1010$

So, the calculus table becomes

D	p	D	q	q
1	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0

The most left truth function of the calculus table $\begin{matrix} 1 \\ 0 \\ 1 \\ 1 \end{matrix}$ is the searched truth function of *Cpq*.

4. A Computer Program to Calculate the Dubois' Succession

```
/*********************************************************************** *
 *                                                                       *
 * DUBOIS' SUCCESSION                                                    *
 *                                                                       *
 *                                                                       *
 * Arturo Graziano Grappone                                              *
 * Editor-in-Chief of "METALOGICON"                                      *
 * Via Carlo Dossi 87 00137 ROMA (ITALY)                                 *
 *                                                                       *
 * Alpha version                                                         *
 *                                                                       *
 *********************************************************************** */

/* INCLUSIONS */

#include <stdio.h>      /* Standard inclusion */


/* CONSTANTS */

/* Array management */

#define IOVF    /* Index overfolw                                 */  17
#define C2OF    /* Cell overflow in 2-dimensioned arrayes         */  (IOVF*IOVF)
#define MDGT    /* Maximum of element digits in the screen        */  2
#define MCOL    /* Maximum of array columns in the screen         */  20
#define MROW    /* Maximum of array rows in the screen ..         */  (80/MDGT)


/* MACROS */

/* Array management */

#define IN(A,B)         /* Double index          */   ((A)*IOVF+(B))
#define MN(A,B)         /* Minime number         */   ((A)<(B)? (A):(B))
#define F(A,B,C)        /* FOR loop for array    */   for((A)=(B);(A)<(C);++(A))


/* OBJECTS */

/* Pause */

class Pause {
 public:
  char exe(void); /* Like to the command PAUSE of MSDOS */
};
```

```
/* Integer Array */

class IntArr {
 public:
       int  a[C2OF];          /* RAM address of array      */
  virtual char init(void);    /* Put a[] to zero           */
  virtual char show(void);    /* Show a[] on the screen    */
};


/* Stack2 Array */

class St2Arr: public IntArr {
 public:
       int  b[C2OF];          /* RAM address of backup array  */
  virtual char init(void);    /* Put a[] and b[] to zero      */
       char push(void);       /* Backup a[] in b[]            */
       char pop(void);        /* Restore a[] from b[]         */
       char test(void);       /* Compare a[] and b[]          */
};


/* Cumulative Array */

class CumArr: public St2Arr {
 public:
  int  N[1];          /* RAM address of counter                  */
  int  c[C2OF];       /* RAM address of cumulative array         */
  char init(void);    /* Put every value of a[], b[], c[], N[] to zero  */
};


/* Sierpinski's Triangle by Dubois' Algorythm */

class Sierpinski: public CumArr {
 public:
  char Sstage1(void);         /* First stage of hyperincursive function      */
  char Sstage2(void);         /* Second stage of hyperincursive function     */
  char Sstage3(void);         /* Third stage of hyperincursive function      */
  char Sstage4(void);         /* Fourth stage of hyperincursive function     */
  char Sexe(void);            /* Dubois' algorythm for Sierpinski's triangle */
};


/* Dubois' Series */

class Dubois: public Sierpinski {
 public:
  char DStage1(char k);       /* First stage of hyperincursive function      */
  char DStage2(char k);       /* Second stage of hyperincursive function     */
  char DStage3(char k);       /* Third stage of hyperincursive function      */
  char DStage4(char k);       /* Fourth stage of hyperincursive function     */
  char Dexe(void);            /* Dubois' algorythm for Dubois' series        */
  char show(void);            /* Show c[] on the screen                      */
};
```

```
/* MAIN OF PROGRAM */

void main( void ){
 /* Declarations */
 Pause  p;
 Dubois x;
 /* Execution */
 x.DExe();
 x.show();
 p.exe();
}


/* FUNCTION BODIES */

char Dubois::show(void){
 /* Declarations */
 int i,j,r,k;
 /* Execution */
 printf( "\nRESULT:\n\n" );
 r=MN(MROW,IOVF);
 k=MN(MCOL,IOVF);
 F(i,0,r){
  F(j,0,k){
   printf( " %02d", c[IN(i,j)]%100 );
  }
  printf( "\n" );
 }
 return 0;
}


char Dubois::DExe(void){
 /* Declarations */
 /* Execution */
  DStage1(SStage1());
 do{
  DStage2(SStage2());
  DStage3(SStage3());
 }while(
  DStage4(SStage4()));
 return 0;
}


char Dubois::DStage4(char k){
 /* Declarations */
 /* Execution */
 return k;
}


char Dubois::DStage3(char k){
 /* Declarations */
 int i,m;
 /* Execution */
 m=IOVF-1;
 F(i,1,IOVF){c[IN(0,i)]=c[IN(m,i)];c[IN(i,0)]=c[IN(i,m)];}
 return 0;
}
```

```
char Dubois::DStage2(char k){
 /* Declarations */
 int i,j,r,h;
 r=IOVF-*N;
 h=IOVF-*N;
 /* Execution */
 F(i,1,r){F(j,1,h){
  c[IN(i+*N,j+*N)]+=(*N+1)*a[IN(i,j)];
 }}
 return 0;
}


char Dubois::DStage1(char k){
 /* Declarations */
 /* Execution */
 return 0;
}


char Sierpinski::SExe(void){
 /* Declarations */
 /* Execution */
  SStage1();
 do{
  SStage2();
  SStage3();
 }while(
  SStage4());
 return 0;
}


char Sierpinski::SStage4(void){
 /* Declarations */
 /* Execution */
 ++N[0];
 if(*N<2) return 1;
 else return !test();
}


char Sierpinski::SStage3(void){
 /* Declarations */
 int i,m;
 /* Execution */
 m=IOVF-1;
 F(i,1,IOVF){a[IN(0,i)]=a[IN(m,i)];a[IN(i,0)]=a[IN(i,m)];}
 if(*N==0) push();
 return 0;
}
```

```
char Sierpinski::SStage2(void){
 /* Declarations */
 int i,j;
 /* Execution */
 F(i,1,IOVF){F(j,1,IOVF){
  a[IN(i,j)]=(a[IN(i,j)]+a[IN(i-1,j)]+a[IN(i,j-1)])%2;
 }}
 return 0;
}


char Sierpinski::SStage1(void){
 /* Declarations */
 /* Execution */
 init();
 a[IN(0,1)]=1;
 return 0;
}



char CumArr::init(void){
 /* Declarations */
 int i;
 /* Execution */
 *N=0;
 F(i,0,C2OF){a[i]=0;b[i]=0;c[i]=0;}
 return 0;
}


char St2Arr::test(void){
 /* Declarations */
 char truth=1;
 int  i;
 /* Execution */
 F(i,0,C2OF){truth*=(a[i]==b[i]);}
 return truth;
}



char St2Arr::pop(void){
 /* Declarations */
 int i;
 /* Execution */
 F(i,0,C2OF){a[i]=b[i];}
 return 0;
}


char St2Arr::push(void){
 /* Declarations */
 int i;
 /* Execution */
 F(i,0,C2OF){b[i]=a[i];}
 return 0;
}
```

```
char St2Arr::init(void){
 /* Declarations */
 int i;
 /* Execution */
 F(i,0,C2OF){a[i]=5;b[i]=3;}
 return 0;
}


char IntArr::show(void){
 /* Declarations */
 int i,j,r,c;
 /* Execution */
 printf( "\nRESULT:\n\n" );
 r=MN(MROW,IOVF);
 c=MN(MCOL,IOVF);
 F(i,0,r){
  F(j,0,c){
   printf( " %02d", a[IN(i,j)]%100 );
  }
  printf( "\n" );
 }
 return 0;
}


char IntArr::init(void){
 /* Declarations */
 int i;
 /* Execution */
 F(i,0,C2OF){a[i]=0;}
 return 0;
}


char Pause::exe(void){
 /* Declarations */
 char c;
 /* Execution */
 printf("\n\nTo continue push [ENTER]... ");
 while( '\n'!=(c=getchar()));
 return 0;
}
```

5. References

DUBOIS D. and RESCONI G. [1992] *Hypericursivity- A new mathematical theory*
Presses Universitaires de Liège
DUBOIS D. [1992] *The Fractal Machine*
Presse Universitaires de Liège
DUBOIS D. and RESCONI G. [1995] *Lecture Notes in Thresold Logic*
AILg Comett European, February 16-17, 1995, Liège, Belgium
DUBOIS D. and RESCONI G. [1995] *Lecture Notes in Incursion*
AILg Comett European, February 20-21, 1995, Liège, Belgium
GRAPPONE A. G. [1995]*On Sentence Calculus* in *Research-in-Progress*, vol. II
IIAS c/o Prof. G. Lasker, School of Computer Science, Windsor, Ontario, Canada
MALATESTA M. [1997] *Primary Logic*
Gracewing Fowler Wright Books, Leominster, Herefordshire, UK
MENDELSON E. [1964] *Introduction to Mathematical Logic,*
D. Van Nostrand Company, Princeton, New Jersey