UNA ECUACIÓN DE LA NATURALEZA

Fernando Galindo Soria
Escuela Superior de Computo (ESCOM)
Instituto Politécnico Nacional
Av. Miguel Othón de Mendizábal y Av. Juan de Dios Bátiz s/n
Zacatenco, Cd. de México
07738 MÉXICO

fgalindo@ipn.mx

Cd. de México, 1 de Junio de 1998

RESUMEN

En este trabajo *se muestra como aplicar la ecuación lingüística S-->e*S* para representar la estructura de múltiples elementos de la naturaleza, incluyendo árboles, nubes, estrellas, montañas, caracoles y ríos*; la ecuación *S--> e*S** indica que *S* se puede sustituir por *e* S** y *S* significa Sistema, *e* es cualquier elemento del sistema, *e** significa que se pueden tener tantos elementos como se quieran y *S** indica que el sistema se puede llamar tantas veces como se quiera, por lo que, es fácil de representar mediante un pequeño programa de computo con el cual y dependiendo del valor de sus parámetros se pueden representar los diferentes objetos de la naturaleza.

Ahora bien, se ve que aunque la ecuación es general, la forma de los objetos depende del valor que toman sus parámetros, por lo que, *mostramos como generar diferentes tipos de objeto dependiendo del valor de sus parámetros y como encontrar y aplicar el espacio de caos (o sea el espacio en el cual se generan familias de objetos parecidos) asociado a cada tipo de objeto.*

Finalmente se ve que aunque esta ecuación se uso inicialmente para representar y generar objetos naturales, conforme paso el tiempo fue quedando claro que su aplicación es mas general. Por lo que también en este trabajo *se generaliza el campo de aplicación de la ecuación y se muestra como se pueden representar sistemas que tienen recursividad sobre la recursividad y que tienen empleo por ejemplo en la Hermeneutica, los Sistemas Conscientes y la generación de objetos clásicos de la teoría de Caos y Fractales como son los fractales de Dragón.*

Finalmente es importante comentar que lo que se esta mostrando es una ecuación de Lingüística Matemática con la cual se pueden modelar y representar múltiples elementos de la naturaleza y los ejemplos de graficación son solo ayudas para visualizar.

INTRODUCCIÓN

Dentro del área del tratamiento de imágenes es común encontrar problemas donde se requiere generar paisajes en los que se incluyan montañas, árboles, nubes, ríos y muchos otros elementos de la naturaleza, por lo que, desde hace algún tiempo y en particular desde el surgimiento de la teoría de Fractales se han desarrollado métodos y técnicas orientados a resolver este tipo de problemas.

Por ejemplo se tienen los trabajos desarrollados por Prusinkiewics y otros [1; 9], S. Peralta y A. Simancas [2] y E. Oppenheiner [3] sobre representación de plantas, arboles y arbustos, el trabajo de Reed y Wyvill [10] donde se muestra como generar rayos, el trabajo de R. Fowler y otros [12] donde se ve como generan conchas marinas muy realistas, en fin el trabajo de Kass y Miller [13] donde se muestra la generación de ondas de agua, playas y montañas, el trabajo de Barnsley y otros [6; 8] donde utilizando sistemas de funciones iterativas se generan plantas, nubes, montañas y paisajes y los trabajos pioneros de A. Lindenmayer sobre sistemas-L que se utilizan desde los años 70´s para la generación de plantas [1; 9] y de Benoit Mandelbrot y Richard F. Voss sobre fractales donde se ve como generar nubes, montañas y planetas aplicando generadores de números que siguen patrones brownianos y 1/f [5; 7].

La mayoría de estos trabajos se han desarrollado para resolver problemas específicos, ya que los métodos que se utilizan para generar rayos, montañas, conchas, nubes o arboles son en general diferentes entre si. Ahora bien en los trabajos de Marshall y otros [14] y L. Max [15] sobre modelos procedulares ya se plantean métodos orientados a la generación de múltiples tipos de objetos de la naturaleza y en los trabajos de Mandelbrot y Voss [5; 7] y Barnsley y otros [8] se introducen herramientas generales como los fractales, ruido 1/f y sistemas de funciones iterativas con las cuales es factible generar múltiples tipos de elementos de la naturaleza.

En particular una cosa que me ha llamado la atención desde los años 70´s es precisamente la búsqueda de mecanismos generales para la representación de elementos de la naturaleza, por lo que desde 1978 y a partir de las ideas de Lindenmayer sobre sistemas-L y Noam Chomsky [16; 17; 18; 19] sobre gramáticas generativas, reglas de producción y de reescritura comencé a buscar *representaciones lingüísticas de los objetos de la naturaleza y de fractales* en general y ya para finales de los 80´s contábamos con una serie de reglas gramaticales que representaban en forma independiente la estructura de arboles, montañas y nubes [20; 21; 22; 23; 2].

En un principio las producciones encontradas eran diferentes unas de otras pero conforme se siguió atacando el problema se detectó que en muchos casos la regla gramatical encontrada era un caso particular de una regla mas general.

En principio esto no fue tan extraño, ya que, por ejemplo se detectó que diferentes tipos de árboles generaban reglas gramaticales diferentes, pero que se podían agrupar en una regla gramatical mas general, por otro lado con las montañas sucedía algo parecido y otro tanto con las nubes, por lo que al final se tenia una producción general para árboles (para estructuras dendriticas) otra para montañas y otra para nubes. La sorpresa surgió cuando se detectó que a su vez todas esas producciones generales eran un caso particular de una regla gramatical mas general aun, que las integraba a todas [24].

Ya que a mediados de 1992 encontramos que la estructura gramatical de múltiples elemento de la naturaleza se puede ver como un caso particular de una regla general a la que llamamos Ecuación General o Fundamental de la Naturaleza y es de la forma *S --> e* S** donde la flecha --> indica que *S* se puede sustituir por *e* S** (En general una regla de reescritura α --> β se interpreta como que α se puede sustituir por β), *S* significa Sistema, *e* es cualquier elemento del sistema, *e** significa que se pueden tener tantos elementos como se quieran y *S** indica que el sistema se puede llamar tantas veces como se quiera. Observe que estamos utilizando el signo *** como un factor de repetición, lo cual no es una notación común dentro de las reglas de producción, pero en este caso facilita la representación.

Ahora bien conforme se ha seguido estudiando las propiedades de esta ecuación se ha encontrado que aunque la ecuación es general, la forma de los objetos depende del valor que toman sus parámetros, por lo que, *en este trabajo mostramos como generar diferentes tipos de objeto dependiendo del valor de sus parámetros y como encontrar y aplicar el espacio de caos (o sea el espacio en el cual se generan familias de objetos parecidos) asociado a cada tipo de objeto.*

Finalmente se ve que aunque esta ecuación se aplico inicialmente para representar y generar objetos naturales como árboles, montañas y nubes, conforme paso el tiempo fue quedando claro que su aplicación es mas general. Por lo que también en este trabajo *se generaliza el campo de aplicación de la ecuación y se muestra como se pueden representar también objetos clásicos de la teoría de Caos y Fractales como son los fractales de Dragón*.

1. DE LA ECUACIÓN AL PROGRAMA.

Ejemplos particulares de la ecuación fundamental son:
S --> e
S --> S
S --> e e
S --> e S
S --> S e
S --> e SS
S --> e eS
S --> e eSS
S --> e SS ... S
S --> e SeeSe... S

Como se puede ver la ecuación S--> e*S* representa objetos con estructura fractal y si se generaliza el tipo de proceso (e) que se puede llamar representa algoritmos recursivos.

El primer caso de la ecuación S--> e*S* se da cuando se tiene la ecuación *S --> e* **(**que se lee *el Sistema S llama a e***)** y hacer un programa a partir de este caso es directo, ya que la ecuación se representa verticalmente y se pasa directamente a pseudocodigo

```
Ecuación        Pseudocodigo
  S              S()
  -->            {
  e                e()
                 }
```

Un ejemplo muy simple de un sistema que cumple con la estructura S --> e es por ejemplo el programa

```
S()
{
   linea(x0,y0,long,w)
}
```

Donde el elemento *e* corresponde a la instrucción *linea(x0,y0,long,w)* o sea una rutina que dibuja una linea a partir de las coordenadas *(x0,y0)* de tamaño *long* y con un ángulo *w*.

Otro caso de la ecuación *S--> e*S** se da cuando se tiene la ecuación *S--> e** que representa un sistema formado por puros elementos (donde cada elemento puede ser un sistema completo). En particular se podría tener $S\text{-->}\ e_1e_2e_3...e_n$ o sea una ecuación cuyos resultados se obtienen al ejecutar e_1 seguido de $e_2, e_3,...$ y e_n.

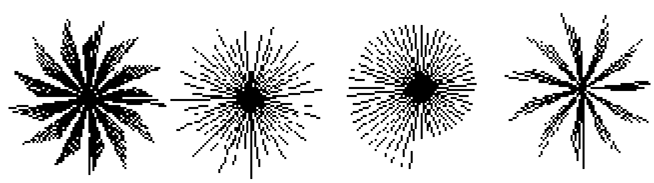
Como otro ejemplo se tiene la ecuación *S --> e S* que es equivalente a un sistema que genera un elemento y se llama recursivamente.

```
Ecuación        Pseudocodigo
  S              S()
  -->            {
  e                e()
  S                S()
                 }
```

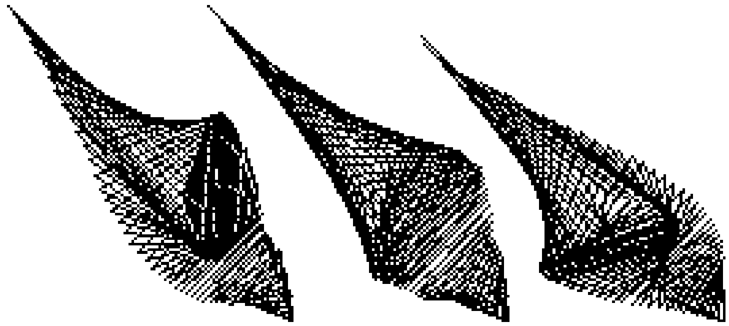
Esta ecuación engloba por ejemplo a la familia de las estrellas ya que la rutina

```
S(x0,y0,long,w)
{
  e(x0,y0,long,w)
  S(x0,y0,long,w+w1)
}
```

Gráfica una recta a partir de un punto (x_0, y_0), con un ángulo *w* y tamaño *t* y el ángulo se cambia entre llamadas; por lo que al final la rutina genera estrellas.



Por otro lado y con un cambio mínimo se genera la familia de los caracoles, para lo cual, únicamente se necesita que el punto inicial (x, y) de la nueva recta sea el punto final de la recta anterior.



Es importante observar que en el caso de un sistema recursivo el proceso continua indefinidamente, por lo que si esto se genera en una computadora es necesario introducir algún mecanismo que permita detener el proceso, como por ejemplo una tecla de Escape, o por otro lado detener el proceso cuando se cumpla alguna condición explícita como por ejemplo que se cumplió un número prefijado de llamados.

El siguiente caso particular de S--> e*S* es la ecuación *S--> eSS* que representa el llamado a un proceso (e) seguido de 2 llamados recursivos (SS):

En particular con esta ecuación se representa la estructura de árboles con 2 ramas, donde el elemento *e* tiene como función dibujar troncos o ramas. Para dibujar un árbol con tres ramas, simplemente se pondría una llamada recursiva más y así sucesivamente, construir árboles con tres, cuatro, cinco o más ramas se vuelve trivial, ya que su estructura queda:

S --> e SSS
S --> e SSSS
..
S --> e SS.....S

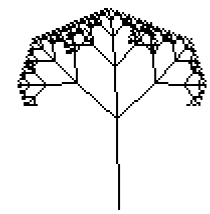
*y en general la estructura de cualquier árbol es de la forma S --> e S** y los programas son muy simples. Por ejemplo hacer un programa a partir de S--> eSSS es directo.

```
Ecuación        Pseudocodigo
  S             S()
  -->           {
  e               e()
  S               S()
  S               S()
  S               S()
                }
```

Para concretizar el proceso es necesario indicarle al sistema el tamaño del tronco (t), el lugar donde se graficara (x,y) y el ángulo entre las ramas (###) Por ejemplo la siguiente rutina

```
S(x, y, t, ###)
{
; dibuja   rama con   inicio en (x,y), tamaño t y ángulo
###
  e(x, y, t, ###)
;calcula los siguientes valores del sistema
  x1=x-t*cos(###)
  y1=y-t*sen(###)
  t1=t/1.7
; llama recursivamente al sistema
  S(x1, y1, t1, ###-47)
  S(x1, y1, t1, ###)
  S(x1, y1, t1, ###+47)
}
```

Dibuja árboles como



Observe que la ecuación general es independiente de parámetros, pero la ecuación para dibujar objetos con n ramas es una ecuación parametrizada de la forma
*S(x, y, t, ###)### e(x, y, t, ###) S(Fi(x), Gi(y), Hi(t), Ii(###))**

donde las funciones Fi, Gi, Hi, Ii son las funciones de transformación de los parámetros y pueden ser de cualquier forma. Por lo que, una representación mas completa para la rutina S() es de la forma

```
S(x, y, t, ###)
{
  e(x, y, t, ###)
  S(F1(x), G1(y), H1(t), I1(###))
  S(F2(x), G2(y), H2(t), I2(###))
  S(F3(x), G3(y), H3(t), I3(###))
}
```

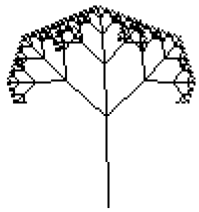
2.- ESPACIO DE CAOS Y PARÁMETROS.

2.1.- Espacio de Caos

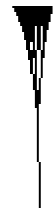
Dependiendo del comportamiento de las funciones se puede generar una gran cantidad de objetos. por ejemplo si tenemos la siguiente rutina

```
S(x, y, t, ###)
{
  e(x, y, t, ###) ;
  x1=x-t*cos(###) ;
  y1=y-t*sen(###)
  t1=t/1.7
  S(x1, y1, t1, ###+w1) ;
  S(x1, y1, t1, ###+w2)
  S(x1, y1, t1, ###+w3)
}
```

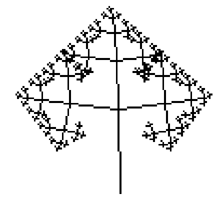
Todos los parámetros tienen el mismo comportamiento en los diferentes llamados excepto el ángulo. Si nos ponemos a jugar con el ángulo podemos encontrar que si $w_1 = -w_3$ y $w_2 = 0$ entonces se generan árboles simétricos con tres ramas como los mostrados en las siguientes figuras:



Árbol generado con
w_1=47, w_2=0, w_3=-47



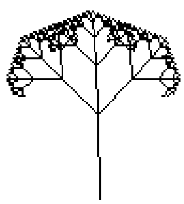
Árbol generado con
w_1=5, w_2=0, w_3=-5

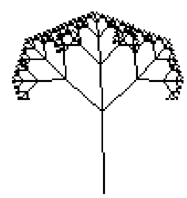


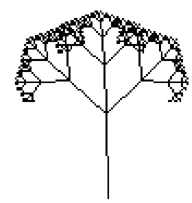
Árbol generado con
w_1=85, w_2=0, w_3=-85

Como se puede ver aun jugando con un solo parámetro los resultado pueden ser disimbolos, en un caso se pueden tener árboles "normales" (w ### 47), tipo cipreses (w ### 5) o araucaria (w ### 85).

Por otro lado si w es igual a 47 o 46 o 48 los árboles se parecen entre si.







Por lo que se tiene un espacio de ángulos donde el comportamiento de los árboles es similar. O sea que se puede encontrar un espacio de valores para los cuales los árboles que se generar son parecidos o pertenecen a la misma familia. A un espacio de este tipo se le llama *espacio de caos*.

Por ejemplo y sin meternos en mucha profundidad podríamos plantear que el espacio de caos donde la variable w toma valores entre [1, 7] es el espacio de los cipreses, [40, 60] es el espacio de los árboles "normales" y [75, 89] es el espacio de las araucarias.

2.2.- Generación de Espacios de Caos

La forma de construir computacionalmente un espacio de caos se reduce a manejar un generador de números pseudoaleatorios, que en principio tienen distribución uniforme, pero que puede tener prácticamente cualquier distribución.

Si se cuenta con una rutina random(n) que genera valores pseudoaleatorios entre 0 y n, generar un valor x en el rango [a, b] se reduce a calcular x= random(b-a)+a

Por ejemplo la rutina

```
S(x, y, t, ###)
{
  e(x, y, t, ###)
  x1=x-t*cos(###)
  y1=y-t*sen(###)
  t1=t/1.7
  S(x1, y1, t1, ###-(random(6)+1))
  S(x1, y1, t1, ###)
  S(x1, y1, t1, ###+(random(6)+1))
}
```

genera cipreses porque su espacio de caos esta entre [1, 7]. Pero si cambiamos el espacio de caos, cambia la familia de elementos generados.

Cuando se tiene un sistema de ecuaciones donde algunas de las ecuaciones tiene un comportamiento aleatorio se dice que se tiene un sistema caótico [25; 26; 27; 28; 29].

Aquí es importante distinguir entre *caos, orden y desorden* ya que es común que se confundan algunos de estos términos, por lo que entenderemos por sistema ordenado a un sistema que se puede describir mediante un conjunto de ecuaciones deterministicas. Cuando en el conjunto de ecuaciones una o varias de estas son ecuaciones que toman valores aleatorios se dice que se tiene un sistema caótico y cuando el número de ecuaciones aleatorias es "grande" (donde grande es un concepto ambiguo ya que en algunos casos puede significar 5 o 6 y en otros 40 o 50) o no se puede describir, o no se sabe como describir el sistema mediante ecuaciones, entonces se dice que se tiene un sistema desordenado.

En general se considera que *los sistemas naturales son caóticos, pero no siempre es fácil encontrar su descripción matemática, para encontrarla se tiene que encontrar la estructura del sistema (por ejemplo S### e*S*), el comportamiento de las funciones involucradas (por ejemplo e), los parámetros de estas funciones (por ejemplo x, y, t, ###) y los espacios de caos en los que viven la estructura, funciones y parámetros.*

Por ejemplo, la estructura de un árbol puede vivir en un espacio de caos, donde cada que se va a generar una rama existe cierta probabilidad de que no se genere, se muera, se genere o se sustituya por una hoja o flor. Las funciones pueden tener múltiples tipos de distribución y los parámetros tomar valores en espacios discretos o continuos, por lo que, como se ve, con muy poco esfuerzo se arma todo un "caos".

Por lo que, desde hace varios años en el área de los sistemas evolutivos [30; 31] se han desarrollando herramientas orientadas a lograr que el sistema encuentre su propio espacio de caos y lo actualice permanentemente. Estas herramientas parten de la premisa de que un sistema esta permanentemente evolucionando (o sea que no existe una etapa de "aprendizaje" y otra de aplicación) y que *por el puro hecho de que fluya la información el sistema se modifica.*

Para ejemplificar lo anterior podemos retomar el ejemplo de los árboles. mediante el enfoque evolutivo, originalmente el sistema esta vacío y cuando llega el primer árbol averigua sus parámetros y si el árbol es un ciprés entonces la maquina toma los parámetros de árbol como el punto medio de los cipreses. Cuando llega un segundo árbol lo mide con el ciprés y si no son similares pregunta que tipo de árbol es. Si le decimos que es un ciprés (o si originalmente encontró que era parecido al ciprés) recalcula la media y la varianza de los cipreses, tomando en cuenta los valores del nuevo árbol.

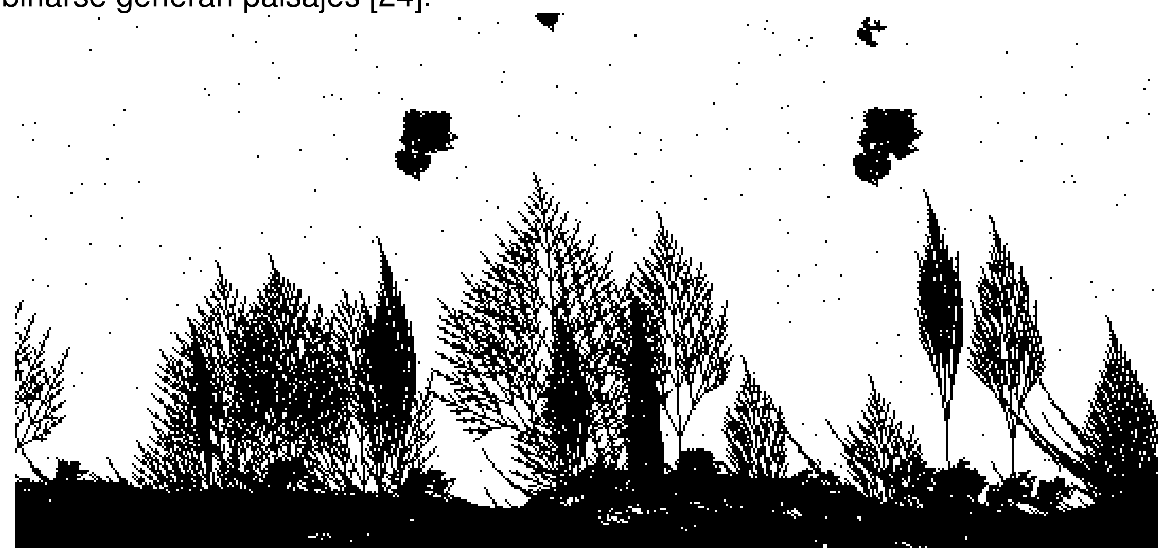
Si no es un ciprés o algún árbol ya reconocido (por ejemplo es una araucaria) habré un nuevo cumulo. Y así sigue permanentemente, cada que entra un nuevo elemento lo asocia con los cúmulos existentes o crea un nuevo cumulo, pero siempre esta cambiando y afinando los espacios de caos, con lo que el sistema evolutivo encuentra el patrón de caos.

3. PAISAJES Y DRAGONES.

3.1.- Paisajes.

Al continuar jugando con los parámetros del sistema (en este caso x,y,l,w) el efecto que se obtiene es precioso, ya que, cada uno de ellos al ser modificado puede ocasionar cambios radicales en la forma de los objetos generados, con lo que, dos objetos con la misma estructura interna (gramática) pueden tener una forma radicalmente diferente.

Por ejemplos si comenzamos a jugar con la rutina que genera árboles y hacemos que todas las ramas se inclinen en la misma dirección en lugar de un árbol se tienen laderas, si seguimos jugando con los parámetros podemos hacer que las ramas se enrollen con ángulos muy cerrados generando objetos como nubes y piedras. Con lo cual vemos que la misma ecuación con diferentes parámetros genera árboles, laderas y nubes y estos al combinarse generan paisajes [24].



Aquí es importante comentar que lo que se esta mostrando es una ecuación de Lingüística Matemática con la cual se pueden modelar y representar múltiples elementos de la naturaleza y los ejemplos de graficación son solo ayudas para visualizar.

3.2.- Dragones y Sistemas Conscientes.

Si continuamos analizando la ecuación general de la naturaleza S--> e*S* observamos que se puede sustituir e por un sistema recursivo A

S###A*S*

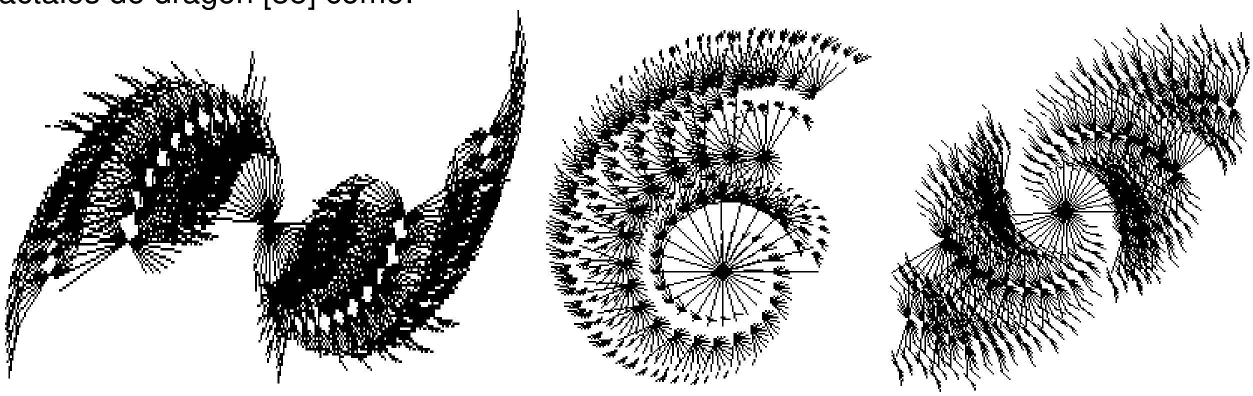
A###e*S*

Con lo que se tienen sistemas que presentan recursividad sobre la recursividad, con lo que, se habré un espacio de investigación enorme y que tiene por ejemplo aplicación en la Hermeneutica [32] o sea el área que busca el significado real de las cosas y maneja como herramienta un ciclo recursivo que consta del análisis del análisis del análisis.

Otra área donde la recursividad sobre la recursividad tiene aplicación es en el área de los sistemas conscientes, los cuales fueron planteados en 1988 por Cuitlahuac Cantu, Ariel Carboney y Raúl de la Rosa [33; 34].

Entre otras cosas en el área de los sistemas evolutivos planteamos que un sistema para ser consciente debe percibir lo que lo rodea, percibirse a si mismo y percibir que se percibe, La percepción de lo que lo rodea y la percepción de si mismo son según el investigador del Psicoanálisis Lacan dos casos particulares de un mismo proceso (como los dos lados de una cinta de Moevius son al final un solo lado) y son parte de los problemas que se atacan con los sistemas evolutivos. Pero percibir que se percibe es una doble recursividad.

También mediante los sistemas de recursividad sobre recursividad se pueden generar fractales de dragón [35] como:



donde cada elemento del fractal original genera a su vez un fractal completo.

CONCLUSIÓN.

En este trabajo se presento una propuesta de modelación de la realidad, basado en la integración de varios paradigmas básicos: los sistemas evolutivos, la ecuación general de la naturaleza y los sistemas caóticos.

En primer lugar se planteo que existe una ecuación de la forma *S--> e*S** basada en la Lingüística Matemática que nos muestra que múltiples fenómenos de la naturaleza tienen la misma estructura y que engloba a las ecuaciones que representan la estructura de troncos, caracoles, estrellas, árboles, nubes y montañas, entre otros.

A partir de ahí se vio que, se puede encontrar un espacio de valores conocidos como espacio de caos donde los objetos que se generan son parecidos o pertenecen a la misma familia.

Finalmente se mostró que si en la ecuación S-->e*S* se sustituye e por un sistema recursivo A se obtienen sistemas como:
S-->A*S*
A-->e*S*
que representan sistemas que tienen recursividad sobre la recursividad y que se aplican por ejemplo en la Hermeneutica, los Sistemas Conscientes y la generación de objetos clásicos de la teoría de Caos y Fractales como son los fractales de Dragón.

FUENTES DE INFORMACIÓN.

1. Przemyslaw **Prusinkiewicz**, Aristid **Lindenmayer** y James **Hanan**. *Developmental Models of Herbaceous Plants for Computer Imagery Purposes*. Computer Graphics, vol. 22, núm. 4, pag. 141, Agosto 1988.
2. Sofia **Bueno Peralta** y Antonio **Simancas López**. *Generador de Árboles Fractales.* en Memorias del III Congreso Nacional sobre Informática y Computación, Jalapa, Ver. México, Octubre 1990.
3. Peter E. **Oppenheimer**. *Real Time Design and Animation of Fractal Plants and Trees*. In Computer Graphics, vol. 20, núm. 4, SIGGRAPH'86, Agosto de 1986.
4. Heinz-Otto **Peitgen** and Dietmar **Saupe** (Editores). *The Science of Fractal Images*. Ed. Springer-Verlag, 1988.
5. Donald **Hearn** y M. Pauline **Baker**, *Gráficas por Computadora 2.- Ed.*, Editorial Prentice Hall, 1995, México.
6. Michael **Barnsley**, *Fractals Everywhere*, Ed. Academic Press, Inc., 1988, Londres.
7. Martin **Gardner**, *White and brown music, fractal curves, and one-over-f noise*, en Scientific American, abril de 1978.
8. Michael **F. Barnsley**, Arnaud **Jacquin**, Francois **Malassenet**, Laurie **Reuter**, Alan **D. Sloan**. *Harnessing Chaos for Image Synthesis*. en Computer Graphics, Vol. 22 Núm. 4, Agosto 1988.
9. Przemyslaw **Prusinkiewicz**, Mark **James**, Radomín **Mech**, *Synthetic Topiary,* en Computer Graphics Proceedings, SIGGRAPH 94, pag 351.
10. Todd **Reed**, Brian **Wyvill**, *Visual Simulation of Lightning,* en Computer Graphics Proceedings, SIGGRAPH 94, pag 359.
11. Deborah **R. Fowler**, Przemyslaw **Prusinkiewicz**, Johannes **Battjes**, *A Collision-based Model of Spiral Phillotaxis,* en Computer Graphics Proceedings, SIGGRAPH 92, pag 361.
12. Deborah **R. Fowler**, Hans **Meinhardt**, Przemyslaw **Prusinkiewicz**, *Modeling seashels,* en Computer Graphics Proceedings, SIGGRAPH 92, .pag 379.
13. Michael **Kass**, Gavin **Miller**, *Rapid, Stable Fluid Dinamics for Computer Graphics,* en Computer Graphics, SIGGRAPH 92, vol. 24 núm. 4, pag 49, Agosto 1990
14. Robert Marshall, Rodger Wilson, Waine Carlson, *Procedure Models for Generating Three-Dimensional Terrain,* Computer Graphics, SIGGRAPH 80 vol. 14 núm. 8, julio 1980
15. Nelson L. Max, *Vectorized Procedural Models for Natural Terrain Waves and Islands in the Sunset,* en Computer Graphics, SIGGRAPH 81, vol. 15 núm. 3, Agosto 1981.
16. Noam **Chomsky**. *Estructuras Sintácticas.* Ed. Siglo XXI
17. Emmon **Bach.** *Teoría Sintáctica.* Ed. Anagrama.
18. **Salomaa**. *Formal Languages*. Ed. Academic Press.
19. **Hopcroft** y **Ullman**, *Formal Languages and Their Relation to Automatas*, Ed. Addison-Wesley, 1969.
20. Fernando **Galindo Soria**, *El Continuo Dimensional: Un Universo Fractal,* Conferencia en el Congreso Nac. de Egresados de Física y Matemáticas, 1989, La Trinidad Tlaxcala.
21. Fernando **Galindo Soria.** *De Fractales y otros Bichos: La Matemática de la Naturaleza (Rumbo a la Matemática Informática),* en Memorias del VI Congreso Nacional sobre Informática y Computación, Octubre de 1993, Mérida, Yucatán.

22. Fernando **Galindo Soria**, *Sistemas Evolutivos de la Naturaleza,* Conferencia Magistral en el Congreso Nacional de Ingeniería en Sistemas Computacionales CONAISCO-90, en Hermosillo Son., abril de 1990
23. Fernando **Galindo Soria.** *Aplicación de la Lingüística Matemática y los Fractales a la Generación de Imágenes.* en Memorias del Simposium Nacional de Computación. México, Nov. de 1991.
24. Fernando **Galindo Soria**, *Aplicación de la Lingüística Matemática a la Generación de Paisajes*, en Memorias del Simposium Internacional de Computación, IPN-CENAC, México 1996.
25. Edmundo **Flores**, *La creación de la nueva ciencia del caos y el ocaso de la Meteorología y de la Econometrìa*. en El Búho tomo IV, núm. 26687, 15 de Julio de 1990.
26. Monique **Dubois**, Pierre **Aften** y Pierre **Bergé**, *El Orden Caótico*, en Mundo Científico vol. 7, núm. 68, pag. 429, Ed. Fontalba, S.A., Barcelona, España.
27. Etienne **Guyon**, Jean-Pierre **Hulin** e Yves **Pomeau**, *La receta de la mezcla: un poco de caos*, en Mundo Científico vol. 7, núm. 75, pag. 1244, Ed. Fontalba, S.A., Barcelona.
28. Hermann **Haken** y Arne **Wunderlin**, *El caos determinista*, en Mundo Científico vol. 10, núm. 108, pag. 1215, Ed. Fontalba, S.A., Barcelona, España.
29. Julio **M. Ottino**, *Mezcla de Fluidos,* en Investigación y Ciencia, núm. 150, pag. 44, marzo de 1989.
30. Fernando **Galindo Soria**. *Sistemas Evolutivos*. en Boletín de Política Informática. México, Septiembre de 1986.
31. Fernando **Galindo Soria**. *Sistemas Evolutivos: Nuevo Paradigma de la Informática.* en Memorias XVII Conferencia Latinoamericana de Informática, Caracas Venezuela, julio de 1991.
32. Marcela G. **Bueno Días B.**, *Hacia una Hermeneutica Informática*, memorias del Congreso Iberoamericano de Informática Educativa, tomo II, pag. 290, Santo Domingo, Rep. Dominicana, junio de 1992.
33. Ariel **Carboney Martínez**. *Del Desarrollo de Sistemas de Información Tradicionales a la Generación Inferencial de Conciencia Autónoma Espontanea*. en Teoría y Práctica de los Sistemas Evolutivos, Edit. Jesús Olivares Ceja, México, 1997.
34. Cuitlahuac **Cantu**, Ariel **Carboney** y Raúl **de la Rosa,** *Comunicación personal sobre el área de los sistemas conscientes,* 1988. Cd, de México.
35. Clifford **A. Pickover**, *Pattern Formation and Chaos in Networks,* en Communications of the ACM, vol. 31 núm. 2, pag. 136, Febrero de 1988.