

EVOLUCIÓN Y SISTEMAS EVOLUTIVOS

Fernando Galindo Soria

www.fgalindosoria.com

fgalindo@ipn.mx

www.laredi.com

Paradigma Evolutivo

La evolución, el crecimiento, el aprendizaje, la vida, el pensamiento, la transformación de nuestra imagen de la realidad, los procesos de descomposición, el desarrollo y transformación de las empresas, sociedades, organizaciones, países, galaxias, universos, etc., son manifestaciones de un mismo proceso general de transformación o cambio al que por facilidad llamamos evolución.

21 de Septiembre del 2002, Octubre del 2009

Resumen

Se presenta un concepto general de *los Sistemas Evolutivos, como sistemas capaces de transformarse permanentemente a partir de los flujos de materia, energía e información que los cruzan*. Para lo cual primero se vera su relación con los sistemas formales y con los sistemas que aprenden. Se mostraran algunos ejemplos de aplicación al tratamiento de lenguajes, reconocimiento de formas, representación del conocimiento, etc. Finalmente se vera el *paradigma evolutivo*, en el que *se plantea que la evolución, el crecimiento, la vida, el aprendizaje, el pensamiento, la transformación de nuestra imagen de la realidad, los procesos de descomposición, el desarrollo y transformación de las empresas, sociedades, organizaciones, países, galaxias, universos, etc., son manifestaciones de un mismo proceso general de transformación o cambio, al que por facilidad llamamos evolución*.

INTRODUCCIÓN

Los Sistemas Evolutivos (Sistema Evolutivo) surgieron como una respuesta a la necesidad de desarrollar sistemas de información (Por ejemplo: nóminas, sistemas expertos, compiladores o sistemas de reconocimiento de imágenes) que reflejaran lo más fielmente posible la realidad que están modelando y capaces de soportar y absorber en tiempo real los cambios que ocurren en ésta, ya sea en sus elementos, en las relaciones entre éstos o en su significado.

Para lograr lo anterior un sistema evolutivo se comporta como un niño que está aprendiendo y aplicando este aprendizaje a su entorno, ya que de entrada, *el Sistema Evolutivo no cuenta con reglas o programas que le digan cómo resolver un problema dado, sino que cuenta con la capacidad de construir su propia imagen de la realidad y con los mecanismos que le permiten percibir esa realidad y actuar dentro de ella*.

El área de los sistemas evolutivos es relativamente nueva, ya que, los primeros trabajos orientados en esta dirección se empezaron a desarrollar en 1983, las primeras conferencias sobre el tema se presentaron en 1984 y los primeros productos funcionando a nivel de prototipo se presentaron en 1985; a partir de ese inicio su crecimiento ha sido explosivo y cada día aumenta la cantidad de productos desarrollados bajo este enfoque al mismo tiempo que se integran nuevos métodos y herramientas para construir sistemas evolutivos.

La variedad de aplicaciones resueltas mediante este enfoque es completamente disímbola y cada vez se amplía más su campo de aplicación haciéndonos pensar que este enfoque marca una nueva forma general o Paradigma de la Informática.

Entre otras se han desarrollado aplicaciones para:

.Generar Esquemas Lógicos de Base de Datos a partir de Lenguaje Natural.

- .Reconocimiento de Imágenes.
- .Generación de Sistemas Expertos.
- .Construcción y explotación de Bases de Conocimiento.
- .Control de robots usando lenguaje natural y de trayectoria.
- .Reconocimiento y corrección de errores ortográficos.
- .Generación automática de sistemas de Información.
- .Construcción de paisajes.

A pesar de no ser exhaustiva, la lista anterior nos permite darnos cuenta de la diversidad de aplicaciones atacadas y por otro lado conviene mencionar que en la mayoría de estas aplicaciones se encuentran presentes herramientas para tratamiento de lenguaje natural, reconocimiento de patrones y representación de conocimiento.

1 INTRODUCCIÓN A LOS SISTEMAS EVOLUTIVOS

Uno de los problemas más graves de la Informática actual se presenta por su poca capacidad para modelar en tiempo real los fenómenos que ocurren en la realidad, ya que es común que cuando un Sistema de Información: Nómina, Compilador, Reconocedor de Imágenes, Sistema de Inventarios, Sistema Experto de Diagnóstico Médico, etc. se libera, ya prácticamente es obsoleto, ya sea porque:

- 1) *El problema modelado se modificó.*
- 2) *Porque el modelo no cubrió los aspectos esenciales.*
- 3) *O simplemente la información y el conocimiento que se tiene sobre el tema ha quedado rebasado por algún nuevo dato o hecho no conocido previamente.*

En general se pueden plantear tres grandes problemas que ayudan a la rápida obsolescencia de los sistemas de información y que obligan a realizar en forma continua lo que se conoce como mantenimiento adaptativo (aquel que se realiza para lograr que el sistema mantenga una imagen lo más cercana al ambiente que se requiere modelar) o por el contrario obliga a los usuarios a lidiar con una herramienta cada vez menos poderosa y más alejada de la realidad que aparentemente está modelando.

El primer problema surge cuando se desarrollan sistemas que modelan fenómenos altamente cambiantes, como por ejemplo los sistemas de algo tan cotidiano como la Nómina, donde prácticamente no se ha terminado de desarrollar cuando ya se tiene que modificar (y no es por mal desarrollo, sino simplemente porque se está tratando de modelar un problema que cambia prácticamente por decreto y en el que es difícil, establecer los patrones de cambiado).

El segundo problema se presenta cuando se desarrollan sistemas que modelan fenómenos para los cuales no existe una regla o patrón ya establecido y bien manejado y del cual se pueda tener un algoritmo en forma relativamente sencilla, tal es el caso de muchos de los problemas actuales de la Inteligencia Artificial. por ejemplo en el reconocimiento de patrones se ha desarrollado gran cantidad de métodos y algoritmos pero sin embargo los problemas no resueltos son cada día mayores.

El tercer problema es un cuello de botella de la Informática que se presenta cuando los sistemas funcionando son incapaces de reflejar la realidad por que los datos y hechos de ésta no son directamente accesibles por el sistema, ya sea porque son difíciles de obtener (por ejemplo, topografía detallada del terreno donde se construirá una carretera) o porque cambian tan rápidamente que los métodos tradicionales de captura no permiten mantenerlos actualizados (por ejemplo, los datos clínicos de un paciente de terapia intensiva).

Es por lo anterior que es necesario replantear el enfoque utilizado para resolver problemas en Informática (representado por áreas como el Desarrollo de Sistemas, la Ingeniería de Software y la Ingeniería de Conocimiento) en el cual la tendencia es a la construcción de sistemas estáticos e incapaces de automantenerse y buscar métodos y herramientas capaces de recrear en forma continua su imagen de la realidad o del problema a resolver. Es dentro de este contexto donde surgen los Sistemas Evolutivos.

2 ARQUITECTURA GENERAL DE UN SISTEMA EVOLUTIVO

Cuando se construye un sistema evolutivo se debe tener en cuenta que se está desarrollando un sistema que debe ser capaz de construir su propia imagen de la realidad, con lo cual, *se da un giro radical a la forma de desarrollar sistemas*, ya que en los métodos tradicionales una persona o grupo de personas analizan un problema y proponen un conjunto de reglas para resolverlo, o sea que, el desarrollador estudia la realidad, construye una imagen de ésta y la representa mediante un programa, con lo cual, si por algún motivo el problema atacado cambia, es necesario que el desarrollador vuelva a estudiarlo e introduzca los cambios al sistema, teniendo siempre una estructura monolítica y de mutua esclavitud entre desarrollador y sistema, ya que cualquier cambio en la realidad obliga al desarrollador a introducirlo al sistema, so pena de quedar obsoleto.

Por el otro lado mediante los sistemas evolutivos se busca que *sea el propio sistema* el que lleve a cabo acciones que le permitan construir su imagen de la realidad, mantenerla actualizada y usarla para interactuar con el medio.

Por lo que, *cuando se desarrolla un Sistema Evolutivo más que darle un conjunto de reglas prefijadas para resolver un problema, lo que se busca es darle la capacidad para que pueda construir y mantener su propia imagen de la realidad.*

2.1 De los Sistemas Formales a los Sistemas Evolutivos pasando por el Teorema de Gödel

2.1.1 Sistemas Formales

Cuando un sistema aplica un conjunto de reglas o instrucciones para resolver un problema se dice que es un *Sistema Formal*, y a los métodos que construyen programas de este tipo se les conoce como métodos de programación deductiva o formal.

Un sistema formal tiene un conjunto de reglas que nos permiten resolver un problema. Cuando hacemos un programas en Basic, C o Java, lo que se esta haciendo es un sistema formal, se le esta diciendo a la computadora, si aplicas esta regla, o esta otra, vas a resolver el problema.

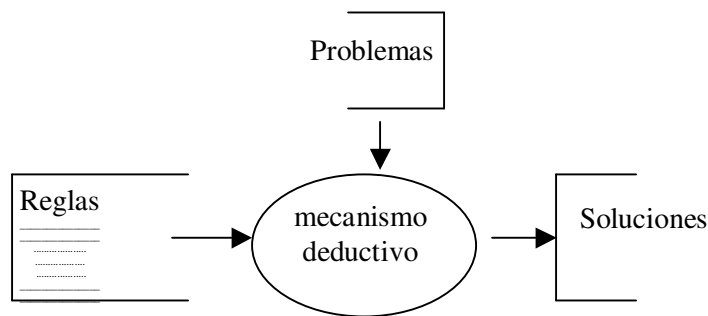


Figura 1. Sistema Formal

Los sistemas formales son los que mas se usan para programar, son muy usados y muy buenos, el problema surge cuando existe algún problema que no puedan resolver, por ejemplo, supóngase que hacemos un programa que reconoce la letra A, en el programa deberían de incluirse las características de la letra A, de modo que identificara A's similares, pero si se quiere reconocer una B no la reconocería , o tampoco reconocería la letra a minúscula, a cuadrada y otras A's incluso, a menos que explícitamente se le diga como hacerlo, lo cual a la larga se vuelve muy pesado, porque siempre existe algo no cubierto por las reglas que tiene el sistema forma.

2.1.2 Sistemas que aprenden

El problema de la programación deductiva o de sistemas formales es que, se orienta a desarrollar sistemas fijos difíciles de modificar en tiempo real (ya que las modificaciones involucran reprogramar el sistema), por lo que, desde principios de los 60, se ha buscado desarrollar herramientas automatizadas capaces de obtener en forma automática el conjunto de reglas del sistema a partir de ejemplos y descripciones generales de un programa, en lo que se conoce como Machine learning o maquinas o sistemas que aprenden.

A una herramienta o programa capaz de encontrar un conjunto de reglas a partir de ejemplos la conocemos como Mecanismo o *Herramienta Inductiva*; el esquema de una herramienta inductiva es el mostrado en la figura 1.

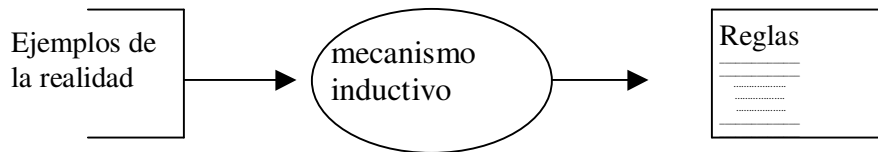


Figura 1. Herramienta inductiva

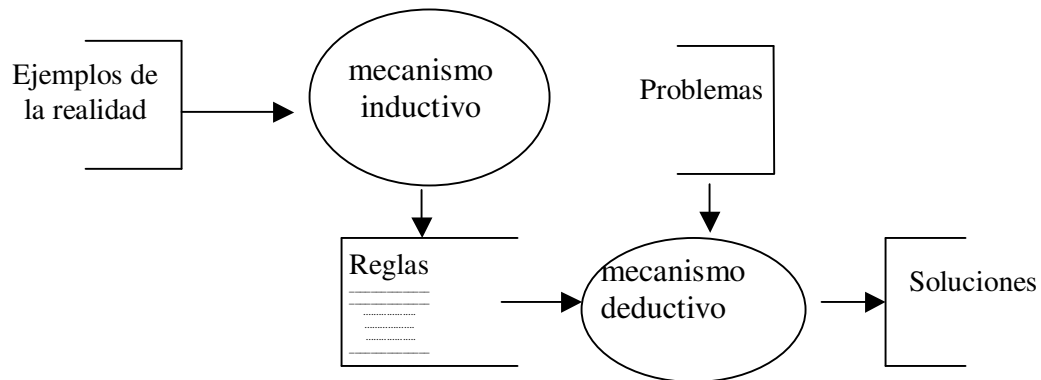
Lo anterior no significa que un sistema formal no sea útil , por el contrario nos han resuelto un gran numero de problemas, por que la forma común de programar es mediante sistemas formales, hagamos de cuenta que los sistemas formales es como la teoría de Newton con respecto a la teoría de Einstein.

La física de Newton la usamos para muchas cosas, por ejemplo, cuando calculamos la velocidad de un proyectil no usamos la teoría de Einstein, con Newton de hace cientos de años nos funciona muy bien, si quiero ver como rebota una pelota, con Newton nos basta, si queremos saber el comportamiento de el proyectil cuando se acerca a la velocidad de la luz entonces ya entramos con Einstein; los sistemas formales son así , para espacios normales restringidos convencionales funcionan muy bien.

El problema es que ya no estamos en esos espacios, los problemas a los que ahora nos enfrentamos ya rebasan un poco a esa cotidianidad, los informáticos rompieron esa cotidianidad desde los años 50 con el surgimiento de la Inteligencia Artificial, cuando empezaron a enfrentar problemas que ya no eran tan fáciles de resolver utilizando sistemas formales, y que requerían de otro tipo de tratamiento, por ejemplo un problema de esa época era el reconocimiento de letras.

En los años 60 surge el área de las Machine learning (Sistemas que aprenden), la idea de una Machine learning es que se invierte la situación, en lugar de que yo me preocupe de darle las reglas para que reconozca un objeto, le doy la capacidad de que reconozca o aprenda como esta hecho el objeto, los sistemas Machine learning tienen un programa al cual por ejemplo le damos una A y el programa se las arregla para encontrar la estructura de esa A. Estos programas construyen un archivo o un sistema de

ecuaciones dónde están representadas el conjunto de reglas que representan a esa A. Lo bonito de un Machine learning es que uno le puede dar por ejemplo una A cuadrada, rectangular, o chiquita y va construyendo la representación de todas esas A's .



Machine learning (Sistemas que aprenden) años 60

Los Machine learning se empezaron a desarrollar desde los años 60, se le daba a la computadora un montón de ejemplos de aquello que quisiéramos que reconociera y encontraba las reglas que describían ese tipo de cosas o sistemas, y ya que tenían ese conjunto de reglas, se las pasaban a un programa formal al que por ejemplo se le daba una A cuadrada y entonces la computadora la reconocía como A cuadrada, lo que nos ahorrábamos era el trabajo de estar encontrando las reglas, El libro básico “Learning Machine” de Nilsson se escribió en 1965 .

El problema de los Machine learning es que tiene dos grandes etapas: una “de aprendizaje” y otra Formal. Por lo que solo resuelve problemas que previamente se le enseñó a resolver, pero cuando llegan problemas que previamente no aprendió no los reconoce, si por ejemplo se le dan un montón de ejemplos de A's reconoce A's, pero si posteriormente se me ocurre colocar una A Itálica, ya no la reconoce si previamente no se le enseñó a reconocer ese tipo de A's.

El problema tanto de los sistemas formales como de los sistemas que aprenden o machine learning es que tarde o temprano les llegara un problema que no pueden resolver.

2.1.3 Teorema de Gödel

En 1930 Kurt Gödel estableció un teorema que lleva su nombre, el teorema de Gödel puede considerarse como un teorema fundamental en informática. El teorema dice en general que: un sistema Formal, es incompleto, o inconsistente, un sistema formal es incompleto cuando tarde o temprano, no importando el número de reglas que tenga, va a existir un problema que se supone debería poder resolver y que no puede resolver, o sea le faltan reglas para poderlo resolver, e inconsistente significa que tarde o temprano, aplicando un mismo grupo de reglas encuentra una solución verdadera y una falsa para el mismo problema.

Si se logra que un sistema formal sea completo será inconsistente y si es consistente entonces será incompleto.

2.1.4 Sistemas Evolutivos

Este tipo de problema nos mantuvo ocupados durante los años setenta y principios de los ochenta, observamos en aquellos años que, cuando se requiere hacer una modificación al código, muchas veces los programas tienen miles de instrucciones y resulta poco práctico y tedioso de realizar cambios en la programación por la complejidad de la codificación, y tomando en cuenta que se realizan actualizaciones y modificaciones frecuentemente se tiene que estar reestructurando continuamente, de donde salió la idea de hacer un sistema que se encargara de hacer las modificaciones al código automáticamente y asimismo que éste se modificara a si mismo.

De donde la idea sería utilizar un programa formal que contuviera las reglas por ejemplo para reconocer una A, y que cuando este no reconociera un tipo de especial de A, mandara llamar a otro tipo de programa que se encargara de aprender, lo que se convertiría posteriormente en el principio o la primera idea de lo que es un **sistema evolutivo o sea un programa que permanentemente esta aprendiendo y reestructurando sus reglas** de donde el programa no mandaría Fatal error o warnings, simple y sencillamente si el programa no sabe, aprende.

Entonces la primera idea fue introducirle un mecanismo de retroalimentación a un sistema que aprende obteniendo los primeros sistemas evolutivos: O sea que para contar con un Sistema Evolutivo, lo único que hace falta es dotar las herramientas con la capacidad de actualizar su imagen de la realidad en tiempo real, con lo que se tiene la arquitectura de la figura 3.

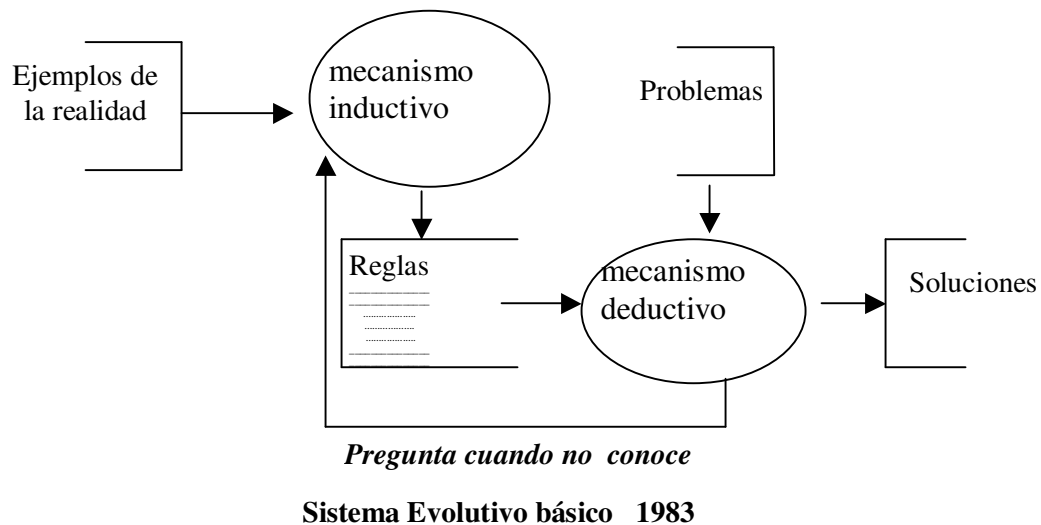


Figura 3. Retroalimentación para actualizar la imagen de la realidad

Generalizando se tiene la siguiente propuesta de Arquitectura de un Sistema Evolutivo en la figura 4.

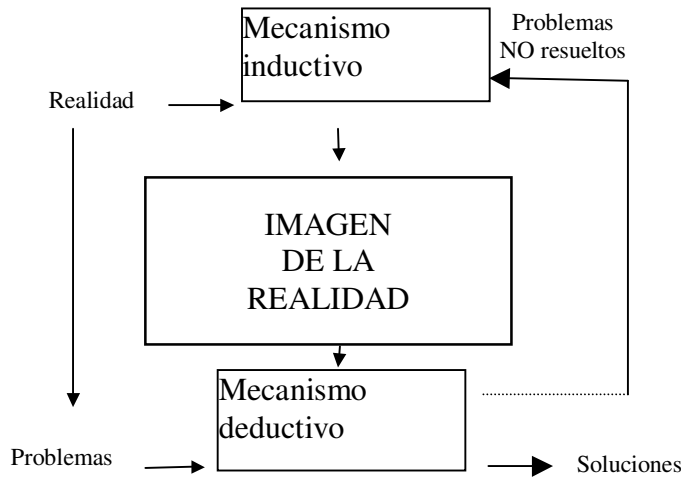
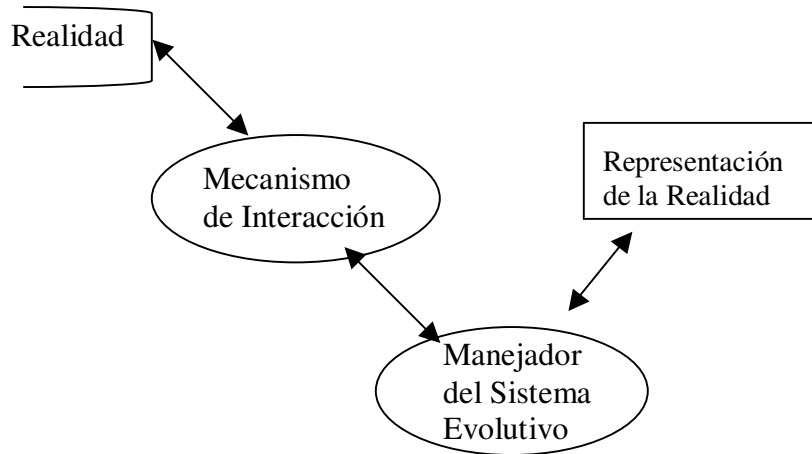


Figura 4. Propuesta de Arquitectura de un Sistema Evolutivo

A partir de esta propuesta se han desarrollado otras y en la actualidad se cuenta con varios enfoques de cómo construir este tipo de sistemas y en general la arquitectura de un Sistema Evolutivo se puede ver constituida por tres grandes módulos interrelacionados.

- .La Representación de la Realidad.
- .El Manejador del Sistema Evolutivo
- .Los Mecanismos de Interacción con el Ambiente.



Sistema Evolutivo 1991

Figura 6. Arquitectura General de un Sistema Evolutivo

El módulo de *Representación de la Realidad* nos permite almacenar una imagen de una realidad dada en término de sus elementos, las relaciones entre éstos o estructura y su significado.

El *Manejador del Sistema Evolutivo* es el mecanismo responsable de construir, mantener actualizada y aprovechar la Imagen de la Realidad a partir de los conocimientos y requerimientos detectados por el Mecanismo Interactivo.

Por su parte *el Mecanismo de Interacción* es el que permite que el sistema evolutivo se comunique con su ambiente ya sea para modificar su imagen de la realidad o llevar a cabo alguna acción sobre su exterior.

Las primeras estructuras de sistemas evolutivos realmente eran compiladores modificados para que cuando encuentran una regla que no esta dentro de la gramática modificara la gramática, la modificación es que, cuando el programa no reconoce una estructura no mande fatal error, si no que modifique la gramática, de donde obtendríamos un sistema evolutivo relativamente sencillo, concepto que ha crecido desde 1983.

4 ENFOQUE EVOLUTIVO

Con el surgimiento de los sistemas evolutivos se tiene un nuevo enfoque para el desarrollo de sistemas, o sea que un problema se puede resolver mediante los enfoque de los sistemas formales, sistemas que aprenden o sistemas evolutivos, Por ejemplo se puede hacer un sistema que reconozca la letra A usando el enfoque de los sistemas formales, donde se le da al sistema un conjunto de reglas que representan completamente a esa A especifica y el sistema reconoce la letra A, también se puede usar el enfoque de los sistemas que aprenden y hacer un sistema que aprenda a reconocer letras (por ejemplo usando redes neuronales, algoritmos genéticos, etc.) o usar el enfoque evolutivo.

Es un enfoque no una herramienta, por ejemplo existen redes neuronales usando el enfoque formal, el enfoque que aprende o el enfoque evolutivo.

Muchas redes neuronales y algoritmos genéticos son sistemas que aprenden

El enfoque evolutivo se puede aplicar prácticamente con cualquier herramienta incluyendo redes neuronales, algoritmos genéticos y sistemas lingüísticos

*En general los sistemas reales no son formales son evolutivos,
por lo que la idea es que los sistemas de informáticos también sean evolutivos*

A continuación mostraremos varios ejemplos de construcción de sistemas evolutivos usando principalmente herramientas lingüísticas y matrices evolutivas

5 SISTEMAS EVOLUTIVOS DE REESCRITURA

Los sistemas evolutivos mas simples se conocen como sistemas de reescritura, son sistemas muy elementales, a los cuales se les dan entradas que son buscadas por la computadora en archivos que tienen dos columnas, este busca la entrada en la primera columna y si encuentra esa entrada le asocia su significado, pero si no encuentra la entrada entonces pregunta y uno le da el significado, un ejemplo es un sistema elemental traductor de Idiomas, si le damos la entrada "Hola Perro", si la computadora no encuentra en su lista esta frase, entonces pregunta, ¿ que es "hola perro "? , y entonces le decimos "Hello Dog", entonces la computadora almacena en su primera columna "Hola perro" y en la otra "Hello Dog", la siguiente vez que introduzcamos "Hola Perro" la computadora va a decir "Hello dog".

Una de las herramientas más poderosas y desconocidas de la matemática actual son las reglas de reescritura, descubiertas dentro de la lingüística matemática y aplicadas inicialmente en esta área y más adelante para la construcción de compiladores y en general de sistemas dirigidos por sintaxis.

La fuerza de esta herramienta se ha diluido, porque normalmente solo se aplica como parte de las gramáticas generativas y en particular como reglas de producción, a pesar de que cualquier sistema, sistema formal o sistema evolutivo puede manejar como herramienta de representación a estas reglas. En general cualquier problema de Informática, Computación o Inteligencia Artificial se puede representar en término de reglas de reescritura (es como si el operador de suma sólo se usara para sumar números naturales, hasta que se libera del entorno de los números naturales adquiere toda su importancia).

Su fuerza es tan grande que valdría la pena que el uso de estas reglas se introdujera sin mucho formalismo desde el nivel de primaria y secundaria, como una herramienta mas de la Matemática, tan importante como la suma y la resta, ya que abre a los alumnos todo un nuevo universo.

5.1 Conceptos Generales

Como primer punto partiremos de que un *sistema de reescritura* $\langle A, B, R \rangle$ esta formado por un conjunto de elementos de entrada **A**, un conjunto de elementos de salida **B** y un conjunto de reglas de reescritura **R**. (Donde **A** y **B** pueden ser iguales o diferentes).

Una *regla de reescritura* es de la forma:

X --> **Y**

Donde $X \in A^*$, $Y \in B^*$ (A^* y B^* representan respectivamente las cerradura de A y B). Y lo anterior se lee como: *X se puede reescribir como Y* y significa que **X** se puede sustituir por **Y**.

Es relativamente fácil construir un sistema de información que representa a los sistemas de reescritura, por ejemplo el siguiente sistema formado por un programa y un archivo con dos columnas :

A1	B1
A2	B2
""	""
An	Bn

```
Programa()
{i=1
 lee Ax
 mientras ((Ax != Ai) y (no fin de archivo))i++
 si (no fin de archivo) escribe Bi
 sino escribe "no reconozco Ax"
}
```

El archivo representa al conjunto de reglas de reescritura:

```
A1 --> B1
A2 --> B2
""
An --> Bn
```

Y el programa representa a un proceso que reescribe A_i como B_i .

5.2 Presentación de la Herramienta General.

Esta idea se puede aplicar por ejemplo para hacer un traductor automático, donde se puede tener una tabla que contiene en la primera columna el texto en un idioma y en la segunda su traducción a otro idioma.

texto	traducción
este es un perro	this is a dog
el perro blanco	the white dog
""	""
el perro negro	the black dog

Y un pequeño programa que realiza la sustitución de reglas.

```
Programa()
{
  i=1
  lee Textox
  mientras ((Textox != Textoi) y (no fin de archivo))i++
  si (no fin de archivo) escribe Traduccióni
  sino escribe "no conozco el texto"
}
```

Como se puede ver el sistema es pequeño y en general trivial de modificar, ya que el conocimiento queda separado del programa, al estar almacenado en registros, por lo que, si se encuentra una nueva regla lo único que se requiere es aumentar un registro en el archivo y el programa no se toca.

Aun mas, se puede hacer que el mismo programa actualice el archivo, de tal manera que si entra un texto desconocido, el programa pueda preguntar (al usuario o a un experto) la traducción asociada, con lo que se tendría un pequeño sistema evolutivo.

texto	traducción
este es un perro	this is a dog
el perro blanco	the white dog
""	""
el perro negro	the black dog

```
Programa()
{
  i=1
  lee Textox
  mientras ((Textox != Textoi) y (no fin de archivo))i++
  si (no fin de archivo) escribe Traduccióni
  sino //entra al dialogo
  escribe "desconozco Textox dame su traducción"
}
```

```

lee Traducciónx
almacena en el archivo Textox , Traducciónx
}

```

Este programa es muy simple de hacer y tiene la ventaja de que no requiere tener almacenado el conocimiento previamente, ya que si el archivo estuviera vacío y se preguntara por **Texto_x** el programa pediría **Traducción_x** y ya tendría la regla **Texto_x -> Traducción_x** con lo que, *el sistema evolutivo puede empezar ha construir la base de conocimiento y 'aprender' desde cero, en tiempo real y fácilmente.*

Una gran cantidad de problemas aparentemente diferentes se pueden reducir en primera instancia al mismo sistema, ya que prácticamente se puede atacar cualquier problema que se pueda representar como una cadena de bits o caracteres, como por ejemplo *el reconocimiento de imágenes, texto, traductores, sistemas experto, señales biofísica, voz, etc.*

DEL SISTEMA DE INFORMACIÓN AL SISTEMA EVOLUTIVO

En general se considera que un Sistema de Información o programa de cómputo tiene la arquitectura de la figura 1.



Figura 1. Arquitectura de los sistemas de información en los 50

Sin embargo este esquema es muy viejo y corresponde a los años 50's

Ahora bien cuando un programa sigue un conjunto de reglas o instrucciones para resolver un problema se dice que es un Programa Deductivo, y a los métodos que construyen programas deductivos se les conoce como métodos de programación deductiva.

El problema de la programación deductiva es que se orienta a desarrollar sistemas fijos difíciles de modificar en tiempo real (ya que las modificaciones involucran reprogramar el sistema).

Ya para finales de los 70's se manejaba un modelo generalizado en el cual se considera que cualquier sistema de información tiene la arquitectura de la figura 3.

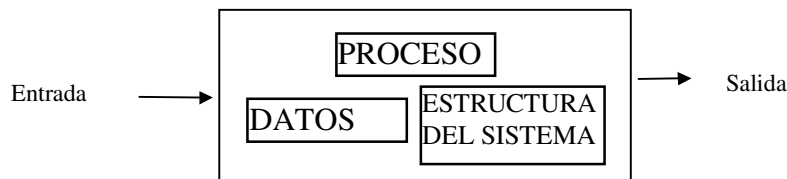


Figura 3. Arquitectura de los sistemas de información en los 70s

El desconocimiento o el hecho de no tomar en cuenta este esquema cuando se desarrollan sistemas es una de las causas principales por las que los sistemas se vuelven altamente estáticos y difíciles de mantener ya que en los sistemas y programas tradicionales las tres componentes se encuentran revueltas, por lo que, un cambio "pequeño" en los datos o procesos o en el orden de atacar un problema ocasiona que prácticamente se tenga que volver a programar todo; por otro lado si se desarrolla el sistema de tal manera que los datos

queden en un lado, los procesos en otro y finalmente la estructura del sistema en otro, el proceso de actualización puede ser relativamente fácil (un caso particular de este enfoque es el del desarrollo de Bases de Datos).

Por lo que se considera que una característica fundamental que se debe buscar cuando se desarrolla un sistema de información es la de que exista una *Independencia Relativa* o sea que los datos, procesos y estructura del sistema queden separados y únicamente exista la relación mínima necesaria entre las tres componentes.

La Independencia Relativa es una de las características distintiva del método de desarrollo basado en Base de Datos y en este documento simplemente se extiende al esquema general de Sistemas de Información.

Otra característica presente en el método de base de datos que se puede extender al desarrollo de sistemas es el hecho de que en el enfoque de base de datos se distinguen explícitamente tres niveles, en el primero se encuentran los datos en sí (nombres propios, edades concretas, direcciones específicas, etc.); en el segundo se tiene la estructura de datos específica donde se almacenan los datos (archivos, registros y campos de la base de datos que se esta manejando) y en el tercer nivel se tiene un programa que se encarga de construir una estructura de datos particular, a partir de una descripción general de la base de datos, este programa es el constructor de la base de datos.

En realidad una base de datos no tiene al principio datos o estructuras de datos particulares, sino que solo cuenta con el constructor que es capaz de generar múltiples casos particulares.

En el caso de sistemas de información esta idea se puede generalizar, buscando que, más que tener datos, procesos o estructuras particulares de un sistema, se cuente con mecanismos que nos permitan construir los datos, procesos y estructuras a partir de una descripción general del sistema. como se muestra en el esquema de la figura 4.

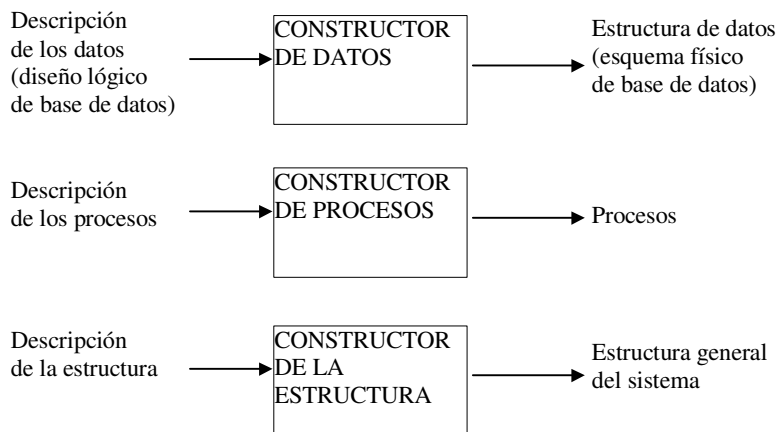


Figura 4. Constructores de datos, procesos y estructura

Estos tres módulos se pueden integrar en el esquema mostrado en la figura 5.

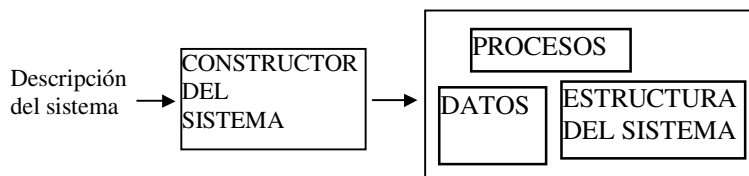


Figura 5. Integración de los constructores en un constructor del sistema

Donde el constructor del sistema es un programa capaz de construir la estructura general de un Sistema de Información a partir de la descripción del sistema, de la misma forma que un constructor de Base de Datos es capaz de construir el esquema físico de una Base de Datos a partir de un Esquema Lógico de Base de Datos.

Si se construye el sistema de información de tal forma que los componentes sean independientes en forma relativa entre sí y se da una interrelación entre el constructor y el sistema de tal forma que cualquier cambio en la descripción del sistema se refleje en tiempo real en el sistema de información, entonces se puede considerar que la imagen de la realidad que tiene el sistema de información es bastante cercana a la que se quiere reflejar.

A una herramienta o programa capaz de encontrar un conjunto de reglas a partir de ejemplos la conocemos como Mecanismo o Herramienta Inductiva; los mecanismos inductivos fueron de las primeras herramientas utilizadas en el desarrollo de constructores y son de las más generales, el esquema de un constructor basado en una herramienta inductiva es el mostrado en la figura 6.

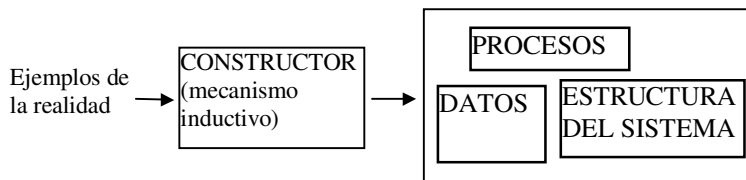


Figura 6. Herramienta inductiva

En el esquema anterior se maneja un mecanismo inductivo para construir una Imagen de la Realidad, sin embargo en esta imagen solo se tienen un conjunto de reglas en términos de procesos, estructura y datos del sistema por lo que es necesario integrar una herramienta capaz de resolver problemas siguiendo esas reglas, o sea un mecanismo de tipo deductivo como se ve en el diagrama de la figura 7.,

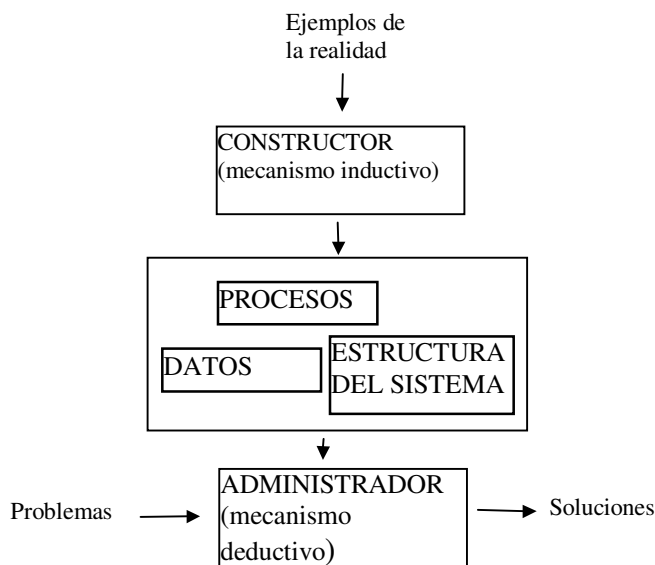


Figura 7. Integración de la componente deductiva

Para contar con un Sistema Evolutivo lo único que hace falta es dotar a esta herramienta con la capacidad de actualizar su imagen de la realidad en tiempo real con lo que se tiene la arquitectura de la figura 8.

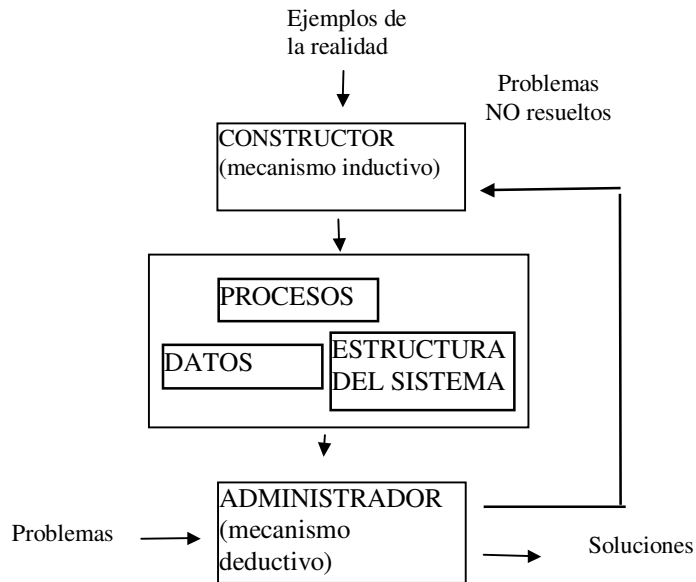


Figura 8. Retroalimentación para actualizar la imagen de la realidad

Desde hace tiempo en el área de Sistemas Evolutivos se ha buscado desarrollar herramientas automatizadas capaces de obtener en forma automática el conjunto de reglas del sistema a partir de ejemplos y descripciones generales de un programa.

De donde la Arquitectura General de un Sistema Evolutivo es la mostrada en la figura 9.

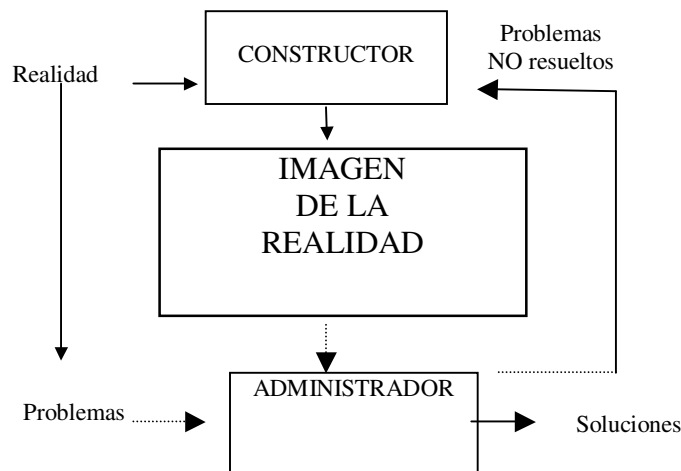


Figura 9. Arquitectura General de un Sistema Evolutivo

Calcula la Regresión de X * Y , Z
 se encuentran las Unidades Léxicas mostradas en la tabla 1.

Calcula	a, acción
la	i, ignora
Regresión	a
de	i
X	d, dato
*	a
Y	d
,	i
Z	d

Tabla 1. Unidades Léxicas

Originalmente el Manejador Léxico no tiene ningún conocimiento acerca del lenguaje a utilizar y solamente cuenta con la capacidad para encontrar los diferentes tipos de unidades léxicas de un lenguaje.

Mecanismo de Dialogo: En un Sistema Evolutivo muchas veces se desconoce hasta el tipo de unidades léxicas que conforman el lenguaje, por lo que, se ha visto que un mecanismo muy general es aquel que permite catalogar en tiempo real las nuevas unidades Léxicas. para lo cual se maneja el esquema de la figura 11.

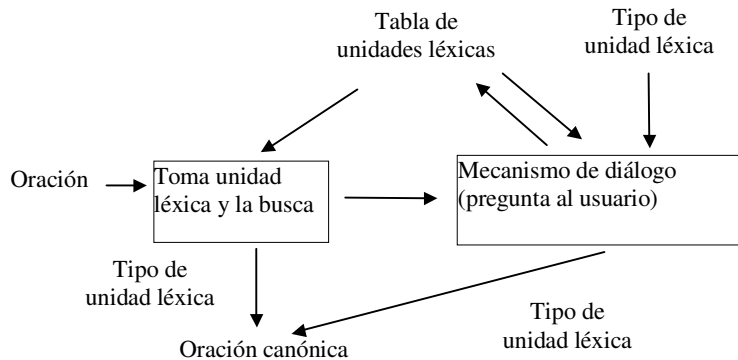


Figura 11. Mecanismo de diálogo

Este mecanismo "trivial" es extremadamente fuerte y refleja el proceso que sigue un niño cuando desconoce una palabra.

El siguiente problema se presenta porque cotidianamente manejamos una palabra como una cadena de caracteres (letras, números, signos) separados por espacios, por ejemplo en la oración “el perro ladra” tenemos tres palabras: “el”, “perro”, “ladra”, el problema es que estamos manejando elementos semánticos y los elementos semánticos no necesariamente son cadenas separadas por blanco, por ejemplo “Buenos Aires” es un elemento semántico que se refiere a una ciudad, “buenos” es otro elemento y “aires” es otro, por lo que en el traductor es conveniente tratar de traducir elementos semánticos y no solo palabras, a la unidad con significado semántico propio se le llama unidad léxica (la unidad léxica corresponde en muchos casos a una palabra, pero existen otros muchos casos en los cuales eso no es cierto).

Las dos variantes mas generales del método son:

Tomar un texto y buscar la cadena mas grande que se repita en el texto y sustituirla por una etiqueta X1, tomar la siguiente cadena que se repita y sustituirla por una cadena X2 y así sucesivamente mientras existan cadenas, con este método asumimos que cada Xi es una “unidad léxica”,

Otro método parte al revez, se buscan las palabra (cadenas separadas por blancos) y a todas las palabras iguales se les asigna la misma etiqueta, luego si se encuentra una combinación de etiquetas repetida varias veces (por ejemplo la cadena de etiquetas X8X3X14 aparece repetida varias veces) se supone que toda esa combinación también es por si sola una unidad léxica y se le asigna su propia etiqueta.

6 MANEJADOR SINTACTICO

A partir de los resultados del Manejador Léxico el Manejador Sintáctico encuentra la estructura del sistema para lo cual, toma como entrada el conjunto de oraciones canónicas y genera la Estructura.

Este mecanismo es tal vez el componente mas importante de un Sistema Evolutivo ya que es el responsable de encontrar las reglas generales o patrones de estructura del sistema y para lograrlo utiliza normalmente métodos de la inferencia gramatical.

Inferencia Gramatical y Sistemas Evolutivos

La Inferencia Gramatical es una herramienta de la lingüística Matemática utilizada originalmente en el área de reconocimiento de Patrones y que posteriormente se ha extendido y usado masivamente en la construcción de Sistemas Evolutivos.

El problema que ataca la Inferencia Gramatical consistir básicamente en encontrar la Gramática (Estructura) que describe a un lenguaje dado a partir de ejemplos de oraciones del Lenguaje, figura 12.



Figura 12. Inferencia gramatical

Las herramientas de la Inferencia Gramatical trabajan con la estructura de las oraciones buscando encontrar una estructura general (o regla sintáctica) a partir de estructuras particulares (u oración).

En sus inicios se desarrollaron un conjunto de algoritmos orientados a resolver problemas específicos y en la mayoría de los casos eran métodos difíciles de entender y mas difíciles de programar, pero conforme se empezó a atacar el problema para construir Sistemas Evolutivos se fueron encontrando nuevos métodos y generalizando el problema, por lo que. en la actualidad se ha encontrado que muchos de los métodos de inferencia gramatical se basan en las operaciones básicas de Factorización, Distribución y Recursividad por lo que a continuación se explicaran estas operaciones.

6.1 Factorización Lingüística.

6.1.1) Introducción.

La factorización lingüística consiste en la búsqueda de cadenas que se repiten varias veces en un texto y verlas como un factor dentro del texto.

Así, si por ejemplo, se tienen las siguientes oraciones:

Juan es hermano de Pedro y
Juan estudia en UPIICSA

se puede detectar que en las dos oraciones se encuentra presente la palabra Juan y que un párrafo equivalente sería:

Juan es hermano de Pedro y estudia en UPIICSA.

Si se observa lo que se ha hecho es detectar que la palabra Juan era común a las dos oraciones por lo que se factorizó (o sea que se sacó como factor común) y se obtuvo un párrafo donde sólo aparece una sólo vez.

Para que se pueda visualizar el proceso sustituiremos fragmentos de la oración por etiquetas de acuerdo a la siguiente tabla:

Fragmento	Etiqueta
Juan	o1
es hermano de	r1
Pedro	o2
y	+
estudia en	r2
UPIICSA	o3

Con lo que el párrafo Juan es hermano de Pedro y
Juan estudia en UPIICSA

quedarían como:

o1 r1 o2 +
o1 r2 o3

Donde se observa que o1 es común a las 2 oraciones.

A las oraciones

o1 r1 o1 y
o1 r2 o3

se les conoce como oraciones canónicas y en general cuando se sustituyen los elementos de una oración por una representación que permita visualizar la estructura de la oración se obtiene una oración canónica.

Recordemos que la factorización algebraica consiste en encontrar los factores comunes en una expresión algebraica y sacarlos de la expresión, como se ve a continuación:

$$. \underline{a}b + \underline{a}c = a(b+c)$$

$$.3x + 3y = 3(x + y)$$

$$a(b * c) + a(e/f) = a(b*c+e/f)$$

Aplicando lo anterior a las oraciones canónicas entonces tenemos que:

$$o_1 r_1 o_2 + o_1 r_2 o_3 = o_1 (r_1 o_2 + r_2 o_3)$$

o sea que el párrafo

$$o_1 r_1 o_2 + o_1 r_2 o_3$$

es equivalente al párrafo

$$o_1 (r_1 o_2 + r_2 o_3)$$

si sustituimos las etiquetas por los fragmentos que representan tenemos entonces que:

$$\frac{\text{Juan es hermano de Pedro y estudia en UPIICSA}}{o_1 \quad r_1 \quad o_2 + r_2 \quad o_3}$$

6.1.2) Generación de Gramáticas.

La factorización Lingüística es una herramienta muy poderosa ya que permite encontrar los factores comunes dentro de un conjunto de oraciones, por lo que, si por ejemplo tengo un conjunto de ejemplos de algún lenguaje, aplicando la factorización se pueden encontrar algunos de los factores comunes o reglas generales del lenguaje.

Lo anterior se puede aplicar para encontrar entonces una Gramática de un lenguaje a partir de ejemplos de las oraciones del lenguaje, ya que si por ejemplo, se tienen las siguientes oraciones canónicas (que se obtuvieron sustituyendo los elementos de las oraciones del lenguaje por etiquetas)

a b c d e
a b m e x y z
a b m g

lo primero que se hace es que como esas oraciones canónicas representan la estructura de las oraciones del lenguaje, entonces las integramos para formar una primera gramática del lenguaje

S --> a b c d e l
a b m e x y z l
a b m g

conocida como Gramática Canónica donde 'S -->' se puede ver como el nombre de una rutina y el símbolo 'l' como un separador entre los diferentes casos de un programa.

Por lo que, la Gramática Canónica se puede leer como:

La Rutina S genera tres posibles tipos de oraciones:

Oraciones del tipo abcde
del tipo abmexyz
y del tipo abmg

Lo cual es congruente con el hecho de que cada uno de los tipos anteriores corresponde a cada una de las oraciones canónicas que se dieron como ejemplo, por lo que podemos afirmar que la Gramática Canónica genera al menos las oraciones que se tenían como ejemplo originalmente.

Como siguiente paso se toman las oraciones de la gramática canónica y se les aplica la factorización lingüística, para lo que se considera al símbolo 'l' equivalente al '+' del Algebra tradicional

```
S --> abcde l
      abmexyzl
      abmg
```

factorizando ab queda:

```
S --> ab( cde l
      mexyz l
      mg )
```

Factorizando m queda:

```
s --> ab( cde l
      m( exyzl
      g ) )
```

Dado que no es común el uso de paréntesis dentro de una Gramática se introducen una serie de variables auxiliares (conocidas como Variables no Terminales) en lugar de los paréntesis, quedando
Introduciendo la variable no terminal X

```
S --> abX
```

```
X --> cdel
      m( exyzl
      g )
```

Introduciendo la variable no terminal Y

```
S --> abX
X --> cdel
      mY
Y --> exyzl
      g
```

Que se lee:

El programa S genera ab y llama a la rutina X

La rutina X genera las cadenas
cde o
my

La rutina Y genera las cadenas
exyz o
g

si analizamos este programa podemos ver que:

y después las opciones
c...
me...
mg...

que si observamos es precisamente la idea de la segunda gramática.

Por lo que la primera gramática se comporta como el programa:

```
Program S
1) a b c d e
2) a b m e x y z
3) a b m g
```

fin programa

y la segunda gramática se comporta como:

```
Program S
ab
  1)cde
  2)m
    1)exyz
    2)g
```

fin programa

O sea que las instrucciones repetitivas sólo se ejecutan una sólo vez.

6.1.4) Aplicaciones.

Una de las aplicaciones de la factorización lingüística se encuentra precisamente en la depuración de programas con el fin de quitar código redundante.

Sin embargo no es la única ya que una gran cantidad de métodos de inferencia gramatical se reducen a la aplicación de la factorización.

Una aplicación que se desarrolló en 1988 en colaboración con Javier Ortiz (en esa época Coordinador de la Maestría en Computación del CENIDET en Cuernavaca, Mor.) consistió en la aplicación de la factorización a la construcción de un Sistema Evolutivo generador de Sistemas Expertos, en el cual, la idea consistió básicamente en tomar una gran cantidad de oraciones en las cuales un experto explica como resuelve un problema y transformar cada oración en una oración canónica incluyendo por ejemplo síntomas (s), diagnósticos (d) y tratamientos (t) e ignorando todo lo demás.

A partir de ahí integrar todas las oraciones canónicas en una Gramática Canónica, mediante factorización agrupar los síntomas comunes y proponerlos como reglas generales hasta construir una cascada de reglas de las más generales a las más específicas que caracterizan un problema o diagnóstico particular.

Por ejemplo si se tiene la oración:

Paciente femenino de 15 años con 38 grados de temperatura y
S1 S2 S3

dolor en el pecho, se le diagnosticó faringitis y se le

$$\begin{array}{ccc} S4 & & d1 \\ \text{recetó } \underline{\text{antibióticos}}, \underline{\text{antistamínicos}} \text{ y } \underline{\text{reposo}} \\ t1 & t2 & t3 \end{array}$$

la oración canónica equivalente sería:

$$S1S2S3S4d1t1t2t3$$

Por otro lado si en lugar de tener una sola oración se tiene la información de todos los pacientes del hospital entonces se pueden obtener cientos o miles de reglas canónicas, las cuales mediante factorización pueden proporcionar las reglas generales de un problema y su tratamiento.

Por ejemplo si se tienen las reglas.

$$S \rightarrow \begin{array}{l} S1S2S3S4d1t1t2t3 \\ S1S2S5d2t1t4 \\ S1S4S6S7d3t5 \end{array}$$

Factorizando S1

$$\begin{array}{l} S \rightarrow S1X \\ X \rightarrow \begin{array}{l} S2S3S4D1T1T2T3 \\ S2S5D2T1T4 \\ S4S6S7D3T5 \end{array} \end{array}$$

Factorizando S2

$$\begin{array}{l} S \rightarrow S1X \\ X \rightarrow \begin{array}{l} S2Y \\ S4S6S7D3T5 \end{array} \\ Y \rightarrow \begin{array}{l} S3S4D1T1T2T3 \\ S5D2T1T4 \end{array} \end{array}$$

Analizando la última gramática se observa que S1 es la característica general de los pacientes y después se tiene a los pacientes con S2 o con S4.

Obtener un sistema experto a partir de las reglas es directo.

6.2 Distribución Lingüística

La distribución lingüística es la operación inversa de la factorización y para entenderla veremos un ejemplo basado en problemas de representación del conocimiento.

Existen varios mecanismos para representar el conocimiento, pero uno de los más usados son las redes semánticas, donde una red semántica está formada por hechos o reglas de la forma **objeto relación objeto (o r o)**, por ejemplo a partir de la oración: **Juan es hermano de Pedro**

se detecta que **Juan** es un *objeto*, **es hermano de** es una *relación* y **Pedro** es otro *objeto* por lo que de esta oración se obtienen directamente los componentes de una red semántica.

Por lo común los documentos no vienen divididos en párrafos de la forma **oro**, por ejemplo si observa este artículo se dará cuenta que existen pocas oraciones donde directamente aparezca un objeto seguido de una relación y de otro objeto, por lo que si se quiere construir la red semántica a partir de las oraciones presentes en un documento se requiere de un análisis mas profundo, esto fue lo que realizo Jesús Olivares en su tesis de licenciatura "Sistemas Evolutivos para Representación de Conocimiento" desarrollada en la UPIICSA del IPN, donde planteo un mecanismo evolutivo para construir un base de conocimientos basada en una Red Semántica Aumentada (Que almacena palabras, imágenes, sonidos, etc.).

La idea de Jesús consistió básicamente en aplicar una serie de técnicas de Inferencia Gramatical basadas en la distribución lingüística, para transformar una oración compuesta de múltiples elementos en un conjunto de oraciones semánticamente equivalentes a la original, donde cada oración final es de la forma **oro**, por lo que, son fácilmente representadas con una red semántica.

Por ejemplo en la oración:

Juan estudia en el poli, trabaja en el INEGI y vive en Lindavista.

se tienen varios hechos pero sin embargo no son directamente representables mediante una red semántica, por lo que se le aplicara la distribución lingüística, por facilidad representaremos los elementos de la oración mediante letras y números, de tal manera que se note directamente cuales son los objetos, cuales las relaciones y cuales los separadores:

Juan	es un objeto	o0
estudia en el	es una relación	r1
poli	es un objeto	o1
,	es un separador	+
trabaja en el	es una relación	r2
INEGI	es un objeto	o2
y	es un separador	+
vive en	es una relación	r3
Lindavista	es un objeto	o3

de donde la oración queda representada como:

$$\mathbf{o0} (r1o1 + r2o2 + r3o3)$$

El paréntesis que esta después de **o0** refleja que tiene relación con todos los demás elementos de la oración.

Si se toma la oración y se le aplica la operación de distribución del álgebra queda:

$$\mathbf{o0} (r1o1 + r2o2 + r3o3) = \mathbf{o0r1o1} + \mathbf{o0r2o2} + \mathbf{o0r3o3}$$

y si sustituimos el significado de los diferentes elementos tenemos que:

$$\mathbf{o0r1o1} \quad \text{Juan estudia en el poli}$$
$$+ \text{ ,}$$

o0r2o2 Juan trabaja en el INEGI
+ y
o0r3o3 Juan vive en Lindavista.

o sea que a partir de una oración compleja se obtuvieron tres oraciones simples del tipo **oro** que mantienen el significado de la oración original y que son directamente representables en una red semántica.

6.3 Recursividad Lingüística

El proceso de recursividad lingüística también busca un conjunto de elementos comunes dentro de las oraciones pero con la diferencia de que se buscan cadenas de elementos que se repiten periódicamente y en forma consecutiva mas de cierto número mínimo de veces (normalmente tres o más veces) y se asume que esa cadena se puede repetir tantas veces como se quiera.

Por ejemplo en la cadena

$$S \rightarrow a \underline{b c} \underline{b c} \underline{b c} \underline{b c} f$$

Los elementos **b c** se repiten consecutivamente 4 veces por lo que se asume que se pueden repetir tantas veces como se desee y esto se expresa introduciendo un elemento auxiliar y haciéndolo recursivo.

$$S \rightarrow a X f$$
$$X \rightarrow \underline{b c} \underline{b c} \underline{b c} \underline{b c}$$
$$\Downarrow$$
$$S \rightarrow a X f$$
$$X \rightarrow b c X$$

Por ejemplo si se tiene

$$S \rightarrow a i a d a d a d a d f$$

Introduciendo recursividad sobre **a d** queda:

$$S \rightarrow a i X f$$
$$X \rightarrow a d a d a d a d$$
$$\Downarrow$$
$$S \rightarrow a i X f$$
$$X \rightarrow a d X$$

El proceso de introducir recursividad es un mecanismo extremadamente poderoso ya que permite generalizar una secuencia de repetición y por el otro lado es un mecanismo peligroso ya que se puede generalizar mas de lo debido.

7 MATRICES EVOLUTIVAS

7.1 Antecedente: Una Representación Matricial para Redes Neuronales.

A mediados de los 70's en un seminario de Inteligencia Artificial organizado en el Centro Nacional de Cálculo (CENAC) del IPN de la Cd. de México en colaboración con la Escuela Superior de Física y Matemáticas (ESFM) del mismo instituto, trabajando junto con Gustavo Nuñez Esquer llegamos a una representación de red neuronal

Donde, a partir del modelo de neurona desarrollado por Mc Cullot y Pitts, generamos nuestra propia propuesta, basada en una representación matricial de la red neuronal, en la cual cada una de las señales de entrada a las neuronas se representa como una columna de la matriz y cada una de las neuronas equivale a un renglón, de tal forma que, en la intersección de cada renglón y columna se almacena el valor que toma la dendrita en caso de que reciba una señal de entrada.

El anterior fue un modelo simple de red neuronal representada mediante una matriz y actualmente es común encontrar representaciones matriciales de redes neuronales, por lo que el modelo desarrollado en 1976 se podría ver como antecedente y como un resultado independiente.

Sin embargo desde el principio se presentó el problema de obtener los valores de las neuronas y es ahí donde a mediados de los 80's, Cuitláhuac Cantú, al estar trabajando sobre reconocimiento de imágenes llegó en forma independiente a una representación matricial prácticamente equivalente a la encontrada para las redes neuronales, pero con capacidades evolutivas

En esta idea se parte de que originalmente la matriz está vacía y lo que hace el sistema es llegar y buscar una imagen, como no encuentra nada la coloca en el primer renglón, mas adelante cuando llega la segunda imagen, si son similares la reconoce y la acumula con la primera si no son similares entonces la coloca en el siguiente renglón y así sucesivamente.

Para lo cual cada imagen se representa como un vector, de tal forma que por ejemplo, si se tiene un gato, un perro y un ratón, cada uno de ellos se almacena como un vector y entre los tres forman una matriz como la siguiente

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	
1	0	1	0	0	1	0	1	0	gato
0	1	0	1	0	1	1	0	1	perro
1	1	0	0	1	0	0	1	1	ratón

Si se perciben varios gatos, en lugar de almacenar cada gato en un vector independiente se suman los vectores que representan cada uno de los gatos y el resultado se toma como la representación de la estructura general del gato y se almacena en la matriz.

(5 1 4 0 0 5 0 4 1 20) gato

Por otro lado si se tienen varios gatos, perros y ratones, la matriz seria de la forma:

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	h	
5	1	4	0	0	5	0	4	1	20	gato
0	3	0	3	0	2	3	2	1	14	perro
1	1	0	0	1	0	0	1	1	5	ratón

Donde cada renglón representa un tipo de objeto y h es un valor donde se acumula el número de puntos en el vector (En este caso se puso h como la suma de los valores del vector por facilidad del ejemplo, sin embargo existen otros métodos para asignar este valor).

Lo anterior es equivalente a que cada gato se dibujara en un acetato y posteriormente se superpusieran los acetatos, con lo que las características repetitivas del gato quedan más recalcadas y equivale a números mayores en el vector que lo representa.

Si se desea reconocer un nuevo objeto, se representa también como un vector, se multiplica por la matriz, se ve en que renglón se obtuvo el máximo valor y se le asocia a ese renglón el objeto. Por ejemplo si tomamos la matriz anterior y llega el objeto 1 0 1 0 0 1 0 1 0, se multiplica por la matriz y les restamos el valor de h quedando: -2 para el gato, -10 para el perro y -3 para el ratón. De donde se propone que el objeto reconocido es un gato.

Como siguiente punto el sistema acumula el nuevo vector en la matriz, con lo que, en el ejemplo anterior el nuevo vector se suma al renglón del gato y la matriz queda:

a ₁	a ₂	a ₃	a ₄	a ₅	a ₆	a ₇	a ₈	a ₉	h	
6	1	5	0	0	6	0	5	1	24	gato
0	3	0	3	0	2	3	2	1	14	perro
1	1	0	0	1	0	0	1	1	5	ratón

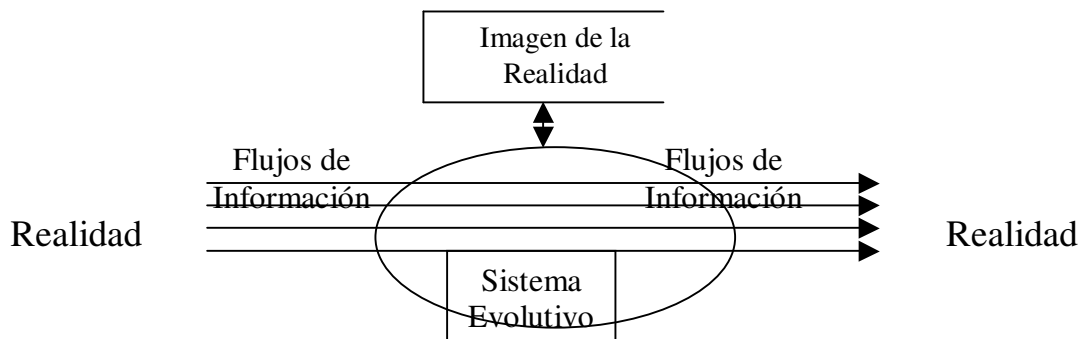
Con lo que, el sistema evolutivo esta transformando permanentemente su imagen de la realidad.

Como se puede observar este método reúne la característica de que esta evolucionando en forma natural y encontrando la imagen acumulada (y por tal, la imagen promedio), con lo que no existe un proceso previo de aprendizaje y otro de aplicación, sino que por el proceso natural de conocer y reconocer las imágenes va evolucionando.

Es importante observar que, la matriz obtenida es prácticamente igual a la de la red neuronal de 1976, con lo que se plantea como un mecanismo para la representación de sistemas evolutivos y en particular de redes neuronales evolutivas, ya que, los valores de la matriz están cambiando en tiempo real y esto es equivalente a modificar las interrelaciones entre las neuronas (ya que en el modelo original cada entrada de la matriz representa una conexión entre neuronas y estas conexiones se están modificando en tiempo real).

Este mecanismo de matriz que se esta transformando permanentemente se conoce como *matriz evolutiva* y tiene la ventaja de que siempre se está actualizando para reflejar una imagen de la realidad.

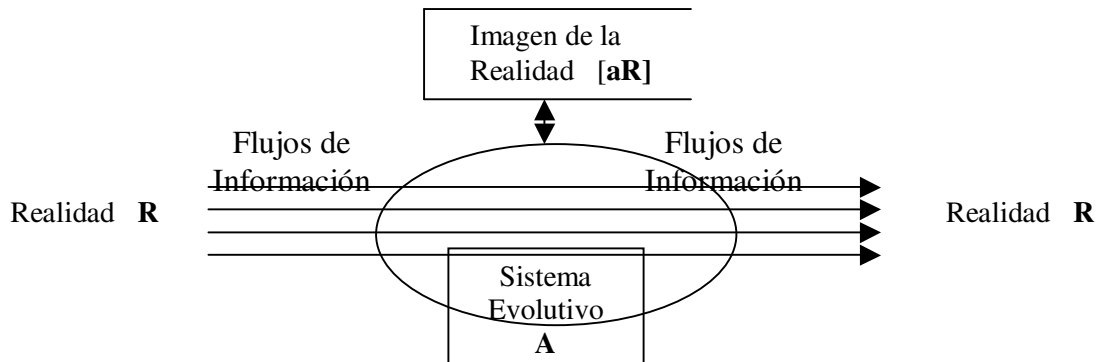
Con las matrices evolutivas se observa que el sistema esta permanentemente evolucionando (o sea que no existe una etapa de "aprendizaje" y otra de aplicación) y que *por el puro hecho de que fluye la información el sistema se modifica*.



Por los flujos de materia, energía e información el Sistema evoluciona

Evolución de la imagen de la realidad

Por el puro hecho de que fluye la información el sistema se modifica.



Por los flujos de materia, energía e información el Sistema evoluciona

R es la realidad, **A** es el sistema evolutivo

A percibe la realidad **R**, **aR**

A construye su imagen de la realidad **R**, **[aR]**

Originalmente la imagen de la realidad **[aR]** esta vacía

Cuando empieza a fluir la información, el sistema empieza a percibir la realidad **aR**
y a construir la imagen de la realidad **[aR]**

$$\mathbf{aR} \rightarrow [\mathbf{aR}]$$

El sistema permanentemente esta percibiendo la realidad **aR**
y su imagen de la realidad **[aR]** permanentemente esta evolucionando

$$\mathbf{aR} \oplus [\mathbf{aR}] \rightarrow [\mathbf{aR}]$$

7.2 MATRICES EVOLUTIVAS Y SISTEMAS EXPERTOS.

Las matrices evolutivas constituyen por si solas una herramienta con múltiples aplicaciones.

Por ejemplo un campo donde se pueden aplicar las matrices evolutivas es en el área de los sistemas expertos, por ejemplo, si partimos de un sistema experto de diagnostico[14] formado por reglas de la forma

$$S_1 \ S_2 \ S_3 \ \dots \ S_n \Rightarrow D_i : T_i$$

donde S_j son los síntomas, D_i los diagnósticos y T_i los tratamientos (o sea que un conjunto de síntomas generan un diagnostico y un tratamiento).

El sistema experto puede representarse como una matriz, donde cada columna corresponde a un síntoma y cada renglón a una regla del experto.

$$\begin{array}{c}
 \\
 r_1 \\
 r_2 \\
 \dots \\
 r_m
 \end{array}
 \begin{array}{cccccc}
 S_1 & S_2 & S_3 & \dots & S_n \\
 \left| \begin{array}{cccccc}
 1 & 0 & 0 & \dots & 0 \\
 0 & 1 & 1 & \dots & 0 \\
 \dots & \dots & \dots & \dots & \dots \\
 1 & 0 & 1 & \dots & 1
 \end{array} \right.
 \end{array}$$

De tal manera que si aparece un 1 en el renglón i y la columna j , quiere decir que la regla i necesita el síntoma j para cumplirse.

Aquí es importante comentar que, una de las características que permite la representación de las reglas de inferencia mediante una matriz es que no importa el orden en el que se presenten los síntomas, o sea que, la cadena lingüística es conmutativa bajo la concatenación lo cual no ocurre por ejemplo con las oraciones del Español las cuales no se pueden representar en general mediante la matriz, porque no se permite la conmutabilidad.

Ahora bien los valores de la matriz pueden ser cualesquiera, por lo que a cada síntoma se le puede dar un valor diferente dependiendo del grado de importancia, o de la probabilidad o de la frecuencia de ese síntoma en particular, con lo que se tiene una matriz borrosa de la forma:

$$\begin{array}{c}
 \\
 r_1 \\
 r_2 \\
 \dots \\
 r_m
 \end{array}
 \begin{array}{cccccc}
 S_1 & S_2 & S_3 & \dots & S_n \\
 \left| \begin{array}{cccccc}
 a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\
 a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\
 \dots & \dots & \dots & \dots & \dots \\
 a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn}
 \end{array} \right.
 \end{array}$$

donde a_{ij} es el peso que tiene el síntoma j en la regla i . Entonces si llega un problema que tiene por ejemplo los síntomas 1, 3, 7, ..., n , con pesos 0.5, 1.3, 0.7, ..., 2.5, solo se requiere representar el problema como el vector: (0.5 0 1.3 0 0 0 0.7 2.5) y operar el vector con la matriz.

La forma más fácil de operar el vector con la matriz consiste en tomar el vector y multiplicarlo por la matriz con lo que se tiene un vector resultante y solo se requiere buscar el elemento i máximo para detectar a que regla corresponde el conjunto de síntomas.

Por ejemplo, si se tiene la siguiente matriz:

$$\begin{array}{c}
 \\
 r_1 \\
 r_2 \\
 r_3
 \end{array}
 \begin{array}{cccccc}
 S_1 & S_2 & S_3 & S_4 & S_5 & S_6 \\
 \left| \begin{array}{cccccc}
 3.2 & 0 & 4.0 & 6.5 & 0 & 0 \\
 2.0 & 1.3 & 0 & 0 & 1.5 & 0 \\
 0 & 3.0 & 0 & 0 & 4.2 & 2.0
 \end{array} \right.
 \end{array}$$

y llega un problema con los síntomas (1.0 0 1.5 0 0 0.7), entonces se multiplica la matriz por el vector quedando el resultado:

$$\begin{array}{l|l} r_1 & 9.2 \\ r_2 & 2.0 \\ r_3 & 1.4 \end{array}$$

de donde se propone la r_1 como el resultado del sistema.

Una de las ventajas de representar al sistema experto mediante una matriz es que ahora se puede ver y atacar con múltiples herramientas incluyendo entre otras la lógica difusa, redes neurales, teoría de la medida y sistemas evolutivos. En particular podemos aplicar las técnicas de matrices evolutivas y partir de que la matriz está originalmente vacía y llena de ceros, cuando llegan los primeros síntomas y no encuentra nada almacena el vector en el primer renglón y asigna el diagnostico (le puede pedir el diagnostico al experto humano).

Por ejemplo, si llegan los síntomas (1.2 0 0 2.3 0 0.7) el sistema busca en la matriz el diagnostico, pero como esta vacía no los encuentra, entonces pregunta por el diagnostico y genera una matriz con un renglón.

$$\begin{array}{l|cccccc|l} & S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & \\ r_1 & 1.2 & 0 & 0 & 2.3 & 0 & 0.7 & D_1 \end{array}$$

Si llegan ahora los síntomas (0.7 0 0 0 3.2 0), como la colisión con el vector almacenado es baja, pregunta por el diagnostico, si el nuevo diagnostico es diferente crea un nuevo renglón, quedando la matriz:

$$\begin{array}{l|cccccc|l} & S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & \\ r_1 & 1.2 & 0 & 0 & 2.3 & 0 & 0.7 & D_1 \\ r_2 & 0.7 & 0 & 0 & 0 & 3.2 & 0 & D_2 \end{array}$$

Si llega una cadena de síntomas con alta colisión, como por ejemplo:

$$(1.0 \ 0 \ 0 \ 0.8 \ 0 \ 1.2)$$

el sistema emite el diagnostico D_1 y acumula el nuevo síntoma sobre la matriz anterior en el renglón 1:

$$\begin{array}{l|cccccc|l|l} & S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & & \# \text{ de} \\ & & & & & & & \text{acumulados} & \\ r_1 & 2.2 & 0 & 0 & 3.1 & 0 & 1.9 & D_1 & 2 \\ r_2 & 0.7 & 0 & 0 & 0 & 3.2 & 0 & D_2 & 1 \end{array}$$

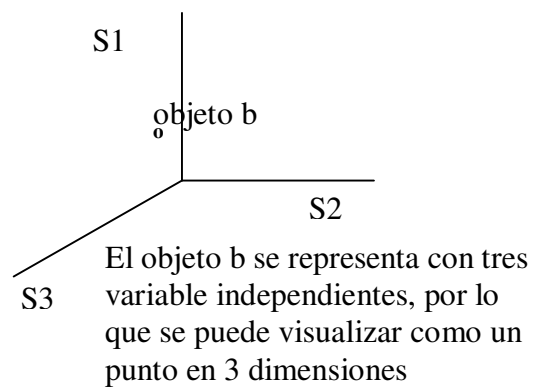
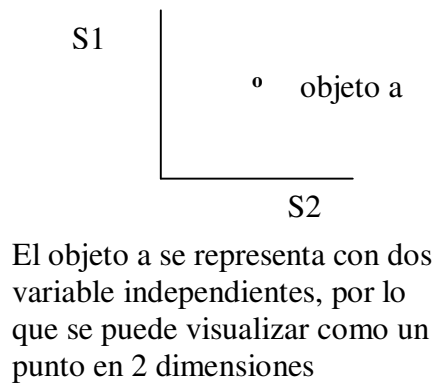
Al tener el vector acumulado se están reforzando los síntomas y en forma natural aumentan los pesos.

El sistema siempre suma. Si llega una cadena de síntomas con alta colisión los suma al renglón diagnosticado. Si llega una nueva cadena crea un nuevo renglón. Si llega una cadena y el experto indica que corresponde a un diagnostico presente, la nueva cadena se acumula a este diagnostico.

Si se toma el vector acumulado y cada síntoma lo dividimos entre el número de acumulados tenemos una regla promedio, por lo que, lo que estamos encontrando es la regla promedio asociada a un diagnostico y mediante el mecanismo evolutivo esta regla se está depurando y afinando cotidianamente, o sea que, en forma natural y permanente se están encontrando y afinando las reglas del sistema.

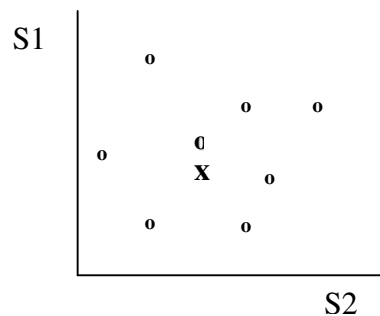
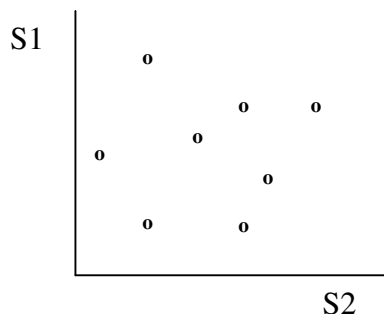
7.3 Una Representación N-Dimensional.

Si analizamos un poco mas a las matrices evolutivas vemos que cada imagen, regla o cadena de síntomas promedio se representa mediante un conjunto de n síntomas o variables, si estas variables son independientes entre si entonces cada objeto (imagen, regla o cadena de síntomas promedio) representado por las n variables se puede visualizar como un punto en un espacio de n dimensiones.



Si un objeto c se representa con cuatro variable independientes, entonces se puede visualizar como un punto en 4 dimensiones, y en general un objeto n que se representa con n variable independientes se puede visualizar como un punto en n dimensiones

O sea que cada vector de la matriz evolutiva representa un punto en un espacio multidimensional y todas las reglas forman una nube de puntos. De donde, por ejemplo, el proceso de encontrar el diagnostico asociado a un vector de síntomas equivale a encontrar el punto que tiene distancia mínima con este vector. (Por facilidad visual mostraremos imágenes en 2 dimensiones , pero la idea se aplica a n dimensiones.)



Ahora bien, conforme aumenta la cantidad de variables que representan a los objetos la posibilidad de que colisionen (asocien el mismo punto a objetos diferentes) disminuye.

Para mostrar lo anterior supondremos que se tienen varios objetos y cada objeto se representa con un valor de una variable o como un punto en una dimensión, en su momento varios objetos podrían tomar el mismo valor y caer en el mismo lugar.

Por ejemplo, si se quiere representar a los alumnos de un grupo por sus calificaciones y éstas toman puros valores enteros y tienen una distribución uniforme entre 0 y 10 entonces la calificación puede tomar 11 posibles valores.

Para el ejemplo de las personas la calificación no es una buena variable ya que en general en un grupo se tienen mas de 10 alumnos y además la distribución no es uniforme. Otra variable mejor podría ser la altura en centímetros (nuevamente supondremos el caso ideal de distribución uniforme y que la probabilidad de tomar cualquier valor es la misma)

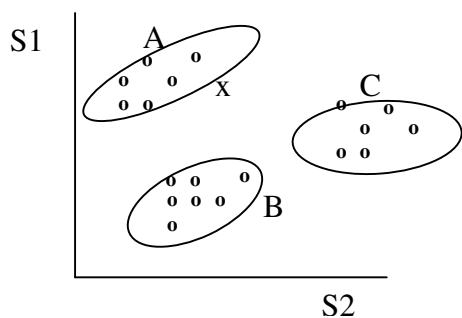
Ahora bien, si se tienen dos variables no correlacionadas y se utilizan combinadas el espacio se multiplica, o sea que si se categoriza a un conjunto de objetos mediante 2 variables independientes y si #1 es el número de puntos en x , y #2 es el número de puntos en y entonces $\#1 * \#2$ es el número de puntos en el espacio de valores (conocido también como espacio de caos).

Por ejemplo, en el caso de los alumnos, si $\#calificación = 11$ y $\#tamaño = 20$ entonces $\#espacio = \#calificación * \#tamaño = 11 * 20 = 220$. Con lo que el espacio crece y las colisiones se reducen.

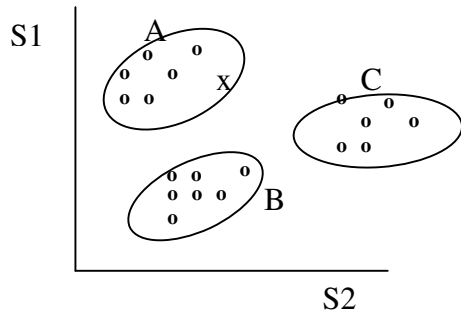
Si se tienen 3, 4 o mas variables rápidamente se llega a espacios con una gran cantidad de posibles valores, por lo que si se tienen objetos caracterizados por 1000 variables la posibilidad de que 2 objetos diferentes tomen el mismo valor es muy pequeña.

7.4 Espacios N-Dimensionales Evolutivos

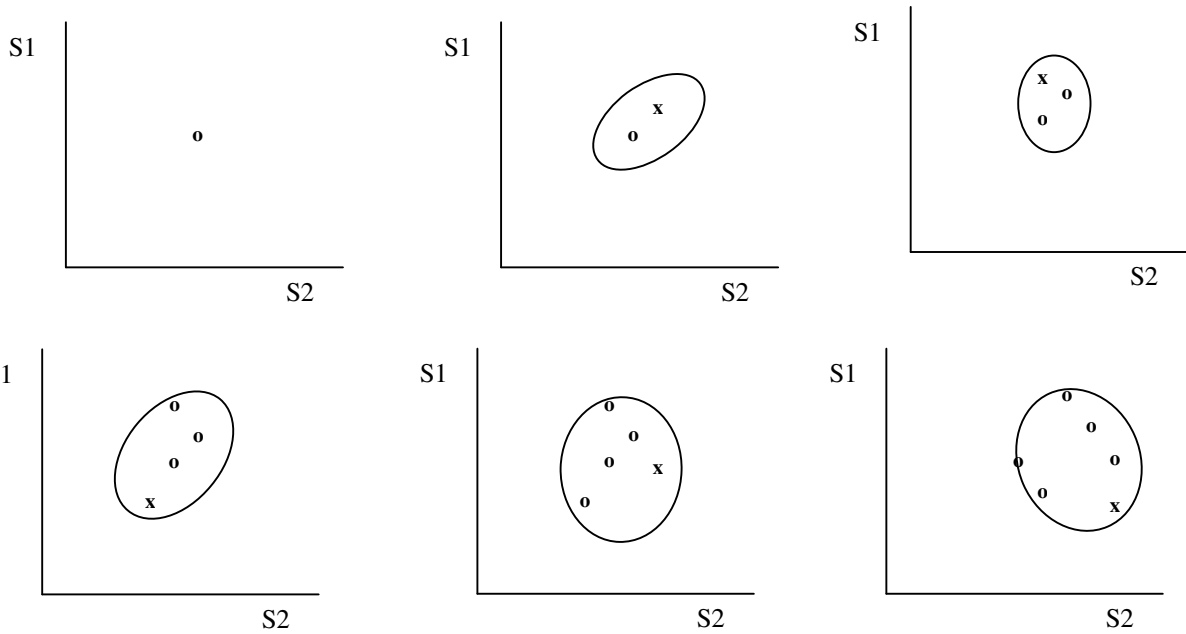
Conforme aumenta el número de variables o síntomas la posibilidad de que los objetos colisionen disminuye. Lo anterior es la base de una técnica de reconocimiento de formas conocida como análisis de cúmulos, que básicamente toma una muestra significativa de objetos similares, los representa como puntos en un espacio n-dimensional, encuentra la media y la varianza de estos puntos y cuando llega un objeto nuevo al sistema, mide la distancia entre el punto nuevo y el punto medio, si esta es pequeña lo reconoce como un objeto similar y si no lo desecha.



Sin embargo es común que cuando se aplica esta técnica, los patrones ya están prefijados, ya sea porque explícitamente fueron asignados o porque se obtuvieron mediante un proceso de aprendizaje y quedaron fijos e inmutables. Ahora bien, en el caso de las matrices evolutivas también cada imagen, regla o cadena de síntomas promedio se representa como un punto en un espacio de n dimensiones, o sea que, todas las reglas forman una nube de puntos, pero con la característica de que *la matriz evolutiva permanentemente se esta modificando*.



Originalmente la matriz evolutiva esta vacía y representa un espacio que también lo esta, más adelante cuando llegan las primeras reglas o imágenes surgen los primeros puntos, pero estos no están fijos, ya que, cuando una regla de la matriz evolutiva se modifica el punto que la representa también cambia de posición, o sea que, en forma natural y permanente el espacio n-dimensional se esta afinando y evolucionando.



Con lo que el sistema evolutivo está transformando permanentemente su imagen de la realidad y no existe un proceso previo de aprendizaje y otro de aplicación, sino que por el proceso natural de fluir la información, el sistema evoluciona.

8 EVOLUCIÓN

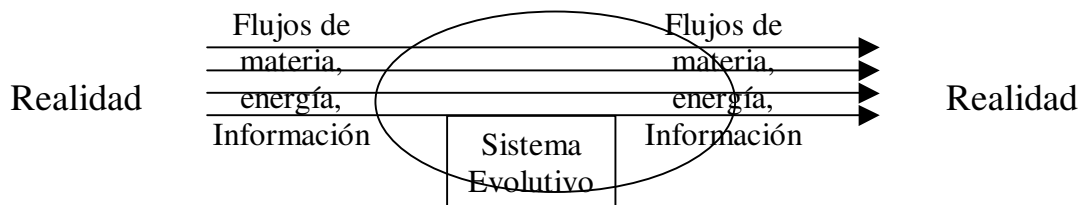
Generalizando lo anterior a sistemas en los que fluye materia, energía e información llegamos a una serie de ideas generadas durante mas de veinte años y en las que en esencia se plantea que *la evolución, el crecimiento, la vida, el aprendizaje, el pensamiento, la transformación de nuestra imagen de la realidad, los procesos de descomposición, el desarrollo y transformación de las empresas, sociedades, organizaciones, países, galaxias y universos, etc., son manifestaciones de un mismo proceso general de transformación o cambio*, y que existen reglas y propiedades generales que se aplican a las diferentes manifestaciones particulares.

Por facilidad *al concepto general lo denominaremos Evolución*, aunque lo podríamos llamar de muchas otras formas, como cambio o transformación. O sea que, cuando nos refiramos a la evolución no nos estaremos refiriendo al concepto particular que tiene asociado, sino al concepto general con el cual integra y representa a todas las manifestaciones particulares.

Todos los organismos se mantienen en un *proceso permanente de evolución*, lo que pasa es que no lo notamos porque *en algunos casos es muy lento* (por ejemplo, los cambios naturales en las piedras y las estrellas tardan mucho en notarse) y *en otros muy rápidos* (una corriente de agua, una nube, etc.) y aun cuando la evolución se realice en tiempo real o a una *velocidad observable* (como la evolución de una empresa o idea, o del conocimiento que tenemos sobre un tema) *no somos conscientes de este proceso porque nadie nos lo ha hecho notar*.

La Evolución es un fenómeno universal e intrínseco a la naturaleza y se presenta como resultado de la interacción de un organismo con su medio, con otros organismos, consigo mismo o cuando diferentes organismos y sociedades interactúan mutuamente en procesos de coevolución, dando como resultado que la organización o sistema (natural, artificial, social, etc.) se transforme y en su caso modifique el ambiente.

Por el hecho de que fluyen la materia, energía, información o algunos otros factores esenciales se da la evolución. Por ejemplo, por el hecho de que fluye a nuestro alrededor la información, nuestra imagen de la realidad cambia. Un sistema evolutivo se puede visualizar como un paisaje con cordilleras, planicies y valles, cuando llueve el agua fluye por ese espacio y por el puro hecho de fluir hace que el espacio se modifique.



Por los flujos de materia. energía. información el Sistema evoluciona

Evolución 1995

Ahora bien, *el cambio se da como resultado de la interacción fractal que se da entre los componentes de un sistema, entre los sistemas, y entre todo esto y el enorme caldo de materia, energía e información donde se encuentra inmerso, ya que al haber interacción fluye algo entre los que interactúan y esto propicia la evolución.*

CONCLUSIÓN

En este trabajo se planteo que *múltiples manifestaciones de la realidad como la vida, la evolución, el aprendizaje, la transformación del universo y muchas otras, son casos particulares de una manifestación general, la cual esta regida por la interacción fractal de múltiples sistemas en un caldo en el que fluye*

permanentemente al menos la materia, energía e información, propiciando con este flujo los procesos de transformación.

Con este trabajo espero dar una idea de la magnitud del área y propiciar que se puede empezar a plantear el surgimiento de una *Ciencia o Disciplina de la Evolución, Interacción o Cambio*, donde se pueda estudiar la *Evolución en sus diferentes manifestaciones y encontrar sus características (fundamentos, reglas, leyes, patrones, etc.) generales e independientes de las manifestaciones particulares en que se presenten.*

FUENTES DE INFORMACIÓN.

- [] Berruecos Rodríguez, Elsa, *Sistema Evolutivo Generador de Esquemas Lógicos de Bases de Datos*, UPIICSA-IPN México, 1990.
- [] Chomsky, Noam, *Estructuras Sintácticas*, Ed. Siglo XXI
- [] Bach, Emmon, *Teoría Sintáctica*, Ed. Anagrama.
- [] Salomaa, *Formal Languages*. Ed. Academic Press.
- [] C. Gonzalez, Rafael y C. Thomason, Michael, *Syntactic Pattern Recognition*, Ed. Addison-Wesley.
- [] Morales Rubio, Ángel Cesar, *Sistema Evolutivo para Tratamiento de Imágenes*, trabajo de titulación, IPN-UPIICSA México, 1992.
- [] Morales Rubio, Ángel Cesar, *Algoritmo Evolutivo Para Tratamiento de Imágenes*, IPN-UPIICSA México
- [] Camacho Villanueva, Sandra, Gómez Rendón, Guadalupe Patricia, Olivares Ceja, Jesús Manuel, *SI-VE: Sistema de Visión Experto*, IPN-UPIICSA, México, 1986
- [] Olicón Nava, Carlos, *Sistema Evolutivo Manejador de Imágenes en 2D con movimiento*, trabajo de titulación, IPN-UPIICSA México 1992
- [] Tisher, *Pc Interno*, Ed. Marcombo, 1992
- [] Jiménez Aviña, J. Antonio, Galindo Soria, Fernando, *Diseño y Construcción de Sistemas Interactivos, Aplicando Experiencias Basadas en el Tratamiento de Imágenes*, en Memorias de la 2da Conferencia de Ingeniería Eléctrica CIE/96, CINVESTAV-IPN, 1996
- [] Minsky, Marvin y Papert, Seymour, *Perceptrons, An Introduction to Computational Geometry*
- [] Minsky, Marvin, *Finite and Infinite Machines*
- [] Kleene, S. C., *Representation of Events in Nerve Nets and Finite Automata*, Automata Studies
- [] Cantú Rohlík, Cuitláhuac, comunicaciones personales y cursos de Representación del Conocimiento, Inteligencia Artificial y Redes Neuronales
- [] Torres Hernández, Luis E, Longoria, Luis C., Rojas Salinas, Antonio, *Aplicación de los Sistemas Evolutivos en el Análisis de Espectros de Rayos Gamma*, en Memorias del Cie/95, Primera Conferencia de Ingeniería Eléctrica CIE/95, CINVESTAV-IPN, Septiembre 11-13 de 1995 México, D.F.
- [] De La Cruz Sánchez, Eduardo, Longoria Gándara, Luis C., Carrillo Mendoza, Rodolfo A., *Sistema Evolutivo para el Diagnóstico de Fallas en Maquinas Rotatorias*, en Memorias del Cie/95, Primera Conferencia de Ingeniería Eléctrica CIE/95, CINVESTAV-IPN, Septiembre 11-13 de 1995 México, D.F.
- [] Arzola Carvajal, Irene, Cruz Reyes, José Rafael, *Sistema Evolutivo para el Reconocimiento de Texto Taquigrafico*, en Memorias del Cie/95, Primera Conferencia de Ingeniería Eléctrica CIE/95, CINVESTAV-IPN, Septiembre 11-13 de 1995 México, D.F.
- [] García García, Diana Karla, Salcido Bustamante, Sergio, Ventura Silva, Alfonso, *Sistema Evolutivo de Reconocimiento de Formas en dos Dimensiones*, en Concurso Nacional de Ciencia y Tecnología, CONADE, México, 1996
- [] Olivares Ceja, Jesús Manuel, *Sistemas Evolutivos para Representación del Conocimiento*, UPIICSA-IPN.

- [] Olivares Ceja, Jesús Manuel, *Sistema Evolutivo Reconocedor de Textos*, IPN-CIC, México, febrero, 1997 en Teoría y Práctica de los Sistemas Evolutivos Versión Beta, México 1997
- [] Galindo Soria, Fernando, *Algunas Propiedades Matemáticas de los Sistemas Lingüísticos*, Memorias del Simposium Nacional de Computación. México, Nov de 1992.
- [] Galindo Soria, Fernando, *Sistemas Evolutivos*, en Boletín de Política Informática, INEGI-SPP, México. Septiembre de 1986
- [] Galindo Soria, Fernando, *Sistemas Evolutivos de Reescritura*, en Memoria del curso tutorial de Sistemas Evolutivos, 1er Congreso Internacional de Investigación en Ciencias Computacionales, Metepec México, 1994
- [] Galindo Soria, Fernando, *Sistemas Evolutivos de Lenguajes de Trayectoria*, En VI Reunión de Inteligencia Artificial, Memorias, Ed. Limusa, Junio 1989, Querétaro, Qro.
- [] Galindo Soria, Fernando, *Una Representación Matricial para Sistemas Evolutivos*, Conferencia Magistral Simposium Internacional de Computación, IPN-CENAC, México, 1993
- [] Galindo Soria, Fernando, *Sistemas Evolutivos: Nuevo Paradigma de la Informática*, Memorias de la XVII Conferencia Latinoamericana de Informática, Caracas, Venezuela, Julio, 1991
- [] Galindo Soria, Fernando, *Arquitectura Lingüístico-Interactiva para Sistemas Evolutivos*, en La Inteligencia Artificial en México, Ed. Universidad Tecnológica de la Mixteca (UTM) 1993
- [] Galindo Soria, Fernando, y Ortíz Hernández, Javier, *Sistema Evolutivo Constructor de Sistemas Expertos*, CENIDET-DGIT, UPIICSA-IPN, Noviembre, 1988, en Teoría y Práctica de los Sistemas Evolutivos, Versión Beta, México 1997

2 Panorama general de Los Sistemas evolutivos

ENTREVISTA CON EL LICENCIADO FERNANDO GALINDO SORIA

ESCOM día 8 de Octubre de 1999 tuvimos una entrevista con el Lic. Fernando Galindo Soria quien ha contribuido directamente en la investigación sobre Los sistemas evolutivos, desde su nacimiento hasta el día de hoy. He aquí un resumen del mismo.

INSTITUTO POLITÉCNICO NACIONAL

"Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas"

Curso de "Ingeniería del Conocimiento"

Este curso fué realizado por la Lic. Claudia Marina Vicario Solórzano

3 DE LOS SISTEMAS FORMALES A LOS SISTEMAS EVOLUTIVOS PASANDO POR EL TEOREMA DE GÖDEL 29 abril 96