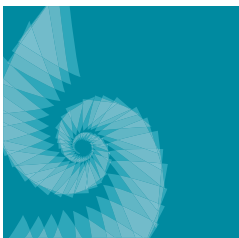




Teoría y Práctica de los
SISTEMAS EVOLUTIVOS





Primera edición: 1998

Teoría y Práctica de los Sistemas Evolutivos

Editor

Jesús M. Olivares Ceja

Segunda edición 2009

SmartDsign

Editor

Daniel Doctor Soriano

Diseño Gráfico y arte

Heidi Mariana Campos Morales

Se permite la libre copia de esta obra, en digitales, citando a los autores originales.

Índice

Prefacio: Evolución.....	11
Introducción.....	15
CAPÍTULO I RUMBO A LOS SISTEMAS CONSCIENTES	
I.1 Introducción General a los Sistemas Evolutivos	
Jesús Manuel Olivares Ceja.....	19
I.2 Desarrollo Histórico de los Sistemas Evolutivos	
Fernando Galindo Soria	21
I.3 Aplicación de un Reconocedor de Lenguaje Natural Restringido a la Recuperación de Datos	
Gabriel Cordero Sánchez	27
I.4 Introducción a la Inteligencia Artificial	
Roberto Mendoza Padilla	39
I.5 Nuevo Método para el Desarrollo de Sistemas de Información	
Roberto Mendoza Padilla	43
I.6 La Comunicación Hombre-Máquina utilizando Lenguaje Natural Restringido	
Juan Martín González Vázquez	51
I.7 SI-VE: Sistema de Visión Experto	
María Sandra Camacho Villanueva, Guadalupe Patricia Gómez Rendón, Jesús Manuel Olivares Ceja	59
CAPÍTULO II SISTEMAS EVOLUTIVOS	
II.1 Sistemas Evolutivos	
Fernando Galindo Soria.....	69
II.2 Sistemas Evolutivos: Nuevo Paradigma de la Informática	
Fernando Galindo Soria.....	77
II.3 Arquitectura Lingüístico-Interactiva para Sistemas Evolutivos	
Fernando Galindo Soria.....	93

CAPÍTULO III APLICACIÓN DE LA LINGÜÍSTICA MATEMÁTICA A LOS SISTEMAS EVOLUTIVOS

III.1 Enfoque Lingüístico	
Fernando Galindo Soria.....	101
III.2 Programación Dirigida por Sintaxis	
Fernando Galindo Soria.....	109
III.3 Sistemas Evolutivos de Reescritura	
Fernando Galindo Soria.....	137
III.4 Algunas Propiedades Matemáticas de los Sistemas Lingüísticos	
Fernando Galindo Soria.....	149
III.5 Sistemas Evolutivos de Lenguajes de Trayectoria	
Fernando Galindo Soria.....	169
III.6 Aplicación de la Lingüística Matemática a la Generación de Paisajes	
Fernando Galindo Soria.....	179
III.7 Sistema Evolutivo Traductor	
José Rafael Cruz Reyes.....	193

CAPÍTULO IV MATRICES EVOLUTIVAS

IV.1 Una Representación Matricial para Sistemas Evolutivos	
Fernando Galindo Soria.....	201
IV.2 Transformación al Espacio Dimensión-Valor	
Fernando Galindo Soria.....	209
IV.3 Diseño y Construcción de Sistemas Interactivos aplicando experiencias basadas en el Tratamiento de Imágenes	
Fernando Galindo Soria, Jesús Antonio Jiménez Aviña	217
IV.4 Reformed: Reconocimiento de Imágenes por medio de Redes Neuronales, Sistemas Evolutivos y Distribuidos	
Alejandrina Salazar Torres.....	233
IV.5 Aplicación de los Sistemas Evolutivos en el Análisis de Espectros de Rayos Gamma	
Luis E. Torres Hernández, Luis C. Longoria Gándara, Antonio Rojas Salinas.....	237
IV.6 Sistema Evolutivo para el Diagnóstico de Fallas en Máquinas Rotatorias	
Eduardo de la Cruz Sánchez, Luis C. Longoria Gándara, Rodolfo A. Carrillo Mendoza.....	239
IV.7 Sistema Evolutivo para el Reconocimiento de Símbolos Taquigráficos	
Irene Arzola Carvaja, José Rafael Cruz Reyes.....	241
IV.8 Sistema Evolutivo de Reconocimiento de Formas en Dos Dimensiones	
Diana Karla García García, Sergio Salcido Bustamante, Alfonso Ventura Silva.....	243
IV.9 Sistema Evolutivo Reconocedor de Textos	
Jesús Manuel Olivares Ceja.....	253

CAPÍTULO V SISTEMAS EVOLUTIVOS APLICADOS AL TRATAMIENTO DE IMÁGENES

V.1 Algoritmo Evolutivo para Tratamiento de Imágenes	
Ángel Cesar Morales Rubio.....	263
V.2 Sistema Evolutivo para Generar Figuras y Realizar Paisajes con Movimiento en 2D	
Carlos Olicón Nava.....	271
V.3 Sistema Evolutivo Graficador de Moléculas Orgánicas	
Jesús Manuel Olivares Ceja.....	275

CAPÍTULO VI SISTEMAS EVOLUTIVOS APLICADOS AL DESARROLLO DE SISTEMAS

VI.1 Generador de Sistemas como Núcleo de un Sistema Evolutivo	
Fernando Galindo Soria.....	285
VI.2 Eva: Evolución Aplicada	
Claudia Marina Vicario Solorzano, Francisco Javier Aguilar Vallejo	301
VI.3 Sistema Evolutivo Generador de Aplicaciones	
Jesús Manuel Olivares Ceja.....	311
VI.4 Sistema Evolutivo para el Control Estadístico del Proceso en un Ambiente Distribuido	
Guillermo Ambríz Carreón	319

CAPÍTULO VII SISTEMAS EVOLUTIVOS BASADOS EN CONOCIMIENTO

VII.1 Representación del Conocimiento	
Fernando Galindo Soria	331
VII.2 Sistema Evolutivo Constructor de Sistemas Expertos	
Javier Ortiz Hernández, Fernando Galindo Soria.....	341
VII.3 Biblioteca Automatizada	
Fernando Galindo Soria.....	347
VII.4 Sistemas Evolutivos Basados en Conocimiento	
Fernando Galindo Soria.....	357
VII.5 Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos	
Elsa Berruecos Rodríguez.....	365
VII.6 Sistema Evolutivo para Representación del Conocimiento	
Jesús Manuel Olivares Ceja.....	381
VII.7 Compositor Evolutivo de Música Virtual	
Horacio Alberto García Salas	389
Conclusión: Rumbo a la Competitividad Internacional en Informática	
Fernando Galindo Soria.....	399

Prefacio: Evolución

Fernando Galindo Soria¹

En este trabajo se presentan 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 rápido (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.

Los sistemas reales son sistemas evolutivos y están permanentemente interactuando con su medio para poder amoldarse y sobrevivir, y en esa interacción el sistema y el ambiente se impactan y transforman mutuamente a partir de los flujos que los cruzan.

¹ Fernando Galindo Soria escribió este documento el 28 de enero de 1998 siendo profesor-investigador en la Escuela Instituto Politécnico Nacional (IPN).

O sea que 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.

Las diferentes manifestaciones del fenómeno evolutivo son casos particulares de un fenómeno general, lo que cambia es la velocidad relativa de percepción de los observadores y la velocidad de la transformación o evolución. Donde, la velocidad de la transformación depende de la magnitud y velocidad del flujo de los factores esenciales y de la capacidad del sistema para transformarse (por ejemplo de su velocidad de absorción).

Conforme madura un proceso evolutivo es más difícil modificarlo (la velocidad del cambio disminuye), sin embargo sobre un individuo u organismo se llevan a cabo múltiples procesos evolutivos en forma fractal y paralela a diferentes velocidades. Por ejemplo los ciclos de nacimiento, crecimiento, reproducción y muerte, se solapan. El hecho de que uno de ellos disminuya su velocidad o desaparezca no quiere decir que otros no continúen o se aceleren, por ejemplo los procesos de crecimiento pueden disminuir y los de descomposición aumentar.

Aun más, un sistema puede transformarse, simplemente por estar en un espacio aunque en este medio no exista movimiento o sea cuando la velocidad del flujo es cero, por el puro hecho de existir el espacio puede impactar o catalizar un proceso y propiciar la transformación del sistema (por ejemplo, por el hecho de existir una mesa puede facilitar que se realice un proceso aunque ella no tenga una función activa en el proceso).

Dependiendo de la velocidad y comportamiento del cambio (ver pendiente, derivada, puntos singulares, atractores, límite, etc.) se marca el tipo de cambio: adaptación, evolución, revolución, reestructuración o catástrofe, por lo que el cambio es una constante permanente de los procesos evolutivos. Cuando la velocidad de la evolución no es la adecuada o cuando el sustrato donde se asienta sufre un cambio brusco puede sobrevenir la revolución, catástrofe, desintegración o paralización en un punto del proceso.

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.

El problema de lograr que algo inmerso en todo ese caldo de materia, energía e información (donde existen desde espacios desordenados hasta grandes corrientes y espacios de caos), impacte en la transformación de un organismo específico se puede empezar a atacar a partir de las siguientes consideraciones:

Todo lo que interactúa con el organismo específico es significativo y va adquiriendo mayor relevancia conforme mas veces interactua; si el organismo ha interactuado con pocos elementos todos son significativos aunque tengan poca ocurrencia. Si ha interactuado con muchos

elementos, los que tengan poca ocurrencia tienen un nivel de significado bajo y pueden llegar a ser imperceptibles.

Es como la creación de caminos, cuando la gente empieza a cruzar cotidianamente un espacio cubierto de césped transita indistintamente por diferentes lugares y el césped guarda una “memoria” tenue de sus pasos, sin embargo conforme pasa el tiempo se empiezan a marcar las veredas que terminan siendo los caminos naturales.

Conforme transcurre el tiempo se utilizan mas las veredas y los espacios poco usados tienden a integrarse a su entorno y llegar a desaparecer. Y si por algún motivo una vereda deja de usarse, también tiende a integrarse a su entorno.

Tal vez esta es una de las causas por las que el cerebro requiere un espacio “no usado” (se dice que solo usamos el diez por ciento de nuestra capacidad) ya que tal vez no es tan vacío o no usado como se cree, sino que actúa como un gran césped donde todo lo que pasa se graba aunque sea muy tenuemente y en su momento algunos aspectos se marcan y otros desaparecen y permanentemente esta cambiando la forma de ese espacio.

Cuando se esta modelando algo la estructura del modelo esta interrelacionada con la estructura del objeto o problema que se esta modelando. Si la arquitectura del modelo es mas simple o no corresponde a la del problema la modelación se puede complicar.

Por ejemplo, si la arquitectura natural del problema es en paralelo y se utiliza para modelarlo un sistema secuencial el método de solución se puede volver mas complejo y difícil de desarrollar, lo mejor es tratar de que la arquitectura del sistema se acerque a la del problema o lo absorba.

Por lo que, si quiero representar un sistema real mediante reglas estáticas o sistemas formales, tarde o temprano la representación se queda desfasada de la realidad. Los sistemas reales son sistemas evolutivos y la mejor forma de representarlos debe ser evolutiva.

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.

En particular, lo anterior se podría aplicar para entender y aprovechar el comportamiento de las diferentes manifestaciones de los procesos evolutivos, mediante el desarrollo y uso de herramientas de una “Ingeniería Evolutiva” (por ejemplo al aprender evoluciona nuestra imagen

del mundo, por lo que un buen ejercicio seria la creación de espacio de aprendizaje donde el conocimiento fluya libremente).

El hecho de que un sistema evolucione no significa que sea “mejor” en un sentido absoluto (mas grande, mas rico, mas humano, mas inteligente, etc.) sino que se adapta mejor a su ambiente y aumenta sus posibilidades de supervivencia. Un sistema evolutivo tiende a mejorar su interrelación con el medio y en general es “hermoso” y natural dentro de su contexto.

Introducción

Desde principios de los 80s en diferentes instituciones se han desarrollado trabajos teóricos y prácticos en el área de los Sistemas Evolutivos. La mayor parte de estos trabajos se han presentado en foros nacionales e internacionales, por lo que la intención de este libro es integrar en una obra los resultados obtenidos en las diversas aplicaciones en distintas disciplinas del conocimiento humano.

Las contribuciones pertenecen a empresarios e investigadores activos en su campo de conocimiento.

Los Sistemas Evolutivos representan un paradigma de conceptualización de la realidad en donde las cosas, el ambiente que las rodea, los individuos están en cambio constante, interaccionando unos con otros como nos lo ha presentado Fernando Galindo Soria en el prefacio de esta obra.

Considero que este libro tiene varias perspectivas para su lectura.

Desde una perspectiva, el lector puede encontrar trabajos de su interés en diferentes capítulos en donde se aplican técnicas que se complementan para dar un enfoque robusto al tema buscado. Por ejemplo, si alguien busca referencias sobre Reconocimiento de Imágenes se puede ver el trabajo de SI-VE en I.7, luego pasar a ver el concepto de Sistema Evolutivo en II.1 y II.2, continuando con los Lenguajes de Trayectoria en III.6 para adentrarnos de lleno en la técnica de Matrices Evolutivas en el capítulo IV en donde se presentan varios trabajos mostrando diversas técnicas y enfoques inspirados en las ideas Evolutivas. Como complemento puede revisarse el capítulo VI y VII para integrar al sistema de reconocimiento en alguna aplicación que podría ser basada en conocimiento. Bajo este mismo enfoque pueden revisarse temas como Desarrollo de Sistemas, Sistemas Basados en Conocimiento, Graficación por Computadora, Traducción de Lenguajes, Aplicaciones de Inteligencia Artificial y otros más.

Otra perspectiva es la temática, en base a esta, estructuramos el índice del libro, y en la cual pretendemos presentar trabajos teóricos y prácticos relacionados. Iniciando en los trabajos en los que se dieron las primeras aplicaciones de los Sistemas Evolutivos presentados en el capítulo I, luego los conceptos base y la arquitectura inicial de los Sistemas Evolutivos, en el

capítulo II. En el capítulo III concentramos los Sistemas Evolutivos que tienen aplicaciones basadas en la Lingüística Matemática. En el capítulo IV presentamos los desarrollos sustentados en la herramienta de Matrices Evolutivas. El capítulo V contiene las aplicaciones para tratamiento de imágenes. El capítulo VI nos presenta los trabajos de Sistemas Evolutivos aplicados al Desarrollo de Sistemas. En el capítulo VII agrupamos los Sistemas Evolutivos Basados en Conocimiento y temas relacionados con estos. Por último presento un trabajo que nos da ideas hacia donde dirigir nuestros desarrollos, en el sentido de que sean utilizados en un ámbito mundial.

Queremos agradecer a las personas que han hecho posible esta obra, tanto con sus aportaciones teóricas, prácticas, consejos y sugerencias, mismas que sería muy largo enumerar en este lugar.

Espero, en nombre de los autores, que esta compilación de trabajos le sea útil al lector en el ejercicio de sus actividades, además de disfrutarla como lo ha sido para un servidor.

Gracias por su atención.

Jesús Manuel Olivares Ceja
EDITOR

CAPÍTULO I

RUMBO A LOS SISTEMAS CONSCIENTES

I.1 Introducción General a los Sistemas Evolutivos

Jesús Manuel Olivares Ceja¹

En el umbral del tercer milenio vivimos una Revolución Informática, la realidad cambia constantemente, los seres humanos ya somos muchas veces los agentes que propiciamos los cambios en la naturaleza, en los negocios, en el conocimiento, en la manera en que percibimos el universo.

El cambio constante en la realidad nos hace buscar herramientas, métodos, modelos, etc. que nos faciliten y muchas veces que nos permitan establecer modelos para su estudio y comprensión de la manera en que se manifiesta con el propósito firme de pronosticar comportamientos.

De la realidad se toma la información que caracteriza al fenómeno en estudio, posteriormente se elabora un modelo y se aplican los resultados en la realidad, provocando cambios; nuevamente se recopila información, en un proceso donde el contar con la conceptualización evolutiva ha permitido que el proceso referido pueda captarse de una manera más cabal e integrada.

El paradigma de los Sistemas Evolutivos surge entre otros eventos, de observar la dificultad para mantener al día los sistemas de información, y que hoy día continua, es común que cuando se termina o se adquiere un sistema, ya se tiene que modificar o ampliar porque la realidad en que se encuentra ya cambió y que el mismo sistema fue propiciador de ese cambio. Con la consecuencia para quienes no se actualicen de quedar obsoletos o con un sistema que frena la evolución natural de las entidades.

Otro aspecto que caracterizó a los precursores del paradigma de Sistemas Evolutivos, fue la integración “natural” de diferentes disciplinas del conocimiento en donde los especialistas “acuerdan” en un lenguaje común, que es el del fenómeno en estudio.

Conforme los investigadores realizaron trabajos sobre el área, comenzaron a considerar algunas herramientas como típicas en las solución de algunos problemas, por ejemplo, en lo referente a la comunicación hombre-máquina, manejo de señales, generación de paisajes usando fractales, traducción de lenguajes y otros; se utilizaron herramientas de la Lingüística

¹ Jesús Manuel Olivares Ceja escribió este trabajo en abril de 1998.

Matemática. Una variante evolutiva de las redes neuronales (Matrices Evolutivas) le permitió a varios investigadores afrontar problemas en donde muchas veces no se tiene conocimiento de “como” se hacen las cosas, sino únicamente se conoce la entrada y la salida. Las áreas típicas de Desarrollo de Sistemas Basados en Conocimiento se vieron apoyadas con el paradigma de los Sistemas Evolutivos al contar con herramientas que les permitían obtener los programas y/o las reglas a partir de ejemplos que maneja el usuario en su área de especialidad.

Ejemplos de los resultados obtenidos son el Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos y el Sistema Evolutivo para Representación del Conocimiento, en el primero se obtiene el esquema de la base de datos y se actualiza mediante la descripción del ambiente en que se encuentra el sistema; en el segundo es posible encontrar reglas a partir del conocimiento que se le da en forma de oraciones declarativas en lenguaje natural restringido.

Varios de los trabajos realizados se aplicaron en diversas empresas, algunos de ellos se han convertido en productos de mercado y han permitido dar soluciones más “naturales” en el área de aplicación.

Es importante notar que los Sistemas Evolutivos representan más que una herramienta, una forma de conceptualizar la realidad, es decir, un lugar del universo en que los objetos inmersos están cambiando como resultado de su interacción, algunos cambios son muy bruscos y podemos tener conciencia de ellos pero otros no lo son, sino que tardan muchas veces miles o millones de años en apreciarse.

La invitación está abierta para apreciar nuestro bello universo considerando el paradigma de los Sistemas Evolutivos y así facilitar nuestra concepción y solución de problemas.

I.2 Desarrollo Histórico de los Sistemas Evolutivos

Fernando Galindo Soria²

Los Sistemas Evolutivos es un área de la Informática relativamente nueva ya que es alrededor de 1983 cuando se establece el concepto y es en 1986 cuando se presentan los primeros trabajos funcionando bajo este nombre, sin embargo tiene antecedentes desde hace muchos años en los conceptos de:

- 1) Máquinas que aprenden (Learning Machines, de Nilsson, escrito en 1966).
- 2) Redes Neuronales (McCulloch y Pitts desde 1939), demostración formal de la equivalencia entre Autómata Finito y Red Neuronal, Perceptrones).
- 3) Lingüística Matemática (trabajos de Chomsky sobre Gramáticas Generativo-Transformacionales desde 1957).
- 4) Desarrollo de Sistemas (Desarrollo Estructurado de Yourdon, Constantine y otros, durante los años 70s. Programación Estructurada, desde finales de los 60s. Desarrollo Lógico o de Warnier, durante los años 70s).
- 5) Base de Datos (concepto de Independencia de datos, bases de datos relacionales de Codd desde finales de los 60s, modelo de Chen de Entidad-Relación desde 1977).
- 6) Reconocimiento de formas (en particular Reconocimiento Sintáctico de Patrones, desarrollado por González y Tomasolo durante los 60s y el concepto fundamental de Inferencia Gramatical).
- 7) Fractales (propuestos por Mandelbrot desde principios de los 60s y desarrollados por el mismo desde mediados de los 70s).

Como se puede ver existe una gran cantidad de antecedentes “dispersos” sobre el tema y es hasta 1976 cuando en la Unidad de Cómputo de El Colegio de México (COLMEX), en la Escuela Superior de Física y Matemáticas (ESFM) del Instituto Politécnico Nacional (IPN) y en el Centro Nacional de Cálculo (CENAC) del IPN donde se comienzan a desarrollar las primeras investigaciones que culminarían en los Sistemas Evolutivos.

² Fernando Galindo Soria escribió este trabajo siendo profesor-investigador de la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) del Instituto Politécnico Nacional e Investigador Invitado del Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) de la Dirección General de Institutos Tecnológicos (DGIT) en la Ciudad de Cuernavaca, Morelos, noviembre 1988.

Durante 1976 se organiza en el CENAC un seminario sobre Inteligencia Artificial que duró aproximadamente un año y en el cual participaron personas de la ESFM, del COLMEX, de la Escuela Superior de Ingeniería Química e Industrias Extractivas (ESIQUE) del IPN, del Instituto Mexicano del Petróleo (IMP), de la Escuela Superior de Ingeniería Mecánica y Eléctrica (ESIME) del IPN y algunas personas de la Universidad Autónoma de México (UNAM); es dentro de este seminario donde se comenzó a encontrar la relación entre las diferentes áreas antecedentes y donde se empezó a tratar de aplicar estas herramientas para resolver problemas de reconocimiento del lenguaje en el Colegio de México.

Durante 1976-1981 se trabajó en forma extensiva en el desarrollo de herramientas para reconocer lenguajes en forma automática (compiladores, interpretes, constructores de compiladores) y de herramientas basadas en lenguajes (editores, manejadores de archivos, validadores de datos) por lo que para 1980 el problema ya no es la construcción de un traductor (tesis de Lino Díaz “Construcción de Compiladores”) sino la falta de mecanismos que permitan encontrar la estructura de un traductor (gramática o autómeta).

Es a principios de los 80s donde en el COLMEX y en la Maestría en Computación del CENAC se comienzan a buscar algoritmos que permitan encontrar en forma automática la estructura de un traductor y se empieza a utilizar la Inferencia Gramatical (desarrollada en los trabajos sobre Reconocimiento Sintáctico de Patrones) para este propósito.

A finales de 1980 se integra en la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) del IPN la Unidad de Investigación y Desarrollo en Computación (UIDC) y es en este marco donde se empiezan a integrar los trabajos anteriores con las investigaciones desarrolladas por Vicente López Trueba, Christian Zempoaltecatl y Ricardo García sobre los métodos y herramientas de desarrollo de sistemas, principalmente sobre lo que es la arquitectura de un sistema de información y los diagramas de Warnier y es en un seminario sobre informática, desarrollado a finales de 1981 donde se detecta que la estructura de cualquier sistema de información se puede representar directamente como una gramática; por otra parte y en la misma época en los cursos de Lingüística Matemática del CENAC y de la UPIICSA se comienza a estudiar por parte de los alumnos en los cuales el “lenguaje” del problema no es un “lenguaje natural”, como por ejemplo: la estructura de una sustancia química, el dibujo de cuadrados y triángulos y muchos otros (cada alumno desarrolla un proyecto independiente) es a partir de estos trabajos donde se comienza a vislumbrar que “tal vez” los problemas informáticos en general son susceptibles de un tratamiento lingüístico y donde se comienza a proponer un enfoque dentro de los cursos en el cual dado un problema se busca el lenguaje que lo representa y de ahí se encuentra su gramática y el sistema de información, por lo que para 1983 se comenzó a contar con un esbozo de lo que para 1985 se presentó como “Programación Dirigida por Sintaxis” (PDS).

Durante 1983 Gabriel Cordero, otro de los investigadores de la UPIICSA y en esa época Subdirector de Informática en la Secretaría de Educación Pública (SEP), desarrolla una herramienta orientada a manejar en un lenguaje natural restringido requerimientos sobre bases de datos.

Este trabajo permite en una forma relativamente sencilla cambiar los términos usados, de tal forma que dos personas que utilicen el mismo dato pero con nombre diferente lo puedan hacer. A partir de este trabajo y de desarrollos paralelos realizados por Roberto Mendoza Padilla y Juan Martín González Vázquez se comienza a pergueñar el concepto de Sistema Evolutivo como un sistema que es capaz de transformarse de acuerdo a los requerimientos del ambiente.

Para principios de 1985, se presentan en un congreso organizado en el Instituto Tecnológico de Estudios Superiores de Monterrey (ITESM) campus Monterrey los trabajos de estos tres investigadores y en junio de 1985 se presenta la ponencia “Sistemas Evolutivos” en varios foros, donde se plantea en forma general la problemática de los sistemas de información, el concepto de sistemas evolutivos, una primera propuestas de arquitectura, la factibilidad de desarrollar un sistema de ese tipo y los principales problemas que se estaban atacando en esa época. Este documento sirvió como guía de acción ya que permitió orientar la investigación sobre áreas más concretas, siendo tal vez una de las más relevantes la necesidad de dotar al Sistema Evolutivo con un mecanismo de tipo inductivo/deductivo, en contra de los mecanismos tradicionales de tipo deductivo.

Es en la búsqueda de un mecanismo de tipo inductivo donde la Programación Dirigida por Sintaxis adquiere su verdadera magnitud ya que para 1985 este método se encuentra documentado y ya tiene desde 1983 de estarse aplicando cotidianamente para resolver problemas de informática, depurándose a partir de la experiencia obtenida cada semestre, contándose ya con herramientas que automatizan algunas partes del proceso como por ejemplo la búsqueda de Unidades Léxicas a partir de un “Diálogo con el Usuario” y tienen un antecedente de gran magnitud en el proyecto de “Diccionario del Español en México” realizado en el COLMEX desde mediados de los 70s; la búsqueda de la gramática de un sistema (conjunto de reglas que representan la estructura del sistema) tiene como antecedente los métodos de inferencia gramatical y ya para 1985 es común que los alumnos desarrollen sus propios programas que en forma automática lleven a cabo el proceso de Inferencia Gramatical; y por otro lado ya se cuenta con herramientas que en forma automática reconocen un lenguaje a partir de su gramática (tesis de Lino Díaz), generan y construyen bases de datos (alrededor de seis tesis sobre el tema y múltiples generadores desarrollados por los alumnos del CENAC y de UPIICSA en sus cursos hasta 1985), generan en forma automática programas y sistemas (trabajos funcionando en el campo de López Trueba, Ricardo García y Christian Zempoaltecatl desde principios de los 80s)

Por lo que a partir del documento de “Sistemas Evolutivos” se ve como un proceso lógico la investigación de estas herramientas.

El primer problema que se presenta es el de que los métodos de Inferencia Gramatical no son de propósito general y son relativamente complejos de programar por lo que durante 1986 y 1987 en los cursos de Lenguajes Formales, Compiladores y Base de Datos principalmente se continua depurando el método de programación dirigida por sintaxis, buscando disminuir el énfasis matemático y aumentar el énfasis en su facilidad de manejo, campo de acción y buscando su automatización; principalmente en los cursos de base de datos y compiladores dirigidos por Vicente López Trueba, Gabriel Cordero y Alejandro Hernández se comienzan a

encontrar múltiples “trucos” para que la computadora reconozca y aprenda nuevos tipos de unidades léxicas. Y por otro lado en los cursos de lenguajes formales los alumnos comienzan a descubrir métodos de inferencia gramatical propios, destacando los métodos descubiertos por María Sandra Camacho Villanueva para el reconocimiento de imágenes ya que en una “forma natural” permite encontrar la representación de una imagen cerrada (que no se puede representar mediante una gramática libre de contexto) mediante una gramática con atributos, donde una gramática con atributos es un mecanismo en el cual se maneja inmerso dentro de las reglas gramaticales un conjunto de llamados a rutinas. Las gramáticas con atributos tienen sus antecedentes en el lenguaje REC (desarrollado por el Dr. Harold V. McIntosh) y programado múltiples veces en la ESFM y en los trabajos desarrollados ***en el CENAC*** precisamente sobre una herramienta gramatical con ese nombre y ya para 1980 se utilizaban en el CENAC y en la UPIICSA (primero para la construcción de compiladores y más adelante simplificando su notación y generalizándolas como parte integral del método de Programación Dirigida por Sintaxis para el manejo de la semántica).

Es también durante esta época 1986-1987 y a partir de los resultados directos de los trabajos desarrollados en lenguajes formales (cada alumno desarrolla un sistema completo usando la PDS) como se depura completamente el método y se detecta que existen dos operaciones básicas de inferencia gramatical, presentándose para el Congreso de Inteligencia Artificial desarrollado en la ciudad de Mérida, Yucatán en 1988 un documento donde ya aparece el método en forma completa y con una orientación al desarrollo de cualquier tipo de sistema.

Con la automatización creciente del método muchos de los problemas planteados para el desarrollo de Sistemas Evolutivos fueron prácticamente cubiertos, por lo que durante 1988 en los cursos de lenguajes formales dirigidos por Miguel Ángel Torres Aragón y Rafael Cabrera se desarrollaron algunos prototipos de Sistemas Evolutivos por parte de los alumnos y se presentaron en foros externos a la UPIICSA.

Por lo que prácticamente en la actualidad (noviembre de 1988) es “natural” para los alumnos de lenguajes formales estar desarrollando Sistemas Evolutivos utilizando las herramientas de la PDS.

Sin embargo desde 1986 se comenzaron a construir los primeros Sistemas Evolutivos utilizando estas herramientas y en la actualidad existen alrededor de 20 prototipos, desarrollados como trabajos de titulación, trabajos de cursos y trabajos de investigación, tanto por profesores-investigadores como por alumnos-investigadores principalmente de la licenciatura en Ciencias de la Informática de la UPIICSA y del CENIDET-DGIT de Cuernavaca, Morelos.

Los trabajos actuales marcan la inclusión de los fractales como herramientas para representar la estructura de fenómenos naturales ya que son “fácilmente” representables como gramáticas.

El manejo de gramáticas semánticas (gramáticas donde todos los elementos tienen significado). La ampliación de los mecanismos de captación de la realidad con el fin de que se pueda captar la información por múltiples fuentes.

El manejo de Sistemas Evolutivos Distribuidos. La obtención de herramientas generalizadas para no tener que estar programando todo a mano.

El desarrollo de Sistemas Evolutivos cada vez más generalizados y buscando llegar a que no se necesite programar (esta es una de las tendencias más fuertes y se ha desarrollado durante varios años por un grupo que originalmente trabajo en el área de efectos digitales de Televisa coordinado por Cuitlahuac Cantú Rohlik).

El desarrollo de una conciencia automatizada coordinada por Cuitlahuac Cantú Rohlik, Raúl de la Rosa y Ariel Carvoney estos últimos Director y Subdirector de Sistemas de la Secretaría de Programación y Presupuesto (SPP) respectivamente.

El uso extensivo para producción industrial de software mediante herramientas automatizadas desarrolladas por ellos mismos por parte de:

- 1) Vicente López Trueba, director del área de informática de la Secretaría de Comercio y Fomento Industrial (SECOFI).
- 2) Christian Zempoaltecatl, director del área de Sistemas de la Secretaría de Salud.
- 3) Ricardo García, director de la empresa Micrológica Aplicada.

La construcción y manejo para aplicaciones reales de sistemas Evolutivos por parte de:

- 1) Juan Martín González Vázquez, subdirector general de informática en la Secretaria del Trabajo.
- 2) Elsa Hernández Luna, subdirectora de informática en la SECOFI.
- 3) Alfredo Blancas, subdirector de informática en la Coordinación de Transporte del Departamento del Distrito Federal (DDF).

I.3 Aplicación de un Reconocedor de Lenguaje Natural a la Recuperación de Datos

Gabriel Cordero Sánchez³

Resumen

En este documento se muestra la estructura funcional de un reconocedor de lenguaje natural restringido, como una herramienta (de software) de uso general. También se describe su aplicación en la recuperación de datos (interactiva o reportes), actualmente de subsistemas de información administrativos (Recursos Financieros y Recursos Materiales) desarrollados en la Dirección General de Información Administrativa (DGIA) como apoyo en la administración de la Secretaría de Educación Pública (SEP).

El reconocedor, como herramienta de uso general, tiene la característica de operar a partir de una base de datos, en forma independiente a la aplicación particular que se trate.

La recuperación de datos, se basa en solicitudes de tipo imperativo en lenguaje natural restringido, las cuales pueden ser conocidas o no para el reconocedor debido a que una de sus facilidades es la capacidad de registrar (y en cierta forma aprender) frases a nivel de palabra por palabra, de lo que hasta ese momento son instrucciones de usuario desconocidas.

El reconocedor es un programa interactivo que reconoce frases imperativas mediante validaciones de tipo léxico-sintáctico usando una gramática y como resultado genera directivas de acceso a una base de datos. Es capaz de incrementar su lenguaje ampliando de esta manera su nivel de reconocimiento.

Introducción

El documento consta de tres partes fundamentales; la primera incluye a manera de antecedente una breve reseña de la evolución de los sistemas de recuperación de datos (aunque es una realidad que el enfoque primario aún se usa) y se enuncia el objetivo del reconocedor junto con algunas perspectivas de su desarrollo. La segunda parte consta de una descripción del esquema

³ Gabriel Cordero Sánchez publicó este trabajo en abril de 1985 en un Congreso Nacional. Lo inició en 1983 cuando era miembro de la Unidad de Investigación y Desarrollo del Departamento de Computación de la UPIICSA profesionalmente desempeñaba el cargo de Subdirector de Análisis y Diseño de Sistemas en la Dirección General de Información Administrativa (DGIA) de la Secretaría de Educación Pública (SEP).

conceptual (medio ambiente del sistema) y de los elementos del reconocedor, la cual define funcionalmente la operación del mismo dentro del subsistema de recuperación de datos. Por último se presenta el flujo de operación del reconocedor desde el punto de vista interno y da una idea más clara de la estructura de su construcción. Se comenta la posibilidad no solo para la recuperación de datos, sino para la construcción de herramientas más avanzadas.

I Antecedentes

Es evidente (para aquellos que estén relacionados) que ha existido un desarrollo importante en la metodología tanto del análisis y documentación de sistemas como para el diseño y construcción de programas. Lo anterior ha tenido como consecuencia, precisamente basada en la oportunidad de contar con nuevas y mejores herramientas; una evolución en los métodos para recuperación de datos, básicamente de los sistemas en los que están involucrados los datos (viviendo en algún dispositivo de almacenamiento) y los usuarios (que bajo cierto tratamiento —proceso— y a través de un medio, usualmente físico, requieran esos datos como parte de la información necesaria para su actividad dentro del sistema de información de la organización a la que pertenecen).

Una vez que la computadora pasó a ser una herramienta para las actividades operativas de la administración de cualquier organización, la necesidad de establecer un mecanismo de comunicación entre los requerimientos de un usuario y la posibilidad del proceso masivo de datos para su presentación (tradicionalmente en reportes) creció tal vez en mayor proporción que el número de aplicaciones que podían introducirse a una computadora. Esto dio como resultado que hasta ese momento los usuarios, trataran de entender el lenguaje (terminología) de los programadores y estos últimos trataran de conocer y representar la aplicación de aquellos. Surge una nueva disciplina, llamada, el análisis de sistemas administrativos, teniendo su origen en usuarios transformados en programadores y programadores transformados en usuarios. Hoy todavía, existen organizaciones que basan sus aplicaciones en este enfoque tradicional.

En la figura 1 se muestra como fue la relación entre un usuario y sus aplicaciones en una computadora. Un usuario o grupo de usuarios, normalmente con el deseo de automatizar las actividades operativas y con manejo de grandes volúmenes de datos, a través de un intérprete (comúnmente un programador o analista-programador y en el mejor de los casos un experto en sistemas administrativos favorablemente conocido como analista), cuya función además de decidir los datos necesarios para el usuario, era la de interpretar los requerimientos de este último y con esto elaborar un planteamiento del problema para de inmediato, dar paso al diseño y construcción de los programas indispensables en este caso. El usuario obtiene los reportes previa definición del programador, que contienen sus datos (del usuario) previa definición también del programador.

Con la introducción de metodologías para el desarrollo de sistemas y la natural evolución humana en tecnología, la recuperación de datos toma el sentido interactivo como medio principal para la obtención de información. En la figura 2 se puede ver como es el flujo de datos del sistema de recuperación bajo este enfoque, digamos más actual. Este tipo de recuperación de

datos, surge del desarrollo de sistemas de aplicación de software a la solución de problemas en el manejo de datos de un usuario en particular. El usuario auxiliándose con el intérprete (para este caso un analista de sistemas administrativos), logra definir sus requerimientos para el proceso y la presentación de sus datos.

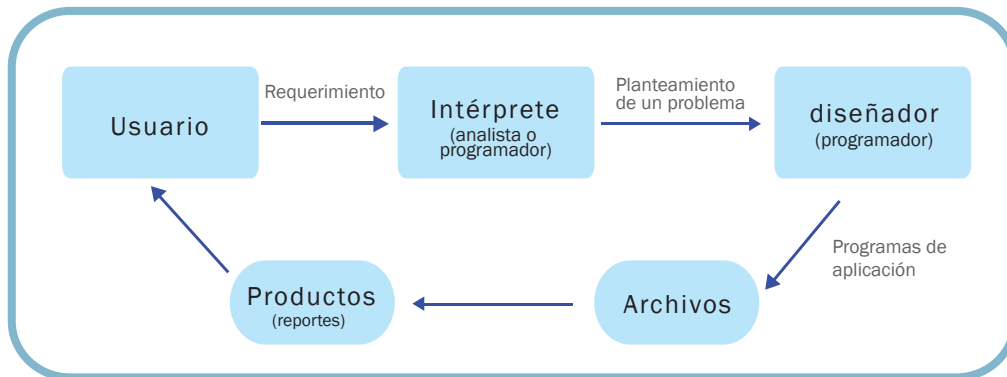


Figura 1. Recuperación de datos basada en el enfoque tradicional del desarrollo de aplicaciones.

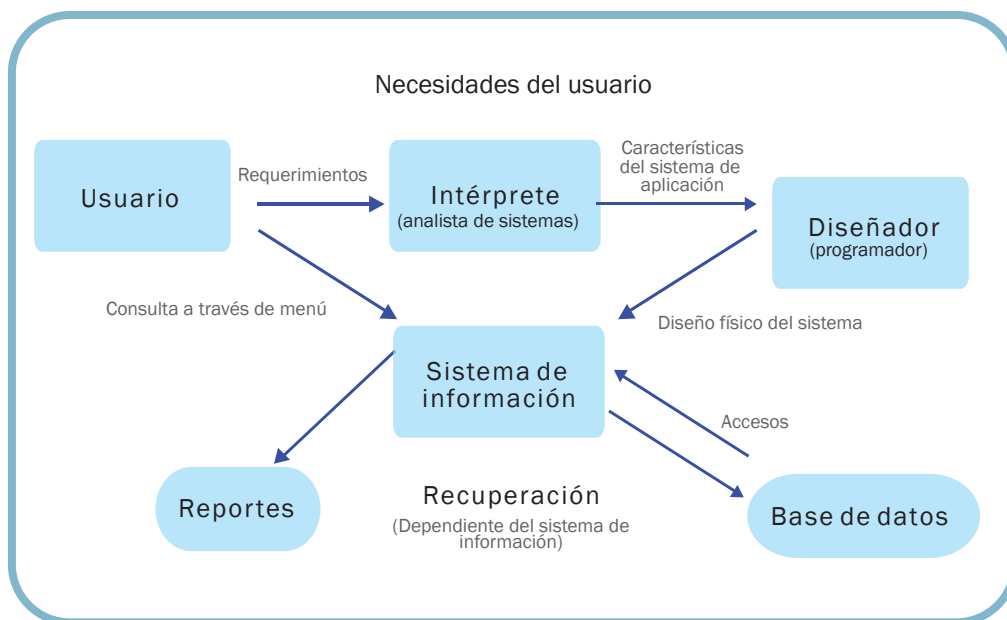


Figura 2. Recuperación de datos basada en el análisis de un sistema de información.

Estos requerimientos son planteados por el analista (para ese entonces un experto en las actividades del usuario) al diseñador de sistemas, quien dependiendo del enfoque del analista y del propio, procede con la definición del “sistema” para satisfacer las necesidades del usuario.

Para este enfoque de la recuperación de datos, el diseñador de sistemas, haciendo uso de paquetes de software comercial define el sistema de aplicación en función de bases de datos exclusivas para un problema de un usuario en particular, y basa la operación de recuperación a través de un menú de opciones, obligando con esto a que el usuario en principio, domine los

términos del lenguaje en que fue definido el menú por el diseñador, **restringiendo además la posibilidad de la recuperación a sólo las opciones de ese menú**. Esto implica que un nuevo requerimiento del usuario tiene que ser diseñado y programado para su inclusión al sistema de aplicación (menú).

El estudio de organizaciones y su representación como sistemas de información ha venido ocasionando que la filosofía de construcción de software cambie hacia un enfoque menos dependiente de los sistemas de aplicación, conduciendo así al **diseño de software de uso general, que se distingue principalmente por la independencia entre el usuario** (como elemento del sistema de información de una organización), **y el sistema de aplicación**, diseñado específicamente como herramienta de apoyo a las actividades de ese usuario dentro de la organización. Este principio, el de **concebir los sistemas de aplicación como una herramienta y no como la solución**, hace necesario el contar con herramientas que permitan al usuario nombrar y tratar a los datos en términos del lenguaje del propio usuario, y no en los que defina el analista (característica de los dos enfoques anteriores), bajo el mismo principio es como fue construido el reconocedor descrito en este documento.

La figura 3 contiene una representación de lo descrito anteriormente, donde varios usuarios a través de un intérprete, en este caso **software de uso general** y no una persona como se vio anteriormente, auxilian su actividad usando el mismo sistema de información soportado en una base de datos general. Por lo que el objetivo primario del reconocedor y de hecho su origen, es evitar que un usuario durante la recuperación de sus datos se vea en la necesidad de conocer términos de computación no indispensables para su propósito.

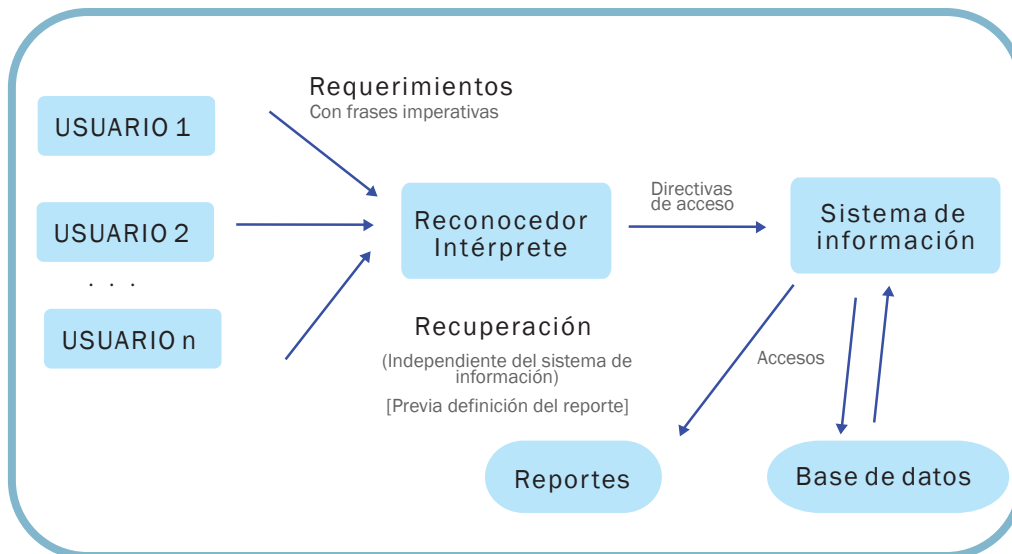


Figura 3. Recuperación de datos basada en el análisis y requerimientos del sistema de información de una organización (construcción de herramientas de uso general)

Este planteamiento deriva en varios objetivos de carácter secundario, ellos son:

- 1) *Facilitar para el usuario la recuperación de sus datos en términos de su propio lenguaje.*
- 2) *Eliminar la dependencia entre la recuperación de datos y los sistemas de información.*
- 3) *Construir una herramienta de uso común y general (no dependiente de algún sistema de información)*
- 4) *Construir una herramienta capaz de registrar nuevos términos de lenguaje de un usuario o de nuevos usuarios (aprendizaje)*

2 Operación de la Recuperación de Datos

La figura 4 muestra un ejemplo de la forma en que se usa el sistema de recuperación de datos, tanto en forma interactiva (por terminal), como por reporte (previa definición). La operación se limita a la solicitud en forma imperativa de algunos datos por consultar, una condición es indicar el nombre del dato deseado dentro de la frase, para poder realizar la consulta (recuperación); Otro requisito necesario para efectos de reconocimiento es el de proporcionar al reconocedor entre // (diagonales) el valor de un dato del que se desee consultar sus características, por ejemplo:

Usuario: Dame el artículo /510101005/

Sistema: 510101005, cinta magnética, 1,200 pies, pieza, ...

Esta misma estructura y condiciones son las necesarias para la generación de reportes, que son distinguidos de la presentación en terminal desde el momento en que se inicia una sesión con el reconocedor.

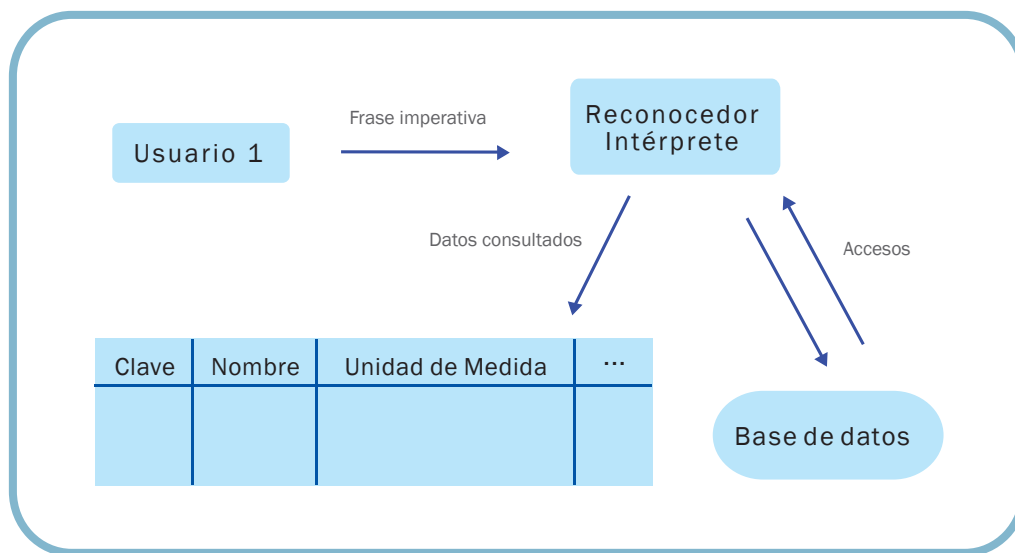


Figura 4. Operación de la recuperación

Las dos condiciones anteriores y las intrínsecamente relacionadas con la gramática incluyendo los elementos del lenguaje y el simple hecho de que el usuario conozca los datos (nombres de los datos) que maneja, constituyen los únicos requerimientos (de entrada) para que el sistema de recuperación pueda ser usado.

Otro punto muy importante y que forma parte del reconocedor es dar la facilidad al usuario para que registre nuevas palabras, es decir, que el reconocedor incremente su lenguaje y con esto su nivel de reconocimiento, las palabras que pueden registrarse en el catálogo de gramática pueden ser o no sinónimos de otras ya existentes. Un ejemplo de una sesión donde se tiene esto es:

Usuario: Dame el objeto /510101005/

Sistema: 'objeto' no la tengo registrada, seleccione un sinónimo de entre: artículo, dependencia, pedido, presupuesto, saldo, ...

Usuario: artículo

Sistema: 'objeto' es sinónimo de 'artículo', gracias.

Sistema: 510101005, cinta magnética, 1,200 pies, pieza, ...

De la misma manera, existe para cada solicitud de datos una forma de presentación. Esta forma de presentación está predefinida y registrada en los directorios correspondientes, (directorio de reportes, directorio de pantallas).

En el caso de la recuperación por pantalla, la presentación de los datos consultados puede ser de dos tipos. Una presentación hecha a través de formatos de pantalla predefinidos con una forma especial (pantalla fija), la otra presentación (de datos) se realiza por medio de una pantalla de formato general que se usa parcialmente condicionada por la misma solicitud (pantallas variables).

La predefinición de los formatos de presentación de datos (formatos de salida) es el único requerimiento (de salida) para una solicitud de consulta, interactiva o por reporte.

3 Esquema Conceptual del Sistema de Recuperación de Datos Utilizando un Reconocedor de Lenguaje Natural Restringido

Como puede notarse hasta esta parte del documento; el reconocedor como objeto principal de esta presentación, es solo un elemento más de lo que forma el sistema de recuperación de datos utilizando lenguaje natural restringido (SRD-LNR).

Como puede observarse en la figura 5, el SRD-LNR consta entre otros, de cuatro elementos, tal vez los más importantes: el primero de ellos es el usuario, quien finalmente es el afectado en forma favorable o desfavorable por la oportunidad, claridad y facilidad con la que pueda obtener información, que para efectos de esta aplicación en particular, están almacenados formando una base de datos. El (o los) usuario (s) usando una terminal, para efectos de la recuperación de datos, puede emitir tres clases de instrucciones (la mayoría de carácter imperativo y algunas asociativo):

- 1) *Solicitar la presentación* (consulta de algún dato o datos), para los cuales esta autorizado, y a través de la misma terminal.
- 2) *Definir un reporte* de sus datos o parte de ellos usando uno de los módulos del generador de reportes, en el que prácticamente la presentación del reporte la decide el usuario.
- 3) *Solicitar la emisión de un reporte* previamente definido indicando su referencia y el conjunto de datos que incluirá.

Las instrucciones 1) y 3), no importa de cual se trate, son recibidas directamente por el reconocedor, el segundo de los elementos del SRD-LNR, el reconocedor una vez identificada e interpretada la instrucción, basado en la gramática ya definida, tiene la función de invocar al generador de reportes o al módulo correspondiente del sistema de información, estos últimos, reciben, producto de la operación del reconocedor un conjunto de directivas de acceso, cuya ejecución da como resultado contar con los datos deseados listos para su presentación.

El sistema de información, tercer elemento del SRD-LNR, es el módulo encargado de realizar cualquier tipo de acceso a la base de datos a partir de las directivas generadas por el reconocedor. Adicionalmente auxiliándose por un manejador de pantallas y un directorio de datos, tiene la función de formatear los datos solicitados cuando estos sean presentados por la terminal. Este es uno de los caminos por el que el usuario recibe respuesta del SRD-LNR.

El generador de reportes, cuarto elemento del SRD-LNR y otro de los caminos por los cuales el usuario recibe respuesta a su solicitud, se auxilia del directorio de datos y basa su operación en las directivas emitidas por el reconocedor, en los accesos a datos por el sistema de información y en los reportes previamente definidos por el usuario y registrados en el directorio de reportes.

Todos los anteriores y algunos otros que serán descritos posteriormente, forman el medio ambiente del reconocedor y son parte del sistema de recuperación de datos.

I El Reconocedor

En el punto anterior, se describió muy generalmente la función del reconocedor, y siendo éste el objeto del documento actual, será descrito a continuación en forma más detallada, conservando el marco trazado por el esquema anterior, y sin pretender con esto establecer la estructura interna y la construcción del reconocedor de no interés para los propósitos de esta presentación.

Tomando como referencia el diagrama de flujo de la figura 6, partimos nuevamente del elemento principal de la recuperación de datos, el usuario, que como ya se explicó solicita datos por medio de instrucciones de tipo imperativo y asociativas. Estas instrucciones son captadas en principio por el módulo de control de accesos, que interactuando con el módulo reconocedor (no confundir con el reconocedor como elemento del sistema), tiene la función de verificar que (de acuerdo al tipo de consulta) el usuario esté autorizado para realizar la consulta de los datos solicitados. Los datos validos por usuario se encuentran registrados en el directorio de accesos.

Una vez que un usuario ha sido identificado como válido, se hace el reconocimiento de la instrucción del usuario apoyado en la gramática previamente definida. Durante el reconocimiento de la instrucción pueden ser detectadas palabras desconocidas, las cuales, el módulo de ampliación del lenguaje con ayuda del usuario; puede asociarlas con algunas otras palabras ya conocidas o registrarlos como palabras nuevas, esto permite que el reconocedor aprenda a reconocer nuevas frases.

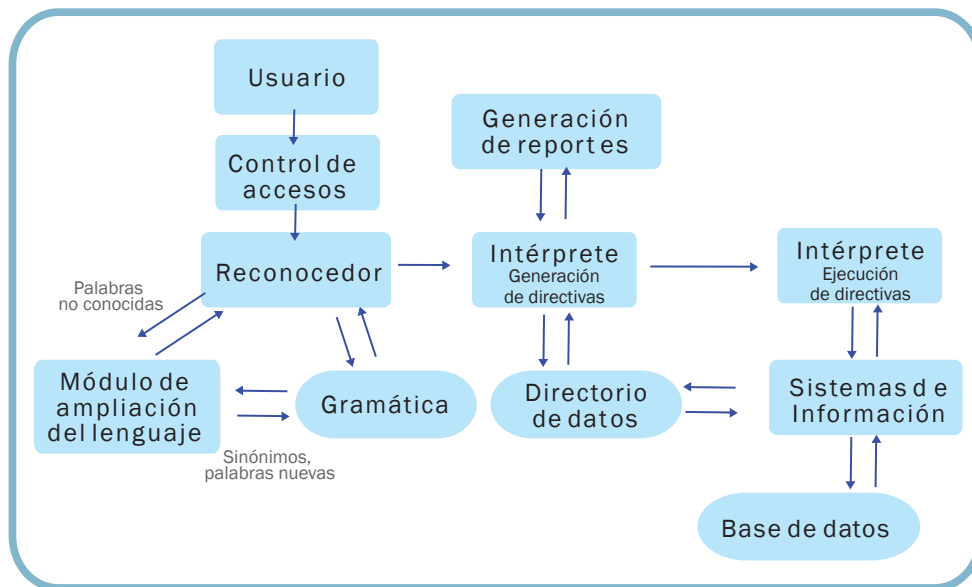


Figura 5. Elementos de sistema de recuperación de datos utilizando Lenguaje Natural Restringido (SRD-LNR)

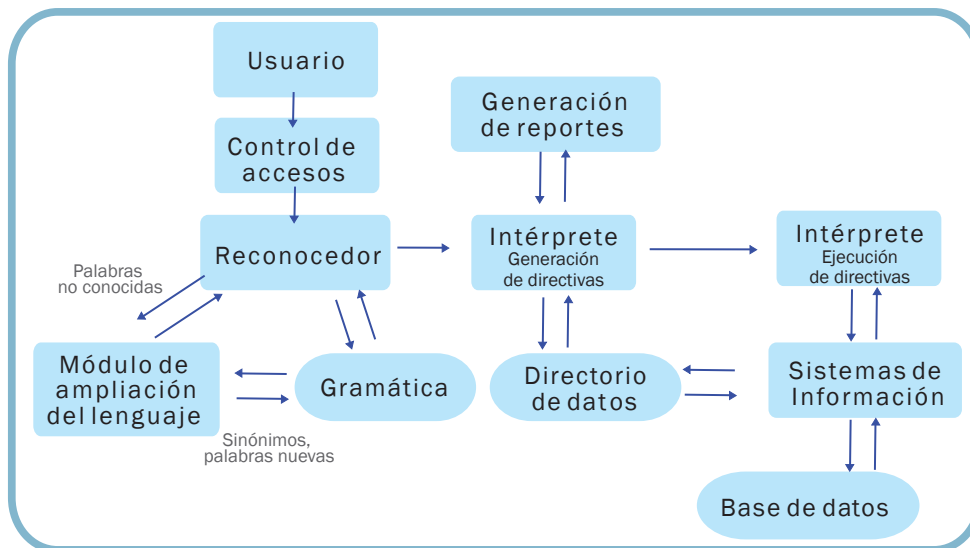


Figura 6. El Reconocedor

Cuando una frase ha sido totalmente reconocida, el reconocedor codifica las instrucciones en el orden en que fueron encontradas en la solicitud del usuario. Producto del reconocedor es una tabla de instrucciones ya codificadas y datos para las instrucciones.

Esta tabla es usada por el módulo intérprete que en una primera fase hace una clasificación de las instrucciones basado en las características y en las relaciones de jerarquía y dependencia de los datos solicitados, ya clasificados genera una secuencia de directivas de acceso que pueden ser válidas tanto para el generador de reportes como para el módulo del sistema de información.

La segunda fase del intérprete consiste en ejecutar las directivas de acceso generadas, por lo regular las que involucran al generador de reportes. El control de esta fase es compartido por el intérprete y el sistema de información, si la consulta solicitada requiere de intervención del usuario. Por último, los datos solicitados, también soportados por el directorio de datos, son formateados con ayuda del manejador de pantallas, que permite que el usuario defina la forma en que deberán ser presentados en la pantalla de la terminal, de esta forma es como se establece el ciclo completo de la solicitud de consulta de un usuario hasta la presentación de los datos por la terminal.

El otro camino para obtener los datos, ya mencionado, que es a través de reportes y la forma en que opera el módulo del sistema de información, serán descritos de manera particular en los siguientes párrafos, aunque en forma breve por no ser el interés principal del presente.

2 El Generador de Reportes

En esta parte, el usuario puede tener comunicación con tres módulos principales del generador de reportes. El módulo de definición de reportes, a través del cual se puede definir la estructura características y contenido de un reporte.

El reporte es registrado en el directorio de reportes previa validación de los datos del contenido contra el directorio de datos. La estructura considera el encabezado, cuerpo y pie del reporte en definición. Como características se entiende las condiciones de clasificación, salto y corte de control del reporte.

El módulo de definición de archivos, permite definir los archivos de datos a partir de los cuales se pueden obtener reportes, además de los que se obtienen a partir de la base de datos. Este módulo, aunque no representado en la figura, sirve de fuente de entrada para el directorio de datos.

Por último, el módulo de explotación o generación de reportes, cuya operación se soporta en reportes, archivos y datos ya definidos como se explicó, y se orienta por las directivas emitidas por el reconocedor con base en las instrucciones del usuario. Cuando el reporte debe generarse a partir de la base de datos, el generador de reportes obtiene los datos a imprimir vía el módulo de control de accesos del sistema de información.

3 El Sistema de Información

El sistema de información, elemento final de la recuperación de datos, tiene dos funciones básicas distribuidas en sus tres módulos principales. La primera, realiza los accesos a la base de

datos con las directivas emitidas por el reconocedor. El módulo de control de accesos es encargado de esta función que conduce a dos posibles salidas, a través del generador la obtención de un reporte y el formateo de los datos accesados, apoyado en el directorio de datos. El formateo de datos se hace de acuerdo a la forma de presentación requerida también previa definición usando el manejador de pantallas.

El último módulo, presentación de datos, se encarga de pintar los datos en el orden requerido en la pantalla de la terminal usada por el usuario.

Este conjunto de módulos fue identificado bajo el nombre de sistema de información por la razón de que es ésta la última parte de la recuperación y en la que están definidas las diferentes formas de presentación de datos, que en ultima instancia fueron solicitadas por el usuario y constituyen uno de los principales apoyos para la toma de decisiones y para guiar su actividad dentro de la organización.

4 Flujo de Datos Durante la Recuperación

Tratando de dar una idea de la estructura interna y de las herramientas que fueron usadas para la construcción del reconocedor y del sistema de recuperación de datos en general, ha sido incluido este párrafo que describe el flujo de datos, desde la solicitud del usuario hasta la respuesta del sistema obtenida por éste.

La instrucción del usuario es descompuesta en palabras, cada una de ellas y su orden, es validada contra la gramática dando la opción a registrar (almacenar en la gramática), palabras no conocidas mediante asociaciones y decisiones del usuario. Una instrucción del usuario puede generar varias directivas de acceso, estas son colocadas en una cola en el orden en que fueron encontradas, simultáneamente asociado a la cola de directivas se genera una lista de los valores de datos que el usuario solicitó. Por medio de una lista bidireccional esta cola de directivas se clasifica, de acuerdo al orden de acceso de los datos solicitados, según el directorio de relaciones. Hecho lo anterior, se procede con la ejecución de las directivas, controlando con una pila (stack) aquellas directivas de ciclo, es decir aquellas que fueron originadas por una solicitud en la que el usuario desea obtener: datos, un conjunto de datos, un conjunto de datos de un conjunto de datos, etc.

La última fase tiene dos salidas, una usando los datos ya accesados, se presentan en un reporte, que como ya fue mencionado, debe estar definido. La otra salida se realiza a través de la pantalla de la terminal de usuario, misma por la que hizo su solicitud, esta parte se auxilia de un manejador de pantallas para efectos de formateo y presentación de los datos.

Conclusiones

El reconocedor, como objeto principal de esta presentación, no significa por si mismo la creación de una herramienta sin igual, tecnológicamente hablando en lo que a computación se refiere; significa en cambio, la factibilidad, y realidad de hecho, del nivel de herramientas que

pueden ser construidas y usadas, en un ambiente en el que posiblemente existan limitaciones materiales pero no de imaginación. La filosofía empleada en esta aplicación de la recuperación de datos y el reconocedor concretamente, establece las bases para el desarrollo y construcción de herramientas con mayor flexibilidad, orientadas hacia el usuario final.

Algunas de las posibles herramientas que pueden ser construidas basadas en el reconocedor son:

- 1) *Generadores de programas o programación en lenguaje natural*, estos solo con una ampliación al reconocedor para que sea capaz de interpretar instrucciones de tipo condicional.
- 2) *Bases de conocimiento* que podría también ser usadas para la recuperación de datos y reduciría en mucho la longitud de la instrucción de solicitud de consulta del usuario.
- 3) *Otra aplicación* y que ya se esta desarrollando es la de permitir a través del reconocedor no solo la recuperación de datos sino además la actualización, ambas con instrucciones de tipo imperativo.

Es importante, al menos por ser un hecho que pudo observarse con el uso de esta herramienta, hacer notar como la persona que opera con el reconocedor (y cabe mencionar que esta persona ha operado con otros sistemas de recuperación de datos), como realmente lo único que requiere es conocer la información que maneja en su trabajo, y al percatarse de no tener una vía preestablecida para obtener datos suele decidir por mejores opciones (presentaciones de datos) diferentes a las que de origen fueron planteadas por el usuario mismo. Esto tiene como consecuencia que un usuario vea y sienta al sistema como una verdadera herramienta y no como un algo normativo que lo eduque, pensamiento, bajo el cual aún se rigen muchos desarrolladores de sistemas.

Bibliografía

- [1] Knuth, Donald, "*The Art of Computer Programming*" vol. I y vol. III
- [2] Vetter, "*Data Base Design Methodolgy*"
- [3] Hopcroft, Aho, "*Introduction to Automata Theory and Languages*"
- [4] Warnier, "*Logical Construction of Programs*"
- [5] _____, "*SAASEP: Sistema de Apoyo Administrativo de la Secretaría de Educación Publica, Subsistema de Pago*"
- [6] _____, "*Feature Analysis of Generalized Database Managment Systems CODASYL Systems Comitee Technical Report*", mayo 1971.

I.4 Introducción a la Inteligencia Artificial

Roberto Mendoza Padilla⁴

En este trabajo se da una descripción introductoria de un área que ha provocado un gran número de controversias, debido a que combina conceptos tanto de ciencias exactas como de las no exactas, esta área es una rama de la informática la cual surgió debido a que los métodos tradicionales no lograban resolver ciertos tipos de problemas, o bien, para automatizar procesos que requerían más bien de la intervención humana debido a su alta complejidad. Esta área puede encontrarse bajo distintos nombres, de acuerdo al problema específico que se trate, teniendo entre otros:

- 1) *Inteligencia de máquinas.*
- 2) *Programación heurística.*
- 3) *Simulación del conocimiento.*

Aunque en realidad estos y otros temas pueden considerarse como subáreas de estudio de la Inteligencia Artificial.

En este punto cabe detenernos y hablar acerca de como es que los expertos en la materia pueden considerar que un sistema tiene o exhibe una o varias propiedades de algo que se considera inteligente:

Se dice que un sistema tiene la propiedad de inteligencia, si al observar su comportamiento se ve que puede adaptarse a situaciones novedosas, si tiene la capacidad de razonar, de entender las relaciones entre sucesos, de encontrar significados y de reconocer la verdad, además que se espera que un sistema con estas características pueda aprender, es decir, mejorar su nivel de eficiencia bajo la base de experiencias pasadas, así como tener una conciencia plena del medio ambiente que le rodea, conociendo cuales son sus órganos funcionales y para que le sirven, así como cuales son sus limitantes, pudiendo inclusive tener la capacidad de autoconfiguración y autorreparación principalmente.

⁴ Roberto Mendoza Padilla escribió este trabajo siendo miembro de la Unidad de Investigación y Desarrollo en Computación (UIDC) de la Unidad Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) en julio de 1985.

Esta área ha sido alimentada por un gran número de enfoques, los cuales pueden agruparse en:

- 1) *El enfoque de la ingeniería*, en el cual el investigador intenta crear un sistema que sea capaz de enfrentarse con tareas intelectuales, sin tomar en cuenta si los métodos y técnicas utilizadas para tal fin, son idénticas o similares a los usados por el hombre; bajo este enfoque se tiene el compromiso de desarrollar sistemas económicos, eficientes y confiables (por ejemplo, reconocimiento de caracteres escritos, como los de un cheque bancario).
- 2) *El enfoque de modelación*, en donde el objetivo de investigación básico es tratar de obtener alguna comprensión de los mecanismos internos de un sistema vivo explicando y prediciendo su comportamiento (por ejemplo, proyectos que simulan la resolución de problemas humanos en una toma de decisiones o la elaboración de un juicio, mediante la utilización de redes neurales o sistemas expertos entre otros).

El enfrentamiento a los problemas tradicionales o a otros nuevos, utilizando este tipo de enfoques ha surgido por un sinnúmero de razones, sin embargo estas pueden agruparse como sigue:

- 1) *Reemplazar la inteligencia humana*, porque esta es muy cara, escasa y a menudo no muy confiable, siendo además perecedera.
- 2) *Establecer teorías de la inteligencia humana*, en forma de modelos de simulación, a aplicarse posteriormente en algorítmica de aprendizaje, resolución de problemas y prueba de teoremas entre otros.
- 3) *Desarrollar sistemas de software y hardware más eficientes*, como por ejemplo, sistemas con tolerancia de fallas, sistemas reconfigurables, sistemas que se autorreparen y sistemas que sean confiables, a pesar que sus componentes presenten fallas o deficiencias, entre otros.

Sin embargo y como en todas las áreas de conocimiento, los intentos por desarrollar estas ideas se han visto obstaculizadas por muchos problemas, entre los cuales podemos citar:

- 1) *Representación del problema*, es decir, buscar o crear las herramientas y técnicas necesarias para “introducir” la descripción del problema a una computadora, para que pueda ser resuelto.
- 2) *Generalidad contra eficiencia*, pues al inicio se pensó que se debían desarrollar sistemas que fueran capaces de resolver un sinnúmero de problemas, tal como un ser humano, sin embargo pronto se vio que era mejor enfocarse a un problema específico, pues el nivel de desarrollo de esta área no es suficiente para cumplir con los requerimientos de la generalidad, pudiendo decir en resumen que, actualmente **la inteligencia humana es de naturaleza múltiple, mientras que la inteligencia artificial se orienta a la resolución de casos concretos**.
- 3) *Búsqueda*, debido que aún en la actualidad es difícil realizar pruebas de eficiencia sobre el comportamiento de sistemas, llegando incluso al caso paradójico de afinar el sistema cada vez con mas pruebas, hasta un punto en el que se considera correcto, pero sin la seguridad de que el sistema funcione correctamente para los casos de prueba iniciales, con la consideración adicional de que las pruebas son solo una ínfima proporción de la generalidad de casos que se le presentarán al sistema en la realidad.

Como se puede apreciar, hasta este momento hemos establecido un marco general de referencia el cual nos conduce ahora a mencionar concretamente cuales son las áreas específicas de acción de la inteligencia artificial, pudiendo agruparlas bajo los siguientes rubros:

- 1) Sistemas expertos.
- 2) Bases de conocimiento.
- 3) Redes neuronales.
- 4) Sistemas de Información: Adaptativos, Dinámicos, Evolutivos.
- 5) Reconocimiento de formas: Análisis de ondas, Percepción remota, Lectura de escritos, Análisis de imágenes.
- 6) Lenguaje natural.
- 7) Prueba de teoremas.
- 8) Educación por computadora.
- 9) Juegos.
- 10) Toma de decisiones.
- 11) Composición musical.
- 12) Confiabilidad de sistemas de hardware y software.
- 13) Robótica.
- 14) Programación automática.
- 15) Cibernética.

Esta clasificación se ha presentado solamente por ubicar al lector, sin embargo es muy importante remarcar que cualquier aplicación puede resolverse utilizando un conjunto de principios fundamentales, los cuales nos ayudan a que un sistema sea inteligente, pues en la realidad, como se mencionó en un principio, la inteligencia artificial se le conoce con varios nombres debido al problema que ataca, asimismo, se han enumerado a las áreas específicas de acción bajo una gran diversidad de nombres, pero detrás de estas existen los mismos principios fundamentales de:

- 1) Adaptarse.
- 2) Razonar: deducción, inducción, elección de alternativas, etc.
- 3) Relación de sucesos.
- 4) Encontrar significados.
- 5) Reconocer la verdad.
- 6) Aprender por experiencia.
- 7) Generación de conocimiento: evolución.

Las técnicas, herramientas y métodos utilizados para lograr que un sistema posea uno o varios de estos principios fundamentales son muy variados encontrando en la mayoría de los sistemas una combinación de ellos, utilizando entre otros enfoques los provistos por las siguientes áreas:

- 1) Análisis de sistemas.
- 2) Teoría de autómatas.

- 3) Gramáticas.
- 4) Lenguajes formales.
- 5) Compiladores.
- 6) Redes.
- 7) Teoría de gráficas.
- 8) Simulación matemática.
- 9) Lógica simbólica.
- 10) Teoría de la comunicación.
- 11) Fisiología.
- 12) Psicología.

En este momento quizá se pregunte el lector como puede automatizar un procedimiento el cual esta siendo manejado con ayuda de la inteligencia humana.

La única consideración a este respecto es la que se refiere a que este procedimiento a automatizar debe poder especificarse y resolverse mediante sucesión de pasos desde su principio a su final, pues algo que ni aún nosotros mismos sabemos como se resuelve, menos lo podrá hacer una computadora.

Conclusiones

Finalmente no queda más que hacer extensiva la invitación hecha por muchos otros investigadores referente a que se introduzcan al estudio de esta área la cual nos ofrece nuevas y muy amplias expectativas de desarrollo en el área de la Informática como ninguna otra en estos días, teniendo el potencial suficiente como para proponer soluciones alternativas a las grandes preguntas: los físicos se preguntan que clase de lugar es el universo y buscan caracterizar su comportamiento sistemáticamente. Los biólogos se preguntan que significa para un sistema físico el estar “vivo”. Mientras tanto nosotros en Inteligencia Artificial nos preguntamos que clase de sistemas de información pueden contestar estas preguntas.

Bibliografía

- [1] Ralston, “Encyclopedia of Computer Science”
- [2] _____, “Handbook of Artificial Intelligence”
- [3] Mendoza Padilla, Roberto, “Sistemas de Informacion Adaptativos”, UIDC-UPIICSA, México.
- [4] Galindo Soria, Fernando, “Sistemas Evolutivos”, UIDC-UPIICSA, México.

I.5 Nuevo Método para el Desarrollo de Sistemas de Información

Roberto Mendoza Padilla

Resumen

En el presente trabajo se describe un nuevo método para el desarrollo de sistemas de información el cual utiliza las herramientas de la Inteligencia Artificial, particularmente se enfoca el problema de los sistemas estáticos y la necesidad de la reprogramación, lo que repercute en un aumento de costos que mediante algunas alternativas propuestas aquí, pueden minimizarse.

Introducción

Esta y otras ponencias son el resultado de varios años de trabajo en relación a la Informática dentro del Departamento de Computación de la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) del IPN en el cual se ha conformado la Unidad de Investigación y Desarrollo en Computación (UIDC).

Es bien sabido por las personas relacionadas con la Informática, y aún por las que no lo están, que el proceso de desarrollo de los Sistemas de Información Administrativos tradicionales para cualquier empresa o institución ha seguido un proceso explosivo en los costos que implican, pasando de la época que los componentes mecánicos y electrónicos jugaban el principal factor económico, al punto en el que los costos de desarrollo y personal lo son, tal como lo muestra la figura 1.

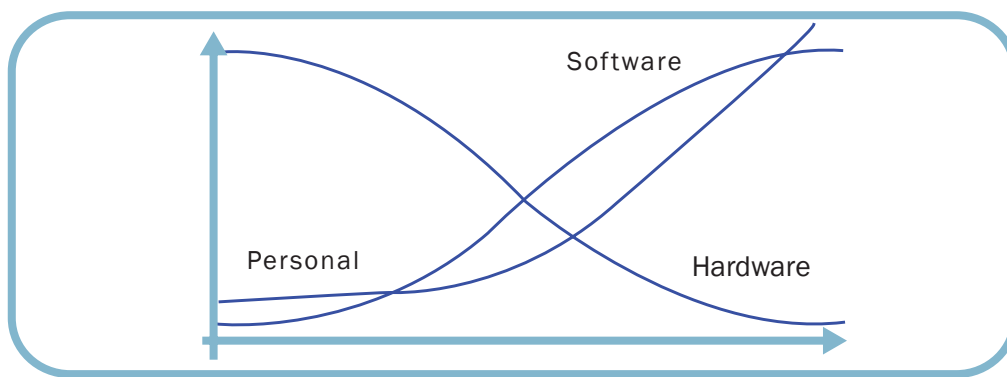


Figura 1. Desarrollo tradicional de sistemas.

Este problema es agravado en gran medida por la falta de técnicas apropiadas para el desarrollo de software, o bien, por el caso omiso que hace el personal involucrado en el área de las ya existentes, como por ejemplo la mala documentación o la ausencia de esta para los sistemas desarrollados o en desarrollo, lo que provoca retrasos considerables y un aumento excesivo de costos al estar combinado con la alta rotación de personal, tal como lo muestra la figura 2. En esta figura se tiene que las líneas verticales muestran puntos en los que hay cambio de personal, provocando rearranques de desarrollo por la falta de soporte documental.

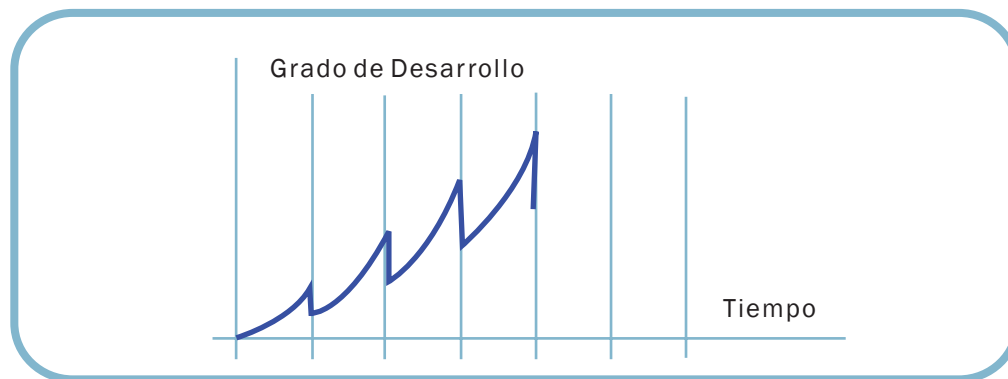


Figura 2. Desarrollo de sistemas sin soporte documental y con rotación de personal.

Un problema adicional sobre el cual se inician los primeros desarrollos es el que se refiere a la ausencia de un mecanismo que permita modificar al sistema de información sin un esfuerzo considerable, pues se llega a dar el caso en el que hay que rehacer gran parte del sistema en uso debido a los nuevos requerimientos del medio ambiente o deficiencias en su diseño.

Estas consideraciones acerca del Desarrollo de Sistemas, normalmente se combinan produciendo frustración entre los usuarios que pensaron que una computadora les solucionaría sus problemas en una forma mágica, tan solo por la gran inversión que hicieron al adquirirla.

Estos problemas y otros no mencionados provocan actualmente altos costos y/o una gran pérdida de tiempo, sin embargo, el presente estudio solo se enfocará al tópico que he denominado “Programación en Masa”, la cual es provocada por el enfoque actual de los Sistemas de Información, en el cual se dice implícitamente que los procesos o programas que lo forman deben ser el reflejo de su medio ambiente, es decir, cada rutina o módulo dará solución a una parte específica del problema, llegando posteriormente al punto de que cuando se modifiquen los requerimientos del medio ambiente, se debe hacer una reprogramación, debido a que este se encuentra reflejado en los programas, habiendo tantos como módulos o funciones tenga éste.

Además de lo anterior, este tipo de enfoque considera de alguna forma que su medio ambiente es estático, pues los programas funcionan en base a una fotografía tomada de la realidad en un momento determinado, permitiendo hacer adiciones, cambios o eliminaciones de funciones de acuerdo a la modularidad con que se haya desarrollado el sistema, pero aún en este caso, habrá necesidad de reconstruir el sistema, y en el peor de los casos, si de acuerdo a los nuevos

requerimientos del medio ambiente se tuvo que cambiar la descripción de un registro, se deberá rearmar cada uno de los programas para formar nuevamente el sistema.

Existe aún otro caso en el que la empresa o institución tiene nuevos requerimientos, y por la imposibilidad de poder cambiar el sistema, ya sea porque no hay alguien que lo haga o porque se compró hecho, se deberá tomar la decisión de adaptar su medio ambiente a las restricciones estáticas de los programas o bien de tener que reconstruir nuevamente el sistema, ahora con los nuevos requerimientos y tratando de no cometer los mismos errores de diseño, pero manteniendo esa filosofía fotográfica.

La alternativa que se presenta aquí pretende ser una herramienta de desarrollo con ayuda de la cual se elimine la incomoda relación, entre un medio ambiente dinámico y un sistema computarizado estático, evitando además la costosa Programación en Masa, logrando finalmente llegar a una gráfica de desarrollo tal como se muestra en la figura 3, obteniendo una drástica reducción en los costos de los sistemas, al mantener un software dinámico sin la necesidad de un gran cuerpo de analistas y programadores.

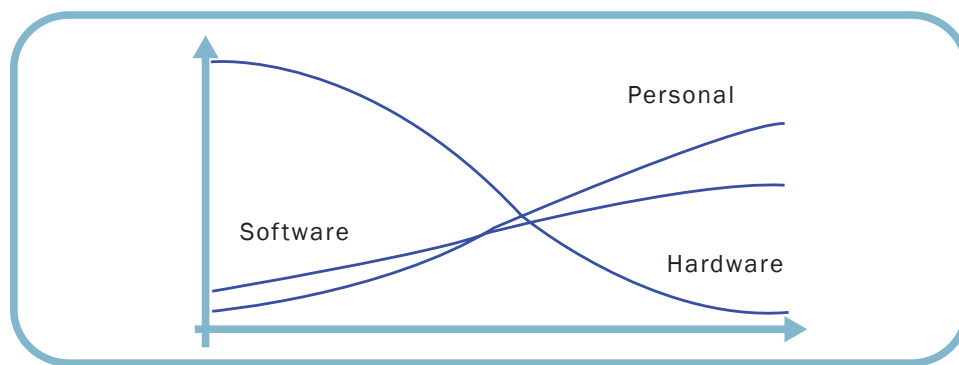


Figura 3. Desarrollo de sistemas “sin programación en masa”.

Esta forma de enfocar a los sistemas viene a cumplir históricamente con el papel que le corresponde, pues se ha visto que de acuerdo al diagrama de la figura 4 que representa a un sistema de información, al inicio lo más importante fue definir los procesos involucrados y las entradas y salidas que se obtenían o requerían para estos (figura 5), posteriormente, el enfoque se dirigió hacia una elaboración de procesos que explotaran las estructuras de datos obtenidas de las relaciones y transformaciones entre las entradas y las salidas (figura 6), finalmente, el enfoque dado aquí le da la principal importancia a las estructuras de control que representan a la arquitectura, medio ambiente o flujo de procesos y datos, a partir de la cual emanan las especificaciones y necesidades que deberá cubrir el sistema computarizado.

La estructura de control a desarrollar tiene la característica de ser independiente de los programas necesarios para emitir los reportes y demás información requerida por los usuarios, así de este manera cuando se deba modificar la representación del sistema, no será necesario modificar los programas que lo interpretan, debido a que estos estarán diseñados de tal forma para que tengan la habilidad de interpretar estructuras de control y flujos de información y no directamente las acciones realizadas en el sistema.

Para lograr lo anterior, es necesario pasar por dos fases de desarrollo, en la primera se dirigen los esfuerzos a representar las operaciones desarrolladas por los usuarios para lograr algún resultado, en una forma tal que no dependa esta de los programas que logran las acciones (figura 7), mientras que en la segunda fase, los esfuerzos se dirigirán a la explotación de los sistemas, mediante el planteamiento de problemas o requerimientos sobre los cuales se deben obtener respuestas concretas (figura 8).

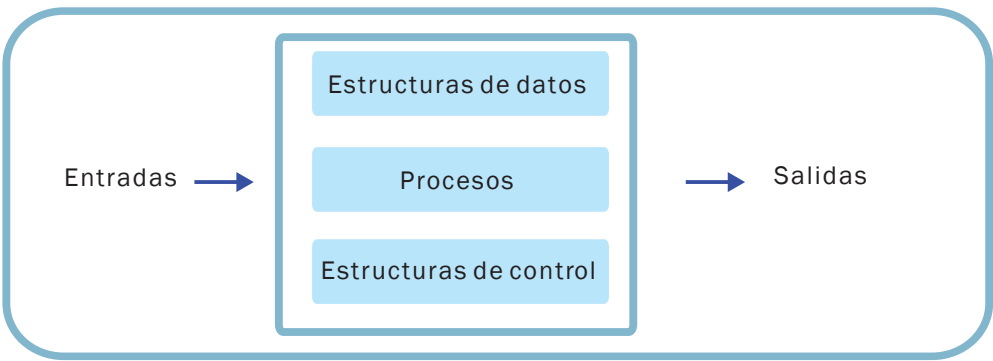


Figura 4. Arquitectura de un Sistema de Información.

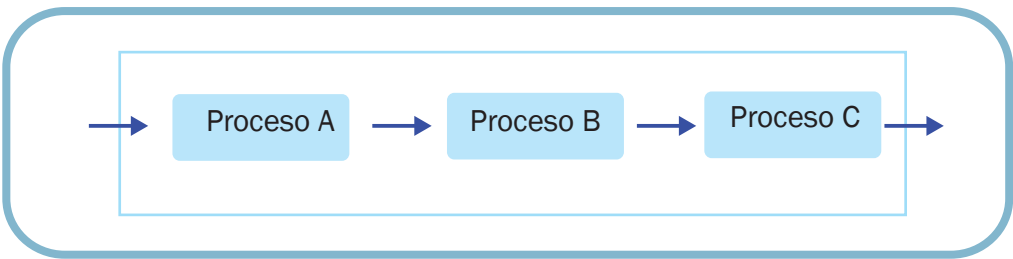


Figura 5. Enfoque inicial del desarrollo de sistemas.

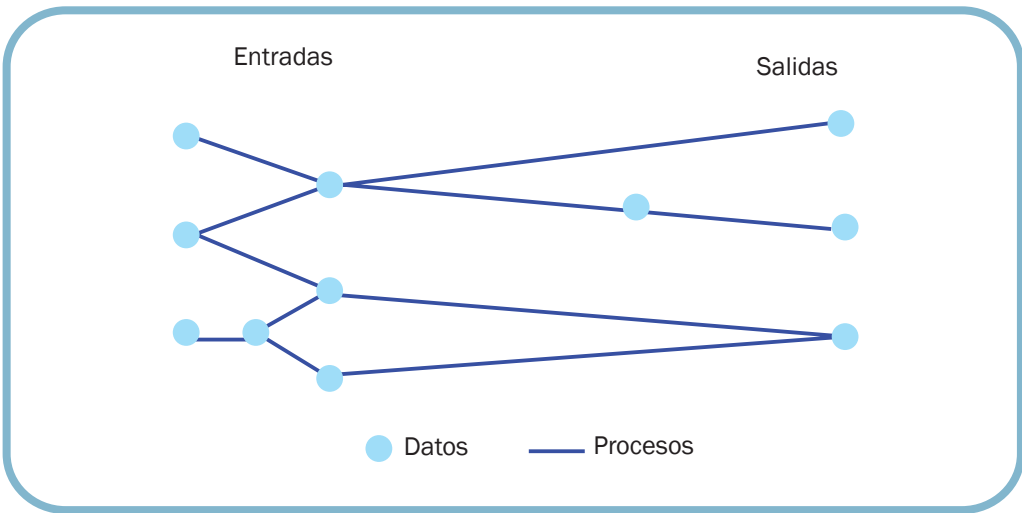


Figura 6. Enfoque actual del desarrollo de sistemas.

Tanto el diseñador como el explotador mostrados en las figuras 7 y 8, pueden ser personas o sistemas, dotados con las habilidades necesarias para poder interactuar con el usuario con el fin de obtener una clara definición del sistema a representar y para dar las soluciones adecuadas a los problemas.

En el caso de tener como responsable del Diseño y Explotación a una persona, tendrá que cumplir con el esquema de la figura 9, y en el caso de que el Diseñador sea un sistema, este deberá estar constituido de tal manera para que pueda proponer un modelo en base a un diálogo con el usuario, utilizando la técnica de prototipos, es decir, debido a la imposibilidad de efectuar una observación, el único medio de que dispondría como fuente de información sería el usuario, con lo que se limitaría a una serie de preguntas-respuestas de donde se obtendría una propuesta (prototipo) del modelo de la organización, en base al cual se harían más preguntas-respuestas hasta llegar a la mayor refinación posible, estableciendo así un ciclo de retroalimentación tal como lo muestra la figura 10.

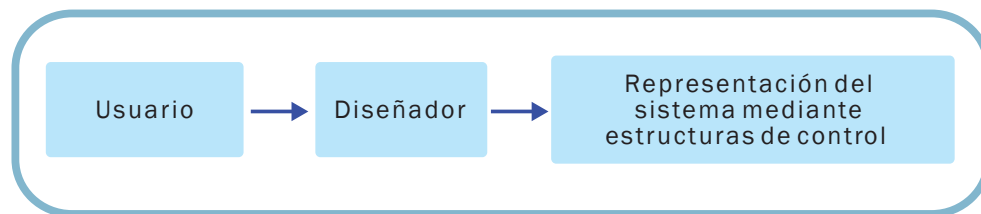


Figura 7. Enfoque propuesto del desarrollo de sistemas.

El diálogo necesario para que se obtenga el modelo, se puede hacer en base a un cuestionario prefabricado sobre el cual se obtendrían respuestas, seleccionando solo las que sean necesarias hacer.

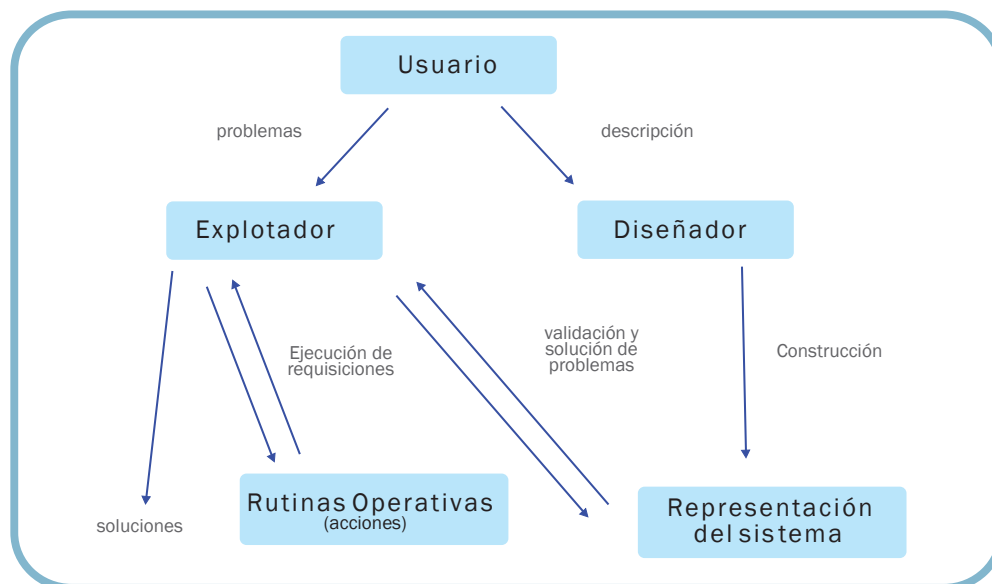


Figura 8. Arquitectura para la definición y explotación de un sistema de información.

En el caso de que se tenga a un sistema para desarrollar la propuesta, deberá tener las herramientas (idealmente) para comunicarse con el usuario en Lenguaje Natural, aunque lo más óptimo sería que este describiera su sistema y/o problema en una forma gráfica sobre una pantalla, utilizando las herramientas del CAD/CAM, esquema a partir del cual se haría la descripción interna de la requisición del usuario.

Después que se ha hecho un análisis de los requerimientos del usuario, en lo que debemos pensar es en la forma de representar internamente esa información de tal manera que refleje la estructura organizacional o la estructura de control del problema, de tal manera que se pueda hacer un programa de computadora que la analice y ejecute las acciones requeridas para dar un resultado de tal manera que si cambiáramos la estructura, el programa no tuviera que modificarse.

La representación de esta estructura se puede llevar a la práctica utilizando diversas herramientas, las cuales estén basadas en el análisis de nodos y líneas de enlace o relaciones entre estos, las cuales denotarían una transformación de un dato o nodo de un elemento en otro. Este tipo de filosofía se puede encontrar en la teoría de las gramáticas, lenguajes formales, bases de conocimiento y otros en los que existan entidades y relaciones entre estas. Para hacer más claros los conceptos aquí manejados, se muestra un ejemplo de su aplicación en las figuras 11 a 13.

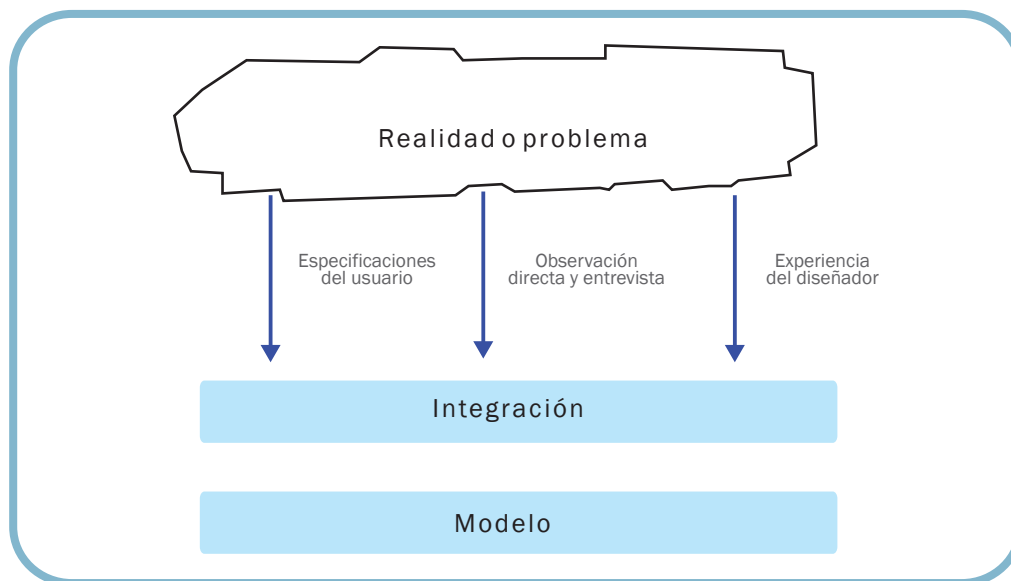


Figura 9. Diseño de sistemas por una persona.

Conclusiones

Habiendo analizado brevemente las desventajas que representa el desarrollo tradicional de los sistemas de información, podemos ver que la alternativa presentada aquí, si bien no desarrollada hasta sus últimas consecuencias, ofrece una opción para evitar un gran desperdicio de esfuerzos y recursos por parte de las empresas, que si bien hay algunas que están saliendo adelante a pesar de las crisis por las que hemos atravesado, la mayoría tienen problemas económicos, viendo a los sistemas de computo como algo que les redituara solo gastos o problemas

por los grandes volúmenes de recursos a asignarle y que prefieren enfocarlos a otras áreas que les permitan seguir subsistiendo.

Es obvio además que algunos de los requerimientos planteados como parte del enfoque (p. ej. manejo de lenguaje natural) está siendo apenas desarrollado en México y solo por algunas entidades de investigación, sin embargo se tiene la firme convicción de que los esfuerzos deberían estar encaminados en mayor grado a este tipo de desarrollos y no a seguir desarrollando los sistemas en una forma más eficiente, pero siempre con la misma mentalidad.

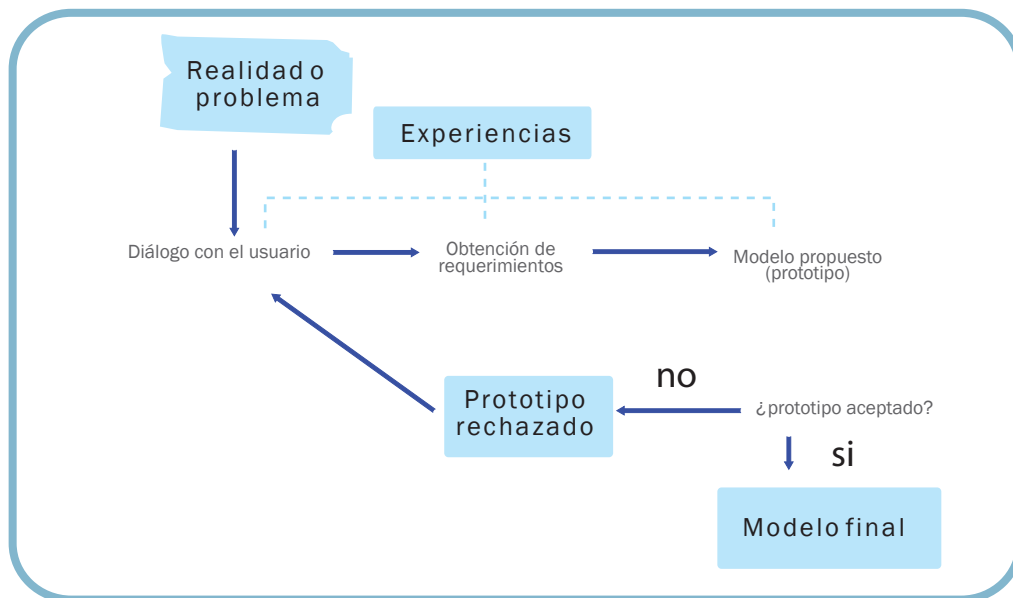


Figura 10. Diseño de sistema asistido por computadora.

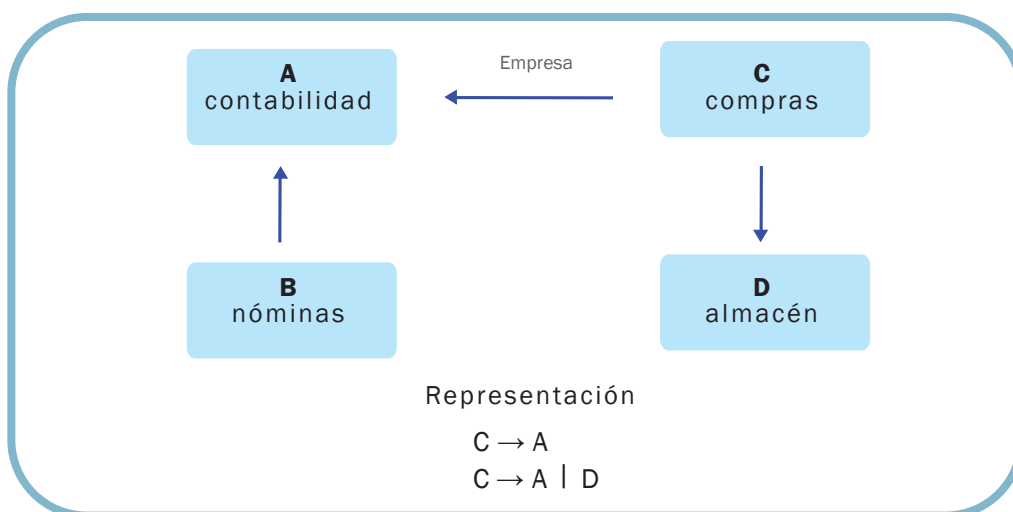


Figura 11. Estructura organizacional y su representación equivalente.

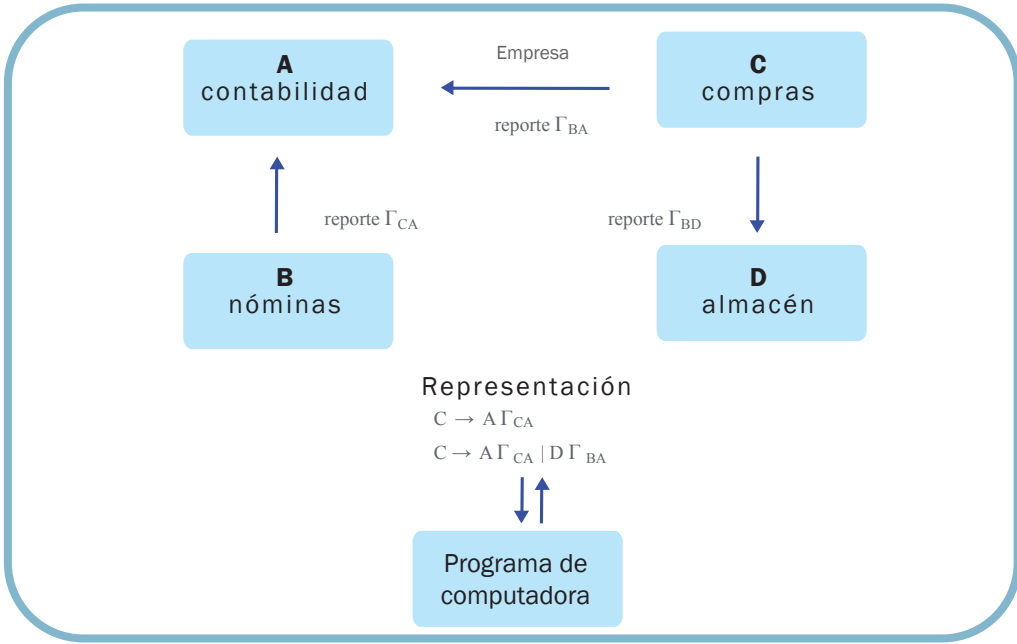


Figura 12. Estructura organizacional con ejecución de acciones.

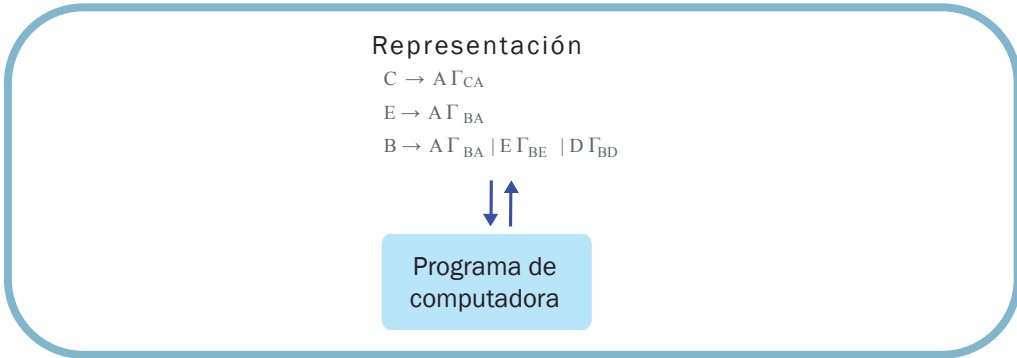


Figura 13. Al cambiar las requisiciones del medio ambiente, el programa es el mismo.

I.6 La Comunicación Hombre-Maquina utilizando Lenguaje Restringido

Juan Martín González Vázquez⁵

Resumen

Se describe un modelo para el procesamiento del lenguaje natural, el cual tiene como módulos principales un decodificador/encodificador, un validador de completez, un estructurador de respuestas y un ejecutor.

Introducción

La tecnología actual para la fabricación de computadoras y las características de estas rebasan cualquier expectativa planteada en sus anales, su aplicabilidad ya es comprobada en la gran mayoría de nuestras actividades, ahora bien si es tan útil e importante su uso, como podemos obtener el mayor provecho de ella, sin necesidad de recurrir a un experto en su uso o a algún tipo de educación especializada para ser uno de ellos, ya que pese a los esfuerzos hechos a la fecha es necesario definir procedimientos complejos y traducirlos a un lenguaje que sea manejado por la marca y versión de computadora que queremos utilizar, o adaptarnos a los esquemas de productos ya hechos para aplicaciones mas o menos parecidas a las nuestras, y sin omitir la posibilidad de que no existan tales productos para nuestra aplicación específica.

Dado lo anterior caemos en la cuenta que una de las principales barreras para el uso de una computadora, es precisamente la comunicación con ella, de donde surge la necesidad de dispositivos que hagan esta comunicación sencilla y eficaz y en donde no sea necesario más que nuestro conocimiento en el área, la problemática que nos afecta y una computadora.

Aparentemente lo expuesto es una imagen de un usuario que su único problema es que necesita información, y no tiene los recursos de tiempo y/o conocimientos en el manejo de una computadora para obtenerla. Sin embargo no es el único tipo de usuario, ya que la vida profesional de algunas áreas en la calle o en el hogar existen actividades que son fácilmente sistematizables

⁵ Juan Martín González Vázquez elaboró este trabajo en abril de 1985, siendo subdirector de informática en la Secretaria de Programación y Presupuesto, en colaboración con el Instituto Nacional de Estadística, Geografía e Informática en la Dirección General de Política Informática en la Dirección de Desarrollo Informático, Proyecto Censo Agropecuario, Instituto Politécnico Nacional Unidad Profesional Interdisciplinaria de Ingeniería Ciencias Sociales y Administrativas, División de Ingeniería Unidad de Investigación y Desarrollo del Departamento de Computación.

y susceptibles de ser automatizadas por lo que necesitaríamos mas que información, acciones y ejemplos claros. Los tenemos en armadoras de autos, robots manipuladores de substancias peligrosas, controladores de tráfico, cajeros automáticos, cajas registradoras, conmutadores telefónicos, juegos infantiles, automóviles, etc., los cuales muchas veces requieren recibir sus procedimientos por medios altamente sofisticados y nuevamente nos encontramos con el problema de comunicación.

A lo que surge un sentimiento general de necesidad de un dispositivo de comunicación eficaz, y se discute sobre la conveniencia de que consista en una interface en el idioma de la persona que lo use y que sea capaz de hacer que la computadora nos entienda de una manera sencilla y eficaz, aunque a su vez surge otra corriente que manifiesta que para manejar los problemas de su área no le es funcional manejar el idioma sino la jerga con todo y simbología de su área común, como podría ser el caso de un matemático o un químico, etc.; los propios informáticos cooperan en esta discusión pidiendo un lenguaje altamente taquigráfico para sus conceptos de manejo de la computadora e información.

Por lo que es necesario establecer las reales características del lenguaje natural, las cuales tienen el requisito primordial de ser generales y soportar los requerimientos de todos sus posibles usuarios, ya que a todos les asiste la razón al declarar “natural” su pretendida forma de comunicación con la computadora por lo que se definen las siguientes características:

- 1) Debe ser *independiente* de un idioma determinado; lo que significa que la formación de enunciados tendrá las reglas, signos y criterios de consideración para la unidad léxica que determine la persona que use el sistema y este debe ser capaz de digerir los aspectos antes mencionados para el idioma en cuestión y aplicarlos a futuros contactos.
- 2) Ser capaz de *codificar mensajes propios del sistema* en el idioma que ha aprendido.
- 3) Debe ser capaz de determinar la *validez del mensaje* en función del contexto en el que se da, el idioma definido y los conocimientos proporcionados al sistema.
- 4) Debe ser capaz de *estructurar procedimientos propios y/o enseñados* además de poder inferir posibles procedimientos complementarios, para la total ejecución del mensaje y también supervisar que no viole ninguna restricción que por medio de la instrucción o la experiencia posea.
- 5) También será capaz de *determinar todos los recursos de la computadora* necesarios para procesar el mensaje, administrarlos y por último informar al emisor del mensaje, el resultado de tal ejecución en el idioma propio del usuario.

Podemos concluir que con lo antes expuesto el manejar “Lenguaje Natural” es algo más que el manejo de la traducción de un idioma a otro y que lleva involucrado desde las características más involucradas con la computadora hasta conceptos tan humanos como experiencia, enseñanza, aprendizaje, etc. y no obstante la presente lo declara como restringido y la razón es que los motivos de la comunicación no abarcan conceptos como sentimiento, instintos, corazonadas y cuestiones difíciles de cuantificar y de representar como un estado en el computador, tales como alegría, tristeza, temor, amor, etc. y se reduce el esquema a obedecer y manejar las propias habilidades en función de la satisfacción de un requerimiento. No obstante no se niega la posibilidad de crear el concepto, representación y respuesta de los ejemplos anteriores.

II Manejador de Lenguaje Natural Restringido

Ahora bien para la temática de lenguaje natural restringido planteamos el siguiente modelo al que nominalmente lo constituyen las siguientes partes:

- 1) *Decodificador/Encodificador*: Se encarga de traducir el mensaje fuente a una notación estándar del sistema que sea procesable por las otras unidades que lo constituyen, además de procesar los mensajes del sistema y trasladarlos al idioma del usuario.
- 2) *Validador de Completez*: Determina si la estructura decodificada tiene algún sentido basado en la que se ha aprendido y que la acción a la que apunta no esta censurada por alguna restricción aprendida o esta limitada por algún concepto que invalide al usuario como ejecutor de un proceso, así, al mismo tiempo que si es necesaria más información para darle el curso adecuado al mensaje la solicite.
- 3) *Estructurador de Respuestas*: Se encarga de subdividir la acción genérica en subacciones más específicas que estén en coordinación con lo que puede hacer la computadora, verificar los procedimientos necesarios para cumplirlos y en caso de que no existan declarados verificar si los puede inferir de los que si lo están. Así como declarar las variables necesarias para el paso de información entre procedimientos. Determinar nuevamente si uno de los procedimientos incorporados no choca con las restricciones conocidas y finalmente generar los requerimientos de información necesarios así como de recursos materiales para ejecutar el procedimiento.

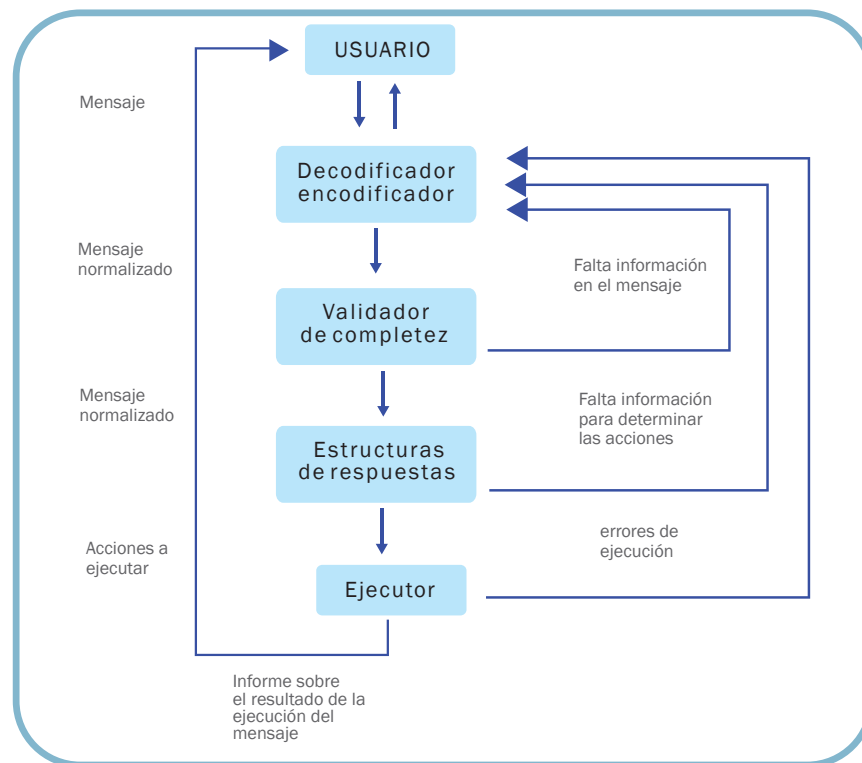


Figura 1. Manejador de Lenguaje Natural Restringido.

4) *Ejecutor*: Se encarga de verificar que los requerimientos estén disponibles y en la cantidad deseada y que estas cantidades tampoco infrinjan las restricciones registradas. Una vez que tenga todo dispuesto debe ejecutar cuando así sea necesario los módulos pertinentes y/o generar los programas, corridas que se requieran para lograr la respuesta pedida, monitorear la ejecución y estructurar un informe del resultado para pasárselo al Decodificador/Encodificador y este a su vez al usuario.

1 Decodificador/Encodificador

Lo Constituyen las siguientes partes:

- 1) *Analizador Léxico*: En primera instancia necesita que se le proporcione los caracteres que puede considerar como validos no obstante cuando aparezca un carácter desconocido pedirá autorización para ponerlos en su acervo de caracteres validos, otra cosa que también pide son los criterios para separar una unidad léxica de otra, las cuales se almacenan en un lugar expreso para ello. Independientemente que tendría las especificaciones para procesar español.
- 2) *Categorizador y Validador*: En función a una gramática. Esta parte no obstante de tener lo necesario para procesar español orientado a proceso de datos es capaz de “Aprender” una nueva gramática en función a ejemplos; la cual también se almacena y enriquece y en sí esta parte tiene como salida las unidades que separa el analizador léxico validadas en su estructura y categorizadas genéricamente por lo que representan en la gramática.
- 3) *Encodificador*: Funciona en forma inversa al categorizador, es decir parte de una serie de unidades separadas y categorizadas, las cuales al seguir la gramática las estructura de acuerdo a esta en un mensaje en el idioma que representa la gramática.
- 4) *Formateador*: Lo único que hace es centrar mensajes y manejar la pantalla de manera de lograr cierta estética dentro del mensaje desplegado a parte de generar el medio ambiente para grabar a disco, cinta, o papel dichos mensajes.

2 Validador de Completez

Sus partes principales son:

- 1) *Resolver de indefiniciones e integrador de significado*: Es resolver de indefiniciones porque en su momento, ciertas unidades léxicas que no haya podido categorizar en el paso anterior, en esta parte las integra a la base de conocimiento por lo que pide al usuario información sobre las relaciones a establecer en la base de conocimientos con otras unidades así como características que la ubiquen en un contexto determinado y/o limitaciones que afecten de alguna forma su operación bajo alguna situación especial y en su momento posibles valores por ausencia. La función del integrador de significado consiste en integrar el mensaje original todas aquellas particularidades de las unidades que influyen o se vean influenciadas por la operación de la unidad en cuestión así como las reglas o relaciones que haya que considerar para su secuenciación.

2) *Validación del significado contra el contexto*: Es la consideración global de todas las unidades ensambladas de acuerdo al significado e integradas como las nuevas características generales del mensaje que se contrastan contra los elementos de la base de restricciones y se determina la validez del mensaje en esta etapa en función de las violaciones a las restricciones, además de que se pide en caso de ambigüedad en las partes integradas de significado, información para hacer la selección adecuada.

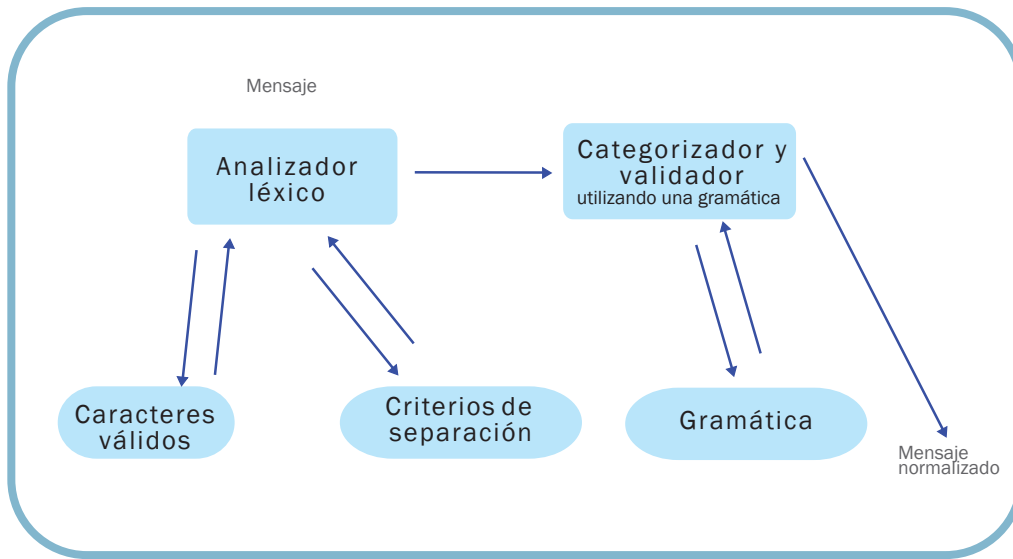


Figura 2. Decodificador/Encodificador.

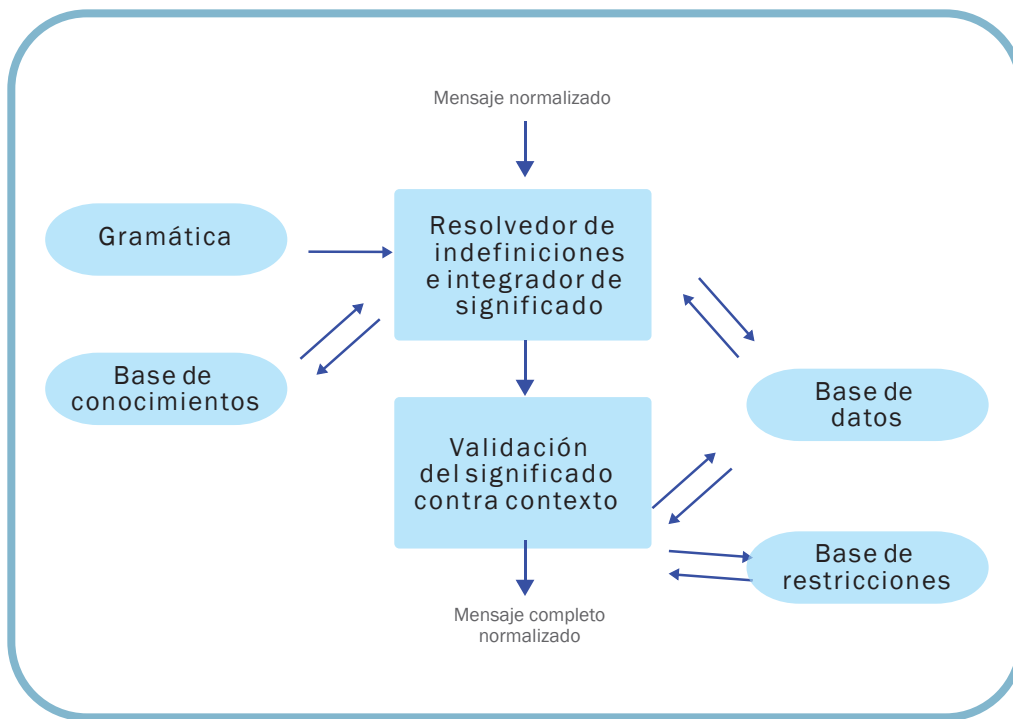


Figura 3. Validador de Completez.

3 Estructurador de Respuestas

Tiene cuatro módulos que completan su función y que son:

- 1) *Determinador de los procedimientos a utilizar*: Esta parte integra los procedimientos que proporcionen las salidas deseadas para cada una de las unidades definidas por el significado y el contexto y en caso de que una unidad no tenga procedimientos declarados debe de tratar de integrar procedimientos que no obstante estén declarados para otras unidades, logren satisfacer las salidas necesarias o en su momento solicitar expresamente la definición de un proceso en particular.
- 2) *Discretizador de procedimientos*: Esta parte se encarga de declarar las variables y/o constantes en los procedimientos de tal forma que todos tengan una secuencia de entradas y salidas validas para la consecución de la salida total, pero en caso de que alguno de los requerimientos tipo variable o constante no pueda ser ubicado en el procedimiento anterior o posterior para completar una entrada o salida determinada se requerirá su definición al usuario a fin de que se integre a un determinado procedimiento como salida además de que se declare el lugar de donde se puede obtener.
- 3) *Determinador de la validez del procedimiento*: En esta etapa se verifica que los accesos, entradas y salidas en función a la operación y contenido no violen ninguna restricción declarada para un procedimiento determinado, o un usuario.
- 4) *Generador de Requerimientos de ejecución*: Una vez que se han ubicado las fuentes que dan información a los procedimientos y los lugares en donde esta se plasma se procede a hacer una lista con todos los requerimientos lógicos y físicos, entendiendo por esto los lugares y formatos necesarios para los datos así como los dispositivos asociados a los datos de entrada y salida y algunos otros insumos necesarios para recuperar la información y/o plasmarla tanto en calidad como en cantidad y si en algún momento cierto recurso es necesario, pero no esta definido se procede a definirlo, verificando que no choque su definición con alguna restricción declarada.

4 Ejecutor

Esta componente del modelo tiene las siguientes partes:

- 1) *Generador de flujo logístico*: Se encarga de establecer los controles de documentación de versiones, actualización y requerimiento de equipo y administrar de alguna forma la concurrencia en el requerimiento de recursos y/o tiempo de máquina y tramitar la mejor operación de los procesos generados por los mensajes del usuario y regidos por las restricciones declaradas por la física de la máquina y/o restricciones a causa del tipo de usuario o procedimiento.
- 2) *Generador de programas, corridas de ejecución y/o llamado a módulos generales*: Una vez que se tiene lo que se va a hacer y con que se va a hacer, es necesario generar o transformar los procedimientos en algún lenguaje comprensible por la máquina, o en algún llamado a uno de los módulos generalizados del sistema, así como generar también los postulados de control que logren que la máquina tenga conciencia real de lo que

necesitamos para la ejecución y finalmente poner en ejecución el proceso recolectando toda la información posible para dar un informe final del resultado de este, pero con la característica de auxiliarse de la base de conocimiento y la gramática para estructurar el informe en forma de unidades léxicas separadas y categorizadas para entregarlas al encodificador/decodificador para que lo transmita al usuario.

3) Estructurador del informe de resultado de la ejecución.

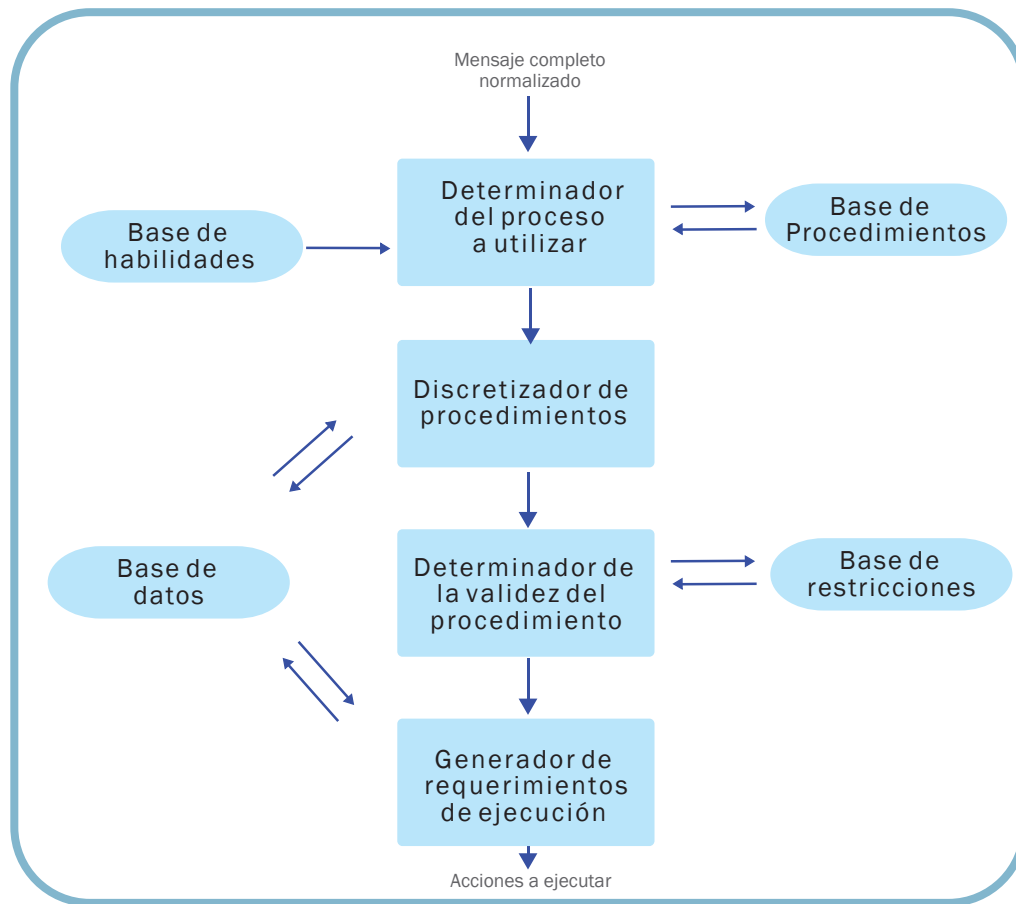


Figura 5. Ejecutor.

Cabe aclarar que en todas las etapas del sistema en general se considera la posibilidad del aprendizaje y un módulo general de “olvido” que evita el crecimiento inmoderado de los datos acumulados con objeto transitorio, además de que los programas, y lenguaje de control generado se borran al termino de la corrida.

Conclusiones

Se presentó un modelo para el procesamiento del Lenguaje Natural Restringido. Es innegable que pese a la complejidad descrita en el modelo, su ámbito es aún restringido aunque su utilidad es grande no obstante el esquema es fácilmente adaptable cuando necesitamos que las

salidas sean acciones ya que en lugar de información como salida serán señales para la operación de mecanismos subordinados a la computadora y se le añade que a cada paso de ejecución solicite información que se enruta como mensaje y cambia si es necesario el curso de acción del procedimiento en general.

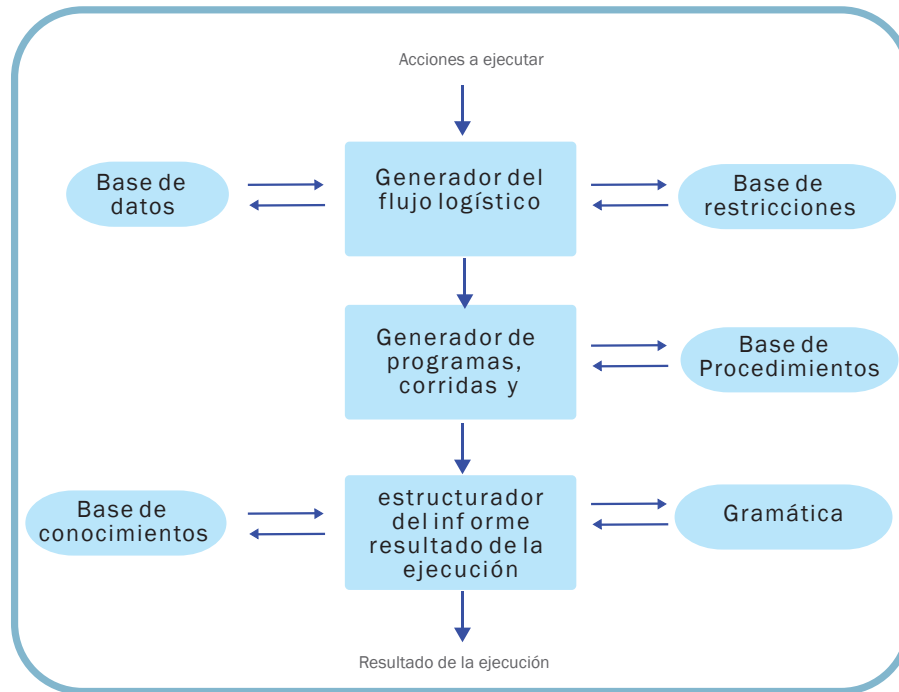


Figura 5. Ejecutor.

Bibliografía

- [1] Galindo Soria, Fernando, "Notas del Curso, Lenguajes Formales", UPIICSA, 1981.
- [2] Galindo Soria, Fernando, "Notas del Curso, Compiladores", UPIICSA, 1982.
- [3] Galindo Soria, Fernando, "Notas del Curso Sistemas Operativos I", UPIICSA, 1983.
- [4] Galindo Soria, Fernando, "Notas del Curso Sistemas Operativos II", UPIICSA, 1984.
- [5] Galindo Soria, Fernando, "Notas del Seminario de Sistemas de Bases de Datos", SPP UPIICSA, 1984.
- [6] Donovan, "Sistemas Operativos", ed. Diana, 1979.
- [7] F. de Saussure, "Curso de Lingüística General", ed. Losada, 1945.
- [8] Chomsky, Noam, "Estructuras Sintácticas", de. Siglo XXI, México, 1974.
- [9] Chomsky, Noam, "Aspectos de la Teoría de la Sintaxis", ed. Siglo XXI, México 1974.
- [10] Greene, Judith, "Psicolingüística", ed. Trillas, México, 1980.
- [11] González Vázquez, Juan Martín, "Diseño de un Lenguaje de Control Icónico", UPIICSA, México, 1984.
- [12] González Vázquez, Juan Martín, "Aspectos Elementales del Lenguaje Natural", UPIICSA, México, 1982.

I.7 SI-VE: Sistema de Visión Experto⁶

Ma. Sandra Camacho V., G. Patricia Gómez R., Jesús M. Olivares C.

Resumen

Este trabajo describe una herramienta para el reconocimiento de imágenes usando entre otros el método de rastreo mediante la técnica de primeros vecinos para obtener el número de forma. Los patrones identificados se ubican y se relacionan para describir la escena completa.

Introducción

Desde el descubrimiento de los metales hasta la fabricación en serie de computadoras sofisticadas de hoy en día, el hombre ha sabido mejorar y modificar su medio ambiente desarrollando investigaciones para diversas aplicaciones. Es así como el hombre ha evolucionado hasta llegar a los avances de los descubrimientos que actualmente son aplicados en todas las ramas de la ciencia. Una de esas ciencias es la computación, en la que se ha venido escudriñando últimamente, llegando a hacer uso de todo lo que se tiene para integrar herramientas muy sofisticadas y poderosas, teniendo entre ellas a la Inteligencia Artificial (IA).

Si bien es cierto que la IA ha tenido gran auge, también es cierto que son muchas y muy variadas sus aplicaciones, tal es el caso de la robótica, los sistemas expertos y el reconocimiento de patrones.

Se pretende pues, seguir buscando materia y explotando el que ya se tiene para llegar a formar herramientas más adecuadas a las necesidades planteadas por el hombre. Uno de los temas de mayor aplicación dentro de la IA es el de “Visión por Computadora”, el cual se enfoca entre otras aplicaciones, a la percepción visual de los robots para su uso industrial.

El problema de desarrollar sistemas de Visión por Computadora se puede dividir en los problemas de adquisición, tratamiento e interpretación de una imagen.

⁶ Este trabajo lo realizaron como parte del curso de Lenguajes Formales en el 5º semestre de la Licenciatura en Ciencias de la Informática en la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas del Instituto Politécnico Nacional (UPIICSA-IPN), concluyéndolo en julio de 1986 y reportado en agosto del mismo año. El Sistema de Visión Experto funciona en una computadora HP-3000 con sistema operativo MPE.

- 1) La *adquisición* utiliza dispositivos tales como cámaras de televisión, láser, ultrasonido, infrarrojos, etc. y de un convertidor analógico-digital para captar y digitalizar la imagen.
- 2) El *tratamiento* incluye la preparación y mejora de la información adquirida (filtrado, ecualización del histograma, reconstrucción, etc.) y la segmentación; es decir la partición de la imagen en zonas con propiedades similares.
- 3) La etapa de *interpretación* de una imagen es la más compleja, pues de ella depende el éxito de la interpretación de la escena. Consta de dos procesos principales: extracción de características o primitivas y el análisis y manipulación de las mismas.

Uno de los trabajos realizados en el campo de visión por computadora, que se enfoca a la tercera etapa (interpretación de la imagen) trabajando con figuras bidimensionales es el Sistema de Visión Experto “SI-VE” cuyo nombre se debe a que posee la capacidad de aprender nuevos patrones.

SI-VE es el sistema que se analiza en este artículo con el propósito de proporcionar los métodos desarrollados para resolver el problema de la interpretación, llamada también “Reconocimiento de Imágenes”.

I El Problema de Visión por Computadora

La robótica trabaja continuamente en lo que es la percepción visual por computadora y procesamiento de imágenes, por lo que han sido desarrolladas numerosas investigaciones al respecto.

Así pues, la visión artificial es un tema causante de varias controversias y estudios que implican el reconocimiento de imágenes así como la ubicación de tales imágenes o figuras que conforman una escena dada.

Aunque resulte familiar el término “Visión por Computadora”, no es igualmente familiar el método y los medios utilizados para cubrir los requisitos necesarios para ello. Esto no implica que sea algo imposible de realizar, pero sí lo bastante abstracto y complicado como para tratarse con la delicadeza y cuidado necesario al desarrollar estudios minuciosos que conlleven a una solución exacta y general que indique el camino a seguir para cualquiera que sea su aplicación por muy específica que ésta sea.

A medida que se profundice en el tema con investigaciones, estudios y experimentos comprobados y puestos en práctica, se proporcionan, a su vez, nuevas herramientas útiles de las cuales partir para llegar a la construcción, de lo que se tiene como máxima pretensión en el área de “Robots Inteligentes” capaces de cubrir cualquier tarea que se les encomiende.

La extracción de características es la primera y más importante etapa de SI-VE, de la cual depende la eficiencia del sistema, la cual pretende llegar a la representación estructural única y exacta de cada figura.

El análisis de la representación estructural elabora un modelo base en forma de red, el cual contiene información tal como distancias y ángulos. Dicho modelo se encuentra listo para ser comparado con los patrones.

La descripción de la escena, que es la última etapa del sistema, establece la relación de todos los elementos que componen la escena con respecto a su universo y la relación existente entre ellas mismas dentro de tal universo.

El crear un método exacto, rápido y confiable para la identificación y descripción de escenas es el objetivo principal que motivó la realización del Sistema de Visión Experto.

II Sistema de Visión Experto para el Reconocimiento de Escenas

La obtención de características de los contornos de las figuras que conforman una imagen, es la pauta que sirve de partida hacia el reconocimiento de escenas compuestas de una o más figuras.

Una vez conocida la problemática involucrada en el área, se tiene que los pasos o puntos principales a cubrir para llegar a una efectiva obtención de resultados en reconocimiento de imágenes es la siguiente:

- 1) *Aceptación de la escena*: Se captura la imagen que contiene la información a procesar a través de una matriz de $M \times N$ caracteres en la que se vacía dicha información. La imagen puede estar formada por una o más figuras de tipo lineal, geométricas, compuestas o en forma de estrella. La digitalización de la escena, que será la forma de capturarla puede ser por pantalla o por editor.
- 2) *Detección del número de figuras que conforman la escena*: Como resultado de un primer seguimiento o búsqueda general de la escena captada, se encuentra el número de figuras que conforman la imagen, distinguiendo cada una de ellas de acuerdo al carácter de identificación asignado para su reconocimiento.
- 3) *Rastreo de figuras detectadas*: Una vez encontradas las figuras integrantes de la escena, se procede a hacer el rastreo por figura consistente en ir siguiendo punto por punto el contorno de la figura por medio del método de primeros vecinos, el cual se realiza siguiendo las localidades de la matriz que han sido, ocupadas por la figura, partiendo del punto inicial.
- 4) *Análisis de figuras*: A partir del rastreo efectuado para cada figura, se obtiene el modelo correspondiente para cotejar con lo almacenado como patrón.
- 5) *Descripción de la escena*: La descripción de la escena es la parte culminante del proceso, la cual consiste en determinar la relación y posición que guarda cada figura componente de la escena.

Cada uno de los pasos mencionados involucran el uso de varias técnicas y métodos de Inteligencia Artificial y Reconocimiento de Patrones.

En general el proceso seguido para llegar al objetivo final es el que se observa en la figura 1.

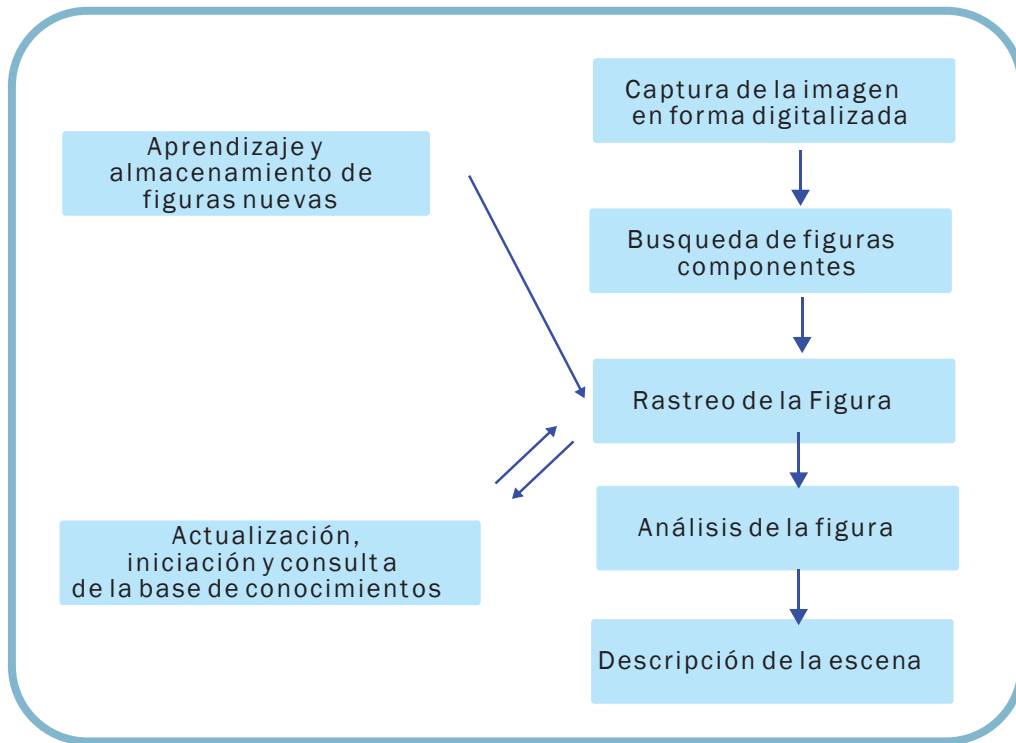


Figura 1. Etapas del método del sistema de visión por computadora.

El aprendizaje es utilizado cuando la figura analizada no se encuentra en la base de conocimientos, en cuyo caso se puede o no adoptarla como patrón de acuerdo a la decisión del usuario.

La etapa de actualización del sistema permite dar de baja, modificar o consultar los patrones. Una particularidad del sistema es que en principio sólo puede distinguir a las figuras dependiendo de la siguiente clasificación:

- 1) *Punto.*
- 2) *Líneas.*
- 3) *Figuras en forma de estrella.*
- 4) *Figuras compuestas.*

Al analizar figuras que no reconoce, realiza un aprendizaje guiado por la persona que maneja el sistema. La arquitectura modular del SI-VE se muestra en la figura 2.

III Método de Rastreo

Consiste en seguir una figura punto por punto, utilizando el método de primeros vecinos, adaptado especialmente para detectar vértices, puntos con bifurcaciones y puntos finales. Se llaman vecinos de un punto a todos los puntos que lo rodean. Estos vecinos tienen una

jerarquía que determina la dirección a seguir. El valor dado a cada dirección se puede observar en la figura 3.

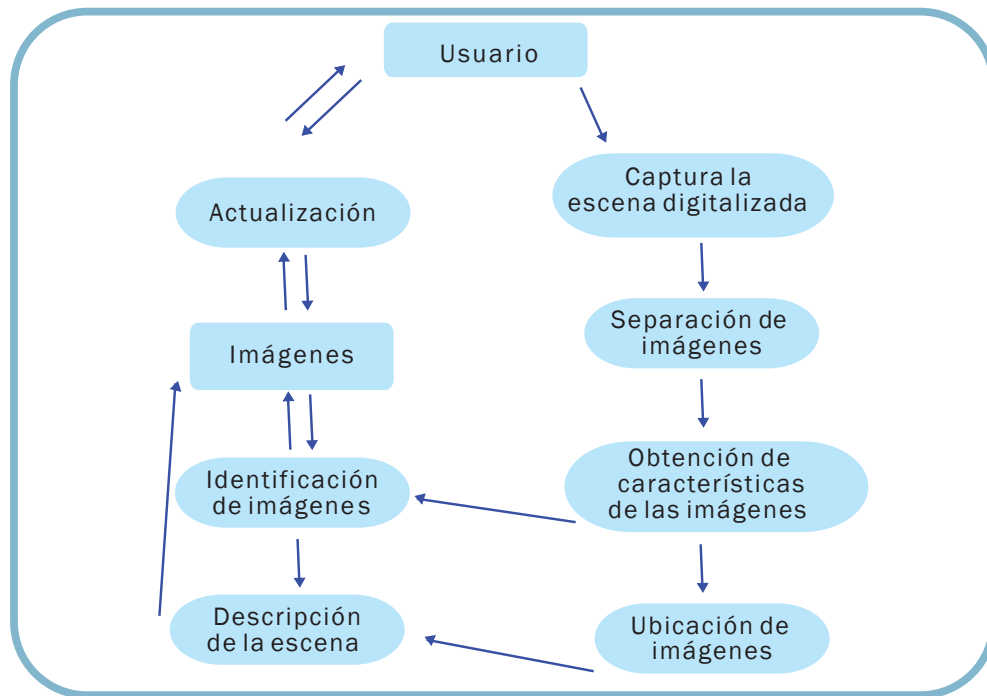


Figura 2. Arquitectura del SI-VE.

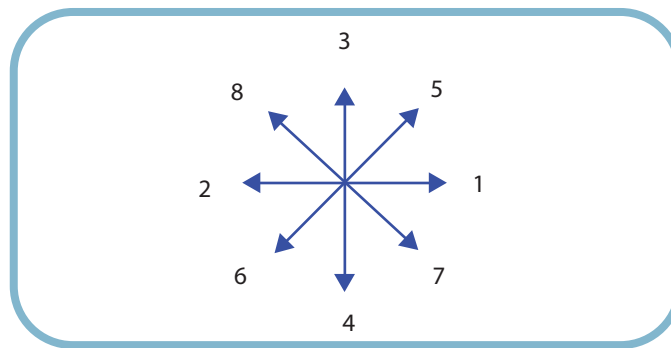


Figura 3. Direcciones de primeros vecinos.

En cada avance se almacena la dirección respectiva para formar lo que se conoce como número de forma, que representa a la figura, con el cual, posteriormente, se analiza la inclinación de las rectas de la figura en análisis.

Cuando hay un cambio de dirección se toma el punto de análisis como fin de recta, e inicio de una nueva en el caso de que haya sólo dos vecinos. Cuando hay tres o cuatro vecinos se realiza un análisis para cada uno de ellos, que determine la pendiente y con ello el punto que realmente deba ser tomado como inicio de esta nueva recta, almacenándolo para su posterior seguimiento. Véanse las figuras 4a y 4b.

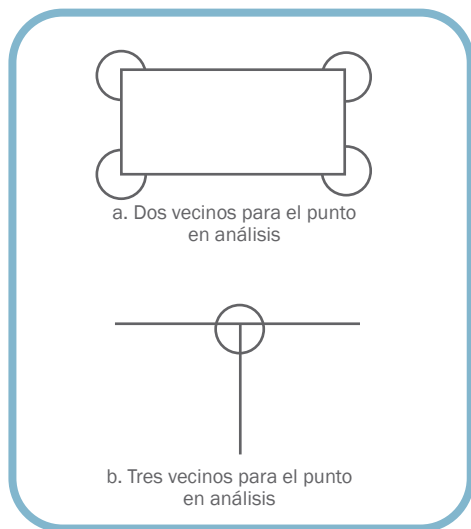


Figura 4. Cambios de dirección.

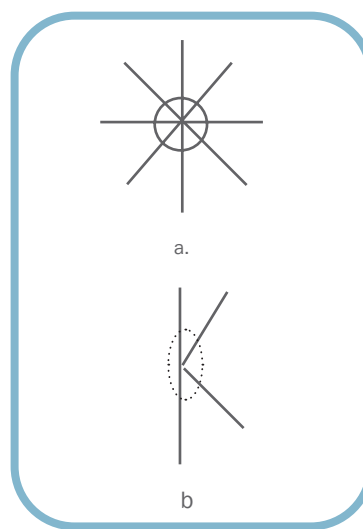


Fig 5. Cambios de dirección.

En el caso de que un punto cuente con más de cuatro vecinos, se analizará el siguiente punto hasta llegar a aquel donde convergen las rectas, dado que cada vecino representa el inicio de una recta. Tal es el caso de las figuras 5a y 5b.

IV Identificación de Imágenes

El proceso de identificación se realiza una vez que fueron extraídos los datos de la figuras, los cuales son:

- 1) *Los puntos que son terminales de una recta, vértices o intersección entre rectas.*
- 2) *Las rectas que tiene la figura.*

Lo anterior se almacena en una estructura que evita la repetición de puntos y rectas (base de datos), con los datos anteriores se genera la estructura en forma de red de la figura en análisis la cual se empleará para ubicarla como patrón. Además de la estructura se obtiene la longitud de sus segmentos y los ángulos que contiene.

Hecho lo anterior, se procede a ubicar a la figura según la familia topológica a la que más se parezca, en una de cinco posibilidades, que son:

- 1) *Punto:* Se trata de figuras que constan sólo de un punto.
- 2) *Cerrada:* Se especifica de esta forma a las figuras que terminan en el mismo punto en el que inician (figura 6a).
- 3) *Lineal:* Son aquellas figuras que tienen forma de línea y nunca llega a cerrarse (figura 6b).
- 4) *Estrellada:* Las figuras de esta familia se caracterizan por tener un punto central y líneas que salen de ella (figura 6c).
- 5) *Compuesta:* Son las figuras que tienen una combinación de las anteriores (figura 6d).

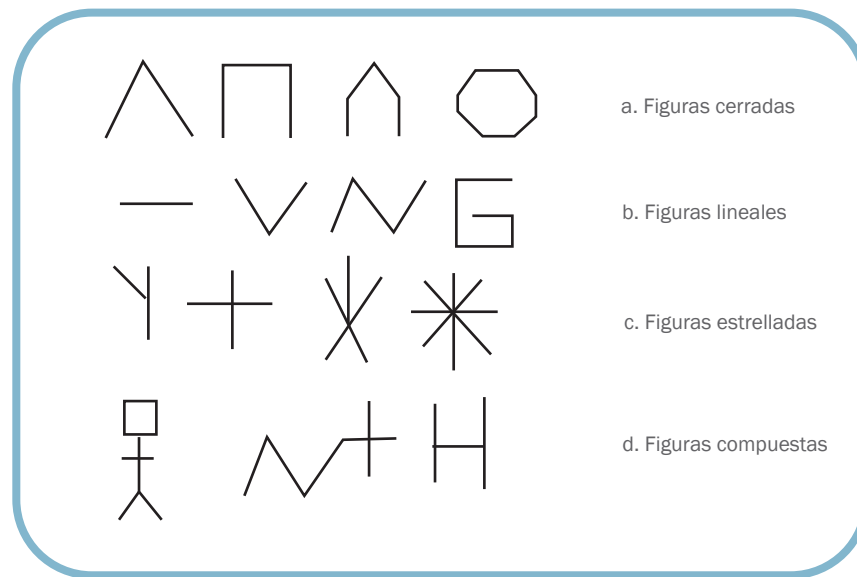


Figura 6. Familias topológicas.

Una vez que la figura queda reconocida en alguna de las familias topológicas, se busca alguna estructura a la cual se parezca. Para esto se emplea un isomorfismo de redes, si la encuentra procede a tomarla como patrón de semejanza y continua el análisis como se describe posteriormente; de otra forma, si no existe alguna estructura a la cual se parezca la figura en análisis, el sistema preguntará si la adopta como patrón o no; si llega a hacerlo almacena los datos que ya tiene y le pide un nombre al usuario para etiquetar la estructura en cuestión. El siguiente nivel de análisis es por figura específica donde el patrón analizado además de pertenecer a la misma familia topológica y tener la misma estructura, posee ángulos y medidas proporcionales al patrón, en cuyo caso se procederá a realizar un aprendizaje, pero ahora por figura específica. *El resultado es el nombre de la figura según el nivel de análisis al cual se haya llegado.*

V Descripción de la Escena

El conocimiento de la figura en sí, muchas veces no es de utilidad por sí solo, pues el uso de la información se da cuando se establecen las relaciones entre estos conocimientos. Hablando de figuras, supongamos una escena cuyos componentes sean letras, es de gran importancia establecer la relación o agrupación de las mismas para conocer el significado de la escena en su totalidad. En el caso de la industria, la posición de las herramientas en la banda y la separación de cada una determinan el movimiento del robot.

Dependiendo del uso o aplicación de la información que se desee obtener, será el planteamiento de la descripción. En el caso del SI-VE la aplicación general utiliza una forma bastante sencilla aunque no muy exacta para describir la escena lo cual se debe a la consideración y libertad de poder modificar en la medida que se requiera dependiendo de su uso. En ella se establecen dos tipos de relaciones. Primeramente cada figura de acuerdo a la posición que mantiene dentro de la escena; y el segundo la relación que guarda cada elemento con los demás. Llamemos a la primera “ubicación” y a la segunda “relación”.

1) *La ubicación*: Se realiza tomando las características extraídas de la figura y con ellas se sobrepone un cuadrilátero que abarca toda la extensión de la figura, dicho cuadrilátero se ubica en la escena según el lugar que ocupe dentro de la misma. Por ejemplo: arriba, abajo, arriba a la derecha, izquierda, etc.

2) *La relación*: Se efectúa después de la ubicación, consiste en determinar que posición ocupa una figura con respecto de otra, comparando todos los puntos que pertenecen a la figura y buscando, entre todas las posibilidades la que sea la más adecuada. Las posibilidades son: arriba, dentro, fuera, junto, encima, etc.

Conclusiones

Cuando alguien habla de casos como el hecho de ver, oír, hablar, entender, etc. se piensa que son funciones obvias y el realizar un programa o sistema que lleve a cabo esto, no tiene mayor problema. Abocándonos al problema particular visto en el desarrollo del presente trabajo, podemos apreciar que no es algo trivial pues involucra una extensa investigación para poder comprender cómo es que los seres vivos ven. Para que un sistema artificial pueda, aunque sea de una forma muy restringida, simular este proceso. Las soluciones planteadas en los estudios hasta hoy realizados no son únicas y definitivas; la solución más adecuada sólo puede obtenerse si se relacionan todas las herramientas y métodos desarrollados en la Inteligencia Artificial.

El beneficio que aporta un dispositivo de visión artificial sería amplio, ya que se utilizaría en robots, para realizar actividades que para el hombre son muy peligrosas, rutinarias o pesadas; ayudará a invidentes o bien en ampliar el campo de visión de tal manera que le permitan ver en lugares donde no puede llegar, como es el caso del océano o del espacio.

CAPÍTULO II

SISTEMAS EVOLUTIVOS

II.1 Sistemas Evolutivos

Fernando Galindo Soria¹

Resumen

En este trabajo se propone y describe en forma general la idea de construir un sistema que en lugar de tener los datos, procesos y estructuras de control de un problema, tenga los mecanismos que le permitan adquirir conocimiento, aprender el lenguaje para comunicarse con las personas (aprender tanto el vocabulario como la sintaxis y semántica del lenguaje), almacenar y relacionar conocimiento, inferir y abstraer nueva información y también los mecanismos para que pueda comunicarse con el medio. A partir de estos mecanismos el siguiente paso es enseñar al sistema como si fuera un bebe que está reconociendo su ambiente y poco a poco aprende a comunicarse y a interpretar las señales que le llegan del exterior. Al principio almacenando datos muy básicos y a partir de estos aprender a reconocer señales externas más complejas y a asociarles un significado y ciertas acciones básicas y así, sucesivamente cambiando acciones y significados ya adquiridos poco a poco, reconocer señales más complejas y por otro lado ir creando redes de conocimiento e ir interconectando estas redes al recibir nuevos datos para crear redes más complejas, además de abstraer a partir de datos concretos reglas generales, las cuales se almacenan “olvidando” en caso de que no se necesiten a los datos concretos.

Introducción

Uno de los problemas más fuertes a los que se enfrenta la informática actual se encuentra en la dificultad de mantener actualizados los sistemas de información, ya que es común que cuando se termina de construir un sistema, existan nuevos requerimientos en la organización o esta haya sufrido cambios en su estructura que no se reflejan en el sistema; por otro lado, aún en el caso de que el sistema esté actualizado es difícil incluir cambios o modificaciones necesarios para mantener una imagen de la organización, ya que el exterior está cambiando en forma dinámica y por su lado el sistema de información es estático, este problema ha propiciado que se empiecen a estudiar mecanismos informáticos que hagan más dinámicos a los sistemas de información y que permitan insertar los cambios externos en forma natural y sencilla.

¹ Fernando Galindo Soria escribió este trabajo en junio de 1985 estando en el Departamento de Computación, Unidad de Investigación y Desarrollo, UPIICSA, IPN. Se presentó en la Universidad La Salle. Este trabajo apareció publicado en el Boletín de Política Informática en 1986.

Dentro de estos mecanismos se ha pensado en desarrollar un sistema basado en las herramientas de la inteligencia artificial, capaz de captar las modificaciones del ambiente y reconfigurarse internamente para representar estas modificaciones, este mecanismo puede caer en alguno de los siguientes niveles:

- 1) *Estáticos*: Capaces de mantenerse en un medio que no cambia o con pocos cambios (nicho ecológico).
- 2) *Adaptativos Estáticos*: Son capaces de adecuarse al medio estable.
- 3) *Transformadores*: Capaces de influir para que cambie el medio.
- 4) *Adaptativos Dinámicos*: Capaces de mantenerse en un medio que se está transformando.
- 5) *Evolutivos*: Capaces de transformarse en un medio que se está transformando.

La idea que se presenta en este trabajo versa sobre el desarrollo de sistemas evolutivos, puesto que si se tiene un sistema evolutivo, el sistema adaptativo dinámico o copia de la organización sería un caso particular del evolutivo.

I Características de un Sistema Evolutivo

La construcción de un sistema evolutivo tiene múltiples problemas interesantes, ya que se trata de encontrar un mecanismo que se desarrolle “crezca” en forma independiente al usuario; entre otros problemas se observan los siguientes:

- 1) El sistema debe contar con un mecanismo que le permita *captar la realidad* que lo rodea, ya que necesita conocer y estudiar el ambiente con el fin de detectar las diferencias y cambios que requiere para poder adaptarse y evolucionar en ese medio.
- 2) El sistema debe ser *capaz de almacenar y representar el conocimiento*, con el fin de construir su propia representación de la realidad y poderla explorar.
- 3) El sistema debe ser *capaz de “generar” nuevo conocimiento* a partir del que tiene almacenado y del que capta del exterior, con el fin de que pueda proponer cambios o modificaciones a su imagen de la realidad incluyendo ese nuevo conocimiento.
- 4) El sistema debe ser *capaz de abstraer a partir de un conjunto de conocimientos*, reglas generales que los representen en forma sintética.
- 5) El sistema debe ser *capaz de establecer un diálogo con el exterior* con el fin de que pueda transmitir su conocimiento y mediante la retroalimentación propiciar el cambio en el exterior.

II Restricciones sobre el Sistema

Como se puede observar, un sistema evolutivo es de una complejidad enorme y es difícil que la tecnología con la que se cuenta actualmente lo pueda soportar; sin embargo, introduciendo algunas restricciones se puede proponer un primer prototipo.

Las primeras restricciones surgen sobre el mecanismo de adquisición del conocimiento, ya que por ejemplo, el ser humano tiene múltiples herramientas para captar información del medio que lo rodea, como son: vista, oído, tacto, etc., y estos mecanismos captan millones de señales

del exterior en un momento dado y estas señales conforman la información que recibe y procesa la persona; sin embargo, en una computadora sólo se tienen algunas herramientas de captura de información del exterior (pantallas, lectoras, etc.) con una capacidad de captación de información muy pequeña en comparación con la del ser humano, por lo que, en un prototipo en principio se puede pensar en un mecanismo de captura de información conformado por un lenguaje escrito, y herramientas de recepción gráfica y de señales.

El segundo grupo de restricciones se presenta sobre los mecanismos de almacenamiento del conocimiento, ya que esta es un área de investigación nueva y existen pocas herramientas sin embargo, ya se pueden almacenar hechos y relaciones entre los hechos.

Por su parte, el problema de generación y abstracción de nuevo conocimiento sólo se puede atacar a un nivel muy elemental, ya que no se tienen herramientas para abstraer de un conjunto de hechos la regla general que los engloba, por lo que principalmente se cuenta con herramientas para inferir nuevo conocimiento, para relacionar conocimientos aparentemente no relacionados (al introducir más información) y para recordar y olvidar en forma selectiva.

Finalmente, los mecanismos de diálogo hombre máquina se encuentran en sus inicios y dependen principalmente de los puntos anteriores, puesto que para establecer un diálogo se requiere que el sistema tenga posibilidad de captar, almacenar y generar conocimiento.

III Tipo de Información en el Sistema

Otro punto que se tiene que considerar al desarrollar un sistema evolutivo se presenta cuando se decide que tipo de información debe ser capaz de captar el sistema.

Para clarificar este punto conviene recordar que en sus inicios la informática se orientaba al desarrollo de los procesos necesarios para resolver un problema, posteriormente se observó que los procesos lo único que hacen es transformar los datos, por lo que los métodos de desarrollo de sistemas de información se centraron en los datos y en las estructuras de datos, de donde el enfoque en el desarrollo de sistemas marca que primero se tienen que localizar los datos que se desea obtener del sistema, de esto se pasa a definir los datos que se tienen que almacenar (para poder generar los que se necesitan) y de ahí proponer los datos que deben entrar, y a partir de todo eso se definen los procesos y estructuras de control que se requieren.

Como se puede observar, de los tres componentes con los que se representa internamente a una organización (datos, procesos y estructuras de control) el enfoque de desarrollo ha migrado de los procesos a los datos, ahora bien, un modelo que se propone actualmente hace énfasis en la relevancia de la estructura de control, ya que es común que al modificarse una organización los procesos y datos involucrados a nivel operativo cambien poco. Sin embargo, la estructura de control se modifica parcial o totalmente (desaparece un departamento o se crea un nuevo puesto, etcétera).

Por lo anterior, se propone contar con sistemas en los cuales se represente de alguna forma el flujo de control de la organización y de ahí se obtenga el flujo de datos y los procesos involucrados.

Entonces en un modelo de este tipo se tienen tres mecanismos que permiten representar la organización, la estructura de datos de la organización y las rutinas operativas, y por otro lado se tiene un programa administrador, el cual, a partir de los tres mecanismos, resuelve los problemas de la organización, para lo cual toma como guía la representación de la organización y almacena, transforma y entrega datos al usuario.

Este modelo tiene la ventaja de que si cambia la estructura de la organización, no es necesario reescribir programas de control sino únicamente modificar la representación de la organización para indicar su nueva estructura e introducir o quitar rutinas operativas.

IV Propuesta de Sistema

De lo anterior surge una propuesta de Sistema Evolutivo, en la cual se tienen mecanismos que permiten construir un modelo de organización incluyendo los tres aspectos considerados dentro de un sistema de información (estructura de la organización, estructura de datos y rutinas semánticas u operativas), para lo cual se propone que el sistema tendría involucradas dos grandes componentes: un constructor y un administrador.

El constructor es el responsable de captar una imagen de la realidad y representarla en los tres mecanismos y, por su parte, el administrador sería el responsable de lograr que el sistema evolucione y dar respuestas a los problemas de los usuarios.

V Constructor del Sistema Evolutivo

Para construir un sistema evolutivo se tienen dos grandes partes, en la primera se debe construir toda la herramienta que permita al sistema captar una realidad, entre estas herramientas se necesita:

- 1) Un mecanismo que permita al sistema “aprender” un *lenguaje de comunicación* con el usuario, incluyendo la capacidad para aprender palabras aisladas y su significado, estructuras oracionales válidas y oraciones válidas y su significado en términos de palabras y de otras oraciones.
- 2) Un mecanismo que permita captar a partir del lenguaje que “aprende”, las *estructuras de la organización* (la cual se encuentra inmersa en la sintaxis de las oraciones), las estructuras de datos y las rutinas semánticas (las cuales corresponden a la semántica asociada con las oraciones).
- 3) Un mecanismo que permita al sistema “aprender” a *reconocer figuras y señales y su significado* y en un momento le permita distinguir entre diferentes señales gráficas y asociarles un significado.
- 4) Mecanismos que permitan *abstraer, de datos concretos, ideas generales* y otros que permitan inferir de unos datos, nuevo conocimiento.

La segunda parte del constructor se debe orientar a crear la base mínima para que un sistema de este tipo pueda funcionar. Esta base podría incluir, entre otros:

- 1) Una *lengua básica* (por ejemplo, español muy elemental) la cual se tendría como punto de partida ya que en un momento dado se le podrían incluir las operaciones más básicas que requiere para funcionar (por ejemplo: lee, escribe, suma, etcétera).

- 2) *Reconocimiento de patrones muy elementales* (por ejemplo: símbolos que indiquen áreas de la organización y flujo de datos entre diferentes áreas)
- 3) *Estructuras gramaticales* de la lengua básica y su representación en términos de estructura de la organización, estructuras de datos y rutinas semánticas.
- 4) *Reglas de abstracción e inferencia* primitivas y basadas, por ejemplo, en las características de las relaciones (transitividad, equivalencia, etcétera).

VI Administrador del Sistema Evolutivo

Por su parte, el administrador tendría una doble función, ya que debe ser capaz de dar respuesta a los requerimientos del exterior y además mantener evolucionando al sistema.

Para lograr que el sistema evolucione, el administrador debe poder captar los cambios en el exterior con el fin de adecuarse a estos, y por otro lado debe abstraer e inferir a partir del conocimiento almacenado nuevos conocimientos y relaciones no captados del exterior.

Con base en el conocimiento generado internamente, el almacenado previamente y el captado del exterior se deben obtener nuevas propuestas o modelos del ambiente que rodea al sistema.

Se debe contar con un mecanismo que permita contrastar estas propuestas con el ambiente real y en su momento facilite el aprendizaje, reforzamiento u olvido de algunos segmentos de conocimiento.

Finalmente, el administrador debe poder dar respuesta a los requerimientos de los usuarios del sistema mediante un diálogo en lenguaje natural o mediante representación gráfica.

VII Casos Particulares de Sistemas Evolutivos

El sistema evolutivo en toda su complejidad sería equiparable a ciertos tipos de organismos vivos que se adaptan y evolucionan en algún ambiente dado, en su momento y dependiendo del grado de complejidad del sistema podría mostrar algunos procesos de tipo “inteligente”; sin embargo, la tecnología actual requiere solucionar varios problemas de una alta complejidad, por lo que en un momento dado se puede pensar en mecanismos más sencillos en los cuales ya se tiene una tecnología básica y a partir de estos podrían “evolucionar” los sistemas evolutivos.

Entre los sistemas con los que se cuenta actualmente se tienen los que se conocen como sistemas expertos, los cuales son capaces de atacar y resolver problemas de áreas de conocimiento específicas con base en que tienen una representación del conocimiento manejado en esa área y un mecanismo que les permite inferir nuevo conocimiento a partir del ya almacenado.

Otro caso particular de sistemas que se puede desarrollar sería el de los adaptativos, tanto estáticos como dinámicos, ya que en este caso en lugar de darles el conocimiento de algún área se les podría proporcionar la descripción de alguna organización específica incluyendo las estructuras de la organización, los datos que se manejan y las rutinas semánticas que se utilizan,

y con base en esa descripción crear un modelo de la organización, el cual sólo se tendría que mantener actualizado introduciéndole los cambios en la organización mediante un proceso de “altas”, “bajas” y “cambios” sobre el modelo.

Finalmente, un sistema evolutivo sencillo se puede desarrollar restringiendo sus mecanismos de adquisición, almacenamiento, generación y presentación de conocimiento y buscando fortalecer aquellos mecanismos orientados a aprender una lengua nueva incluyendo la semántica asociada y los mecanismos que permitan integrar el conocimiento al ya adquirido y proponer nuevos modelos del exterior.

Conclusiones

Un sistema como el propuesto en este documento es factible de desarrollar; sin embargo, existen muchos problemas abiertos y no resueltos, por lo que en un momento dado conviene desarrollar prototipos del sistema y atacar los diferentes problemas en forma particular e integrando los resultados a la idea general.

Actualmente, en la Unidad de Investigación y Desarrollo del Departamento de Computación (UIDC) de la UPIICSA se tienen varios trabajos orientados a la construcción de un sistema de este tipo, incluyendo la búsqueda de mecanismos que permitan a un sistema aprender una lengua, esqueletos de sistemas expertos, sistemas adaptativos estáticos y los problemas de representación del conocimiento.

Sin embargo, la complejidad del problema es enorme y su campo de aplicación ilimitado, ya que tanto los sistemas expertos como los adaptativos pasan a ser un caso particular y en su momento un sistema de este tipo podría evolucionar en múltiples áreas y atacar problemas no cubiertos.

Es por lo anterior que la idea de construir sistemas evolutivos se presenta en este trabajo con el fin de que más personas y grupos se involucren en este tipo de problemas.

Bibliografía

- [1] González Vázquez, Juan Martín, “La Comunicación Hombre Máquina Utilizando Lenguaje Natural Restringido”, Departamento de Computación, UPIICSA, Proyecto Censo Agropecuario, INEGI, SPP, 1985.
- [2] Mendoza Padilla, Roberto, “Sistemas Adaptativos”, Departamento de Computación, UPIICSA-IPN 1985.
- [3] Cordero Sánchez, Gabriel, “Aplicación de un Reconocedor de Lenguaje Natural Restringido a la Recuperación de Datos”, Departamento de Computación, UPIICSA-IPN, 1985, Subdirección de Análisis y Diseño de Sistemas DGIA-SEP
- [4] Giles Galaz, Víctor Miguel Ángel, “Construcción de Sistemas Expertos”, Departamento de Computación UPIICSA-IPN, 1985.

- [5] Martínez Cortés, Rosalía; Najar Alvarez; Mireya, Rodríguez Anzures, Alfredo, “Sistemas Expertos de Diagnóstico Médico”, Departamento de Computación, UPIICSA-IPN, Licenciatura en Informática UPIICSA-IPN, 1985.
- [6] García Moctezuma, Rocío, Fierro A., Ma. Antonieta, Cruz Velázquez, Edmundo, “Biblioteca Automatizada”, Departamento de Computación, Licenciatura en Informática, UPIICSA-IPN, 1985.
- [7] Galindo Soria, Fernando, “Notas del Seminario sobre Desarrollo de Sistemas”, Departamento de Computación, UPIICSA-IPN, 1985.
- [8] Galindo Soria, Fernando, “Representación de Conocimiento”, Departamento de Computación UPIICSA-IPN, 1985.
- [9] Díaz Bello, Lino, “Construcción de Compiladores”, Departamento de Computación UPIICSA-IPN, Unidad de Cómputo, El Colegio de México, Tesis de Titulación ESIME-IPN, 1982.

II.2 Sistemas Evolutivos: Nuevo Paradigma de la Informática

Fernando Galindo Soria²

Resumen

En este trabajo se presenta una introducción a los Sistemas Evolutivos (sistemas capaces de construir y mantener en tiempo real una imagen del ambiente que los rodea, como un ser que construye su propia imagen de la realidad y la utiliza para interactuar con su entorno), incluyendo su descripción general y presentación de algunos de los métodos y algoritmos más usados en la construcción de este tipo de sistemas principalmente la Arquitectura del Sistema Evolutivo y su estructura Lingüística, se describe el Mecanismo Léxico, el Mecanismo Sintáctico, las principales herramientas de Inferencia Gramatical, el Mecanismo Semántico y las principales herramientas de Diálogo y de asociación de significado.

Palabras Clave: Sistemas Evolutivos, Lingüística Matemática, Reconocimiento de Formas, Sistemas de Información, Base de Datos, Inferencia Gramatical, Herramientas Automatizadas.

Introducción

En éste trabajo se describe en forma general la arquitectura y aplicaciones de los Sistemas Evolutivos, donde un Sistema Evolutivo es una herramienta automatizada capaz de construir una imagen del ambiente que la rodea y a partir de esta imagen resolver problemas, como un niño que construye su propia imagen de la realidad y la utiliza para interactuar con su entorno.

Como primer punto se describe en forma general, cuales son los principales problemas de la Informática actual y la arquitectura general de los Sistemas de Información y a partir de este punto se sigue una secuencia en espiral donde en cada vuelta de la secuencia se replantea la arquitectura de los sistemas de información hasta terminar en la conclusión con la arquitectura generalizada de los Sistemas Evolutivos.

I 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

² Fernando Galindo Soria escribió este trabajo en 1990 siendo profesor-investigador en la Licenciatura en Ciencias de la Informática Sección de Graduados e Investigación UPIICSA-IPN.

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 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 porqué se esta tratando de modelar un problema que cambia prácticamente por decreto y en el que es difícil, establecer los patrones de cambio).

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, por lo que los Sistemas de Información orientados al reconocimiento de patrones reflejan un modelo de la realidad normalmente restringido.

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 porqué 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 incapaces de recrear en forma continua su imagen de la realidad o del problema a resolver.

Es dentro de este contexto donde surge el concepto de Sistema Evolutivo como un sistema capaz de crear su propia imagen de la realidad y utilizarla para resolver problemas y en su momento mantener actualizada esta imagen mediante un proceso continuo de actualización.

II Del Sistema de Información al Sistema Evolutivo

1 Rumbo al Constructor

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

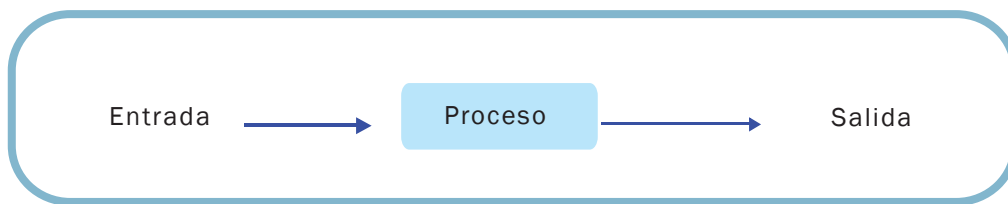


Figura 1. Arquitectura de los sistemas de información en los 50s.

Sin embargo este esquema es muy viejo y corresponde a los años 50's y ya para la década de los 60's se manejaban los esquemas más elaborados mostrados en la figura 2.

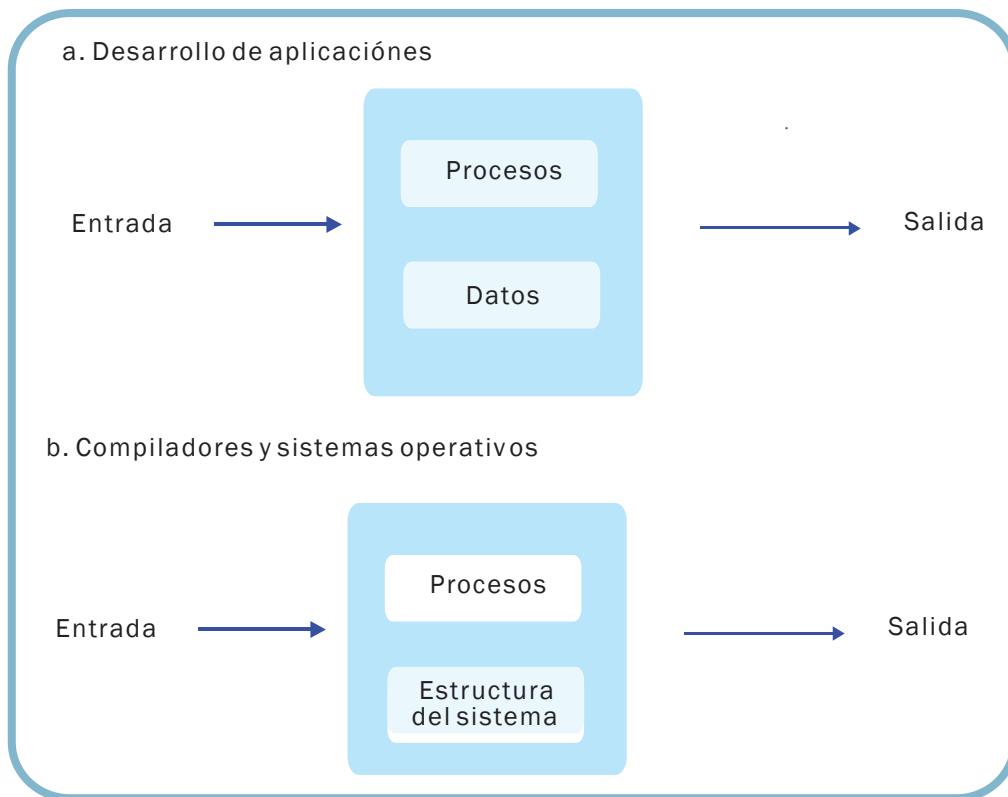


Fig 2. Arquitectura de los sistemas de información en los 60s.

El primer esquema se utilizaba principalmente por la gente orientada al desarrollo de aplicaciones y el segundo por las personas dedicadas a desarrollar herramientas de cómputo como Compiladores y Sistemas Operativos, y 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.

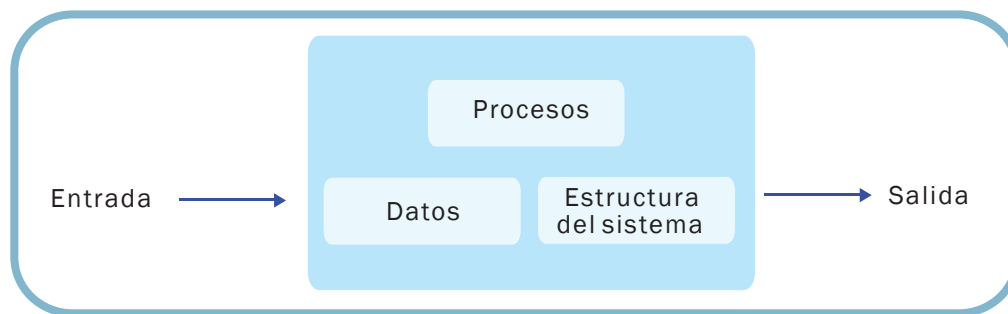


Figura 3. Arquitectura de los sistemas de información en los 70s.

Este esquema integra los enfoques anteriores y enfatiza la necesidad de desarrollar los sistemas tomando en cuenta las tres componentes y no solo los procesos, sin embargo, 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.

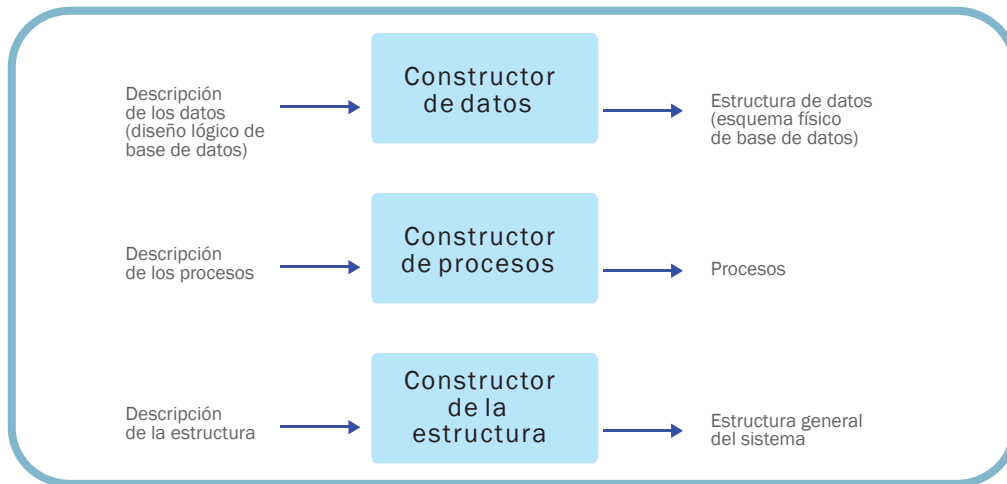


Figura 4. Constructores de datos, procesos y estructura.

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

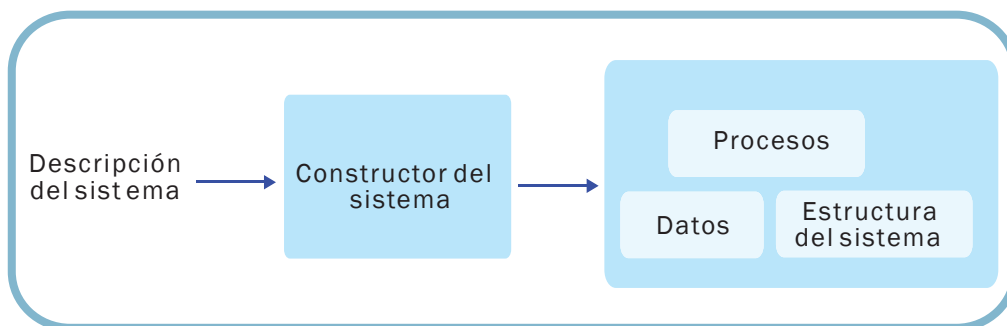


Figura 5. Integración de los constructores en un constructor del sistema.

2 Rumbo a la Arquitectura de un Sistema Evolutivo

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. 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), por lo que 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.

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.

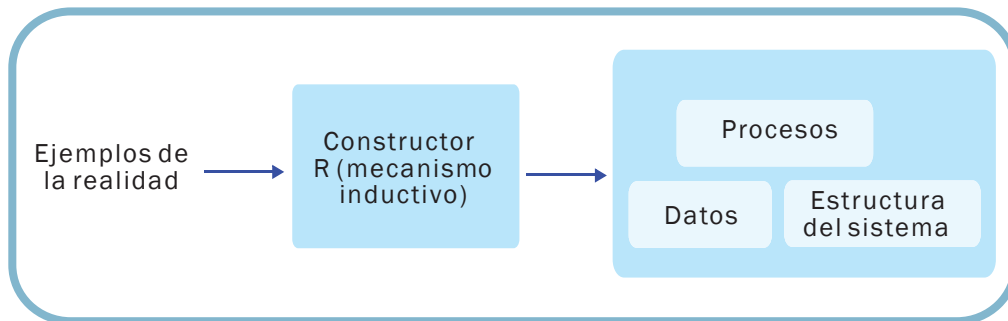


Figura 6. Herramienta inductiva.

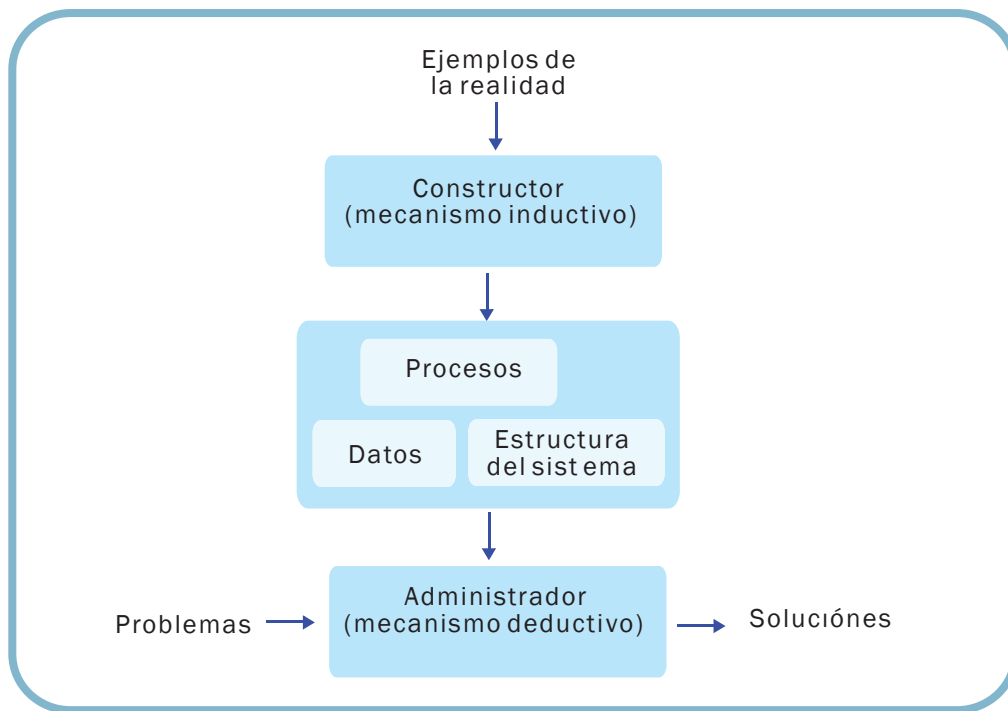


Figura 7. Integración de la componente deductiva.

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

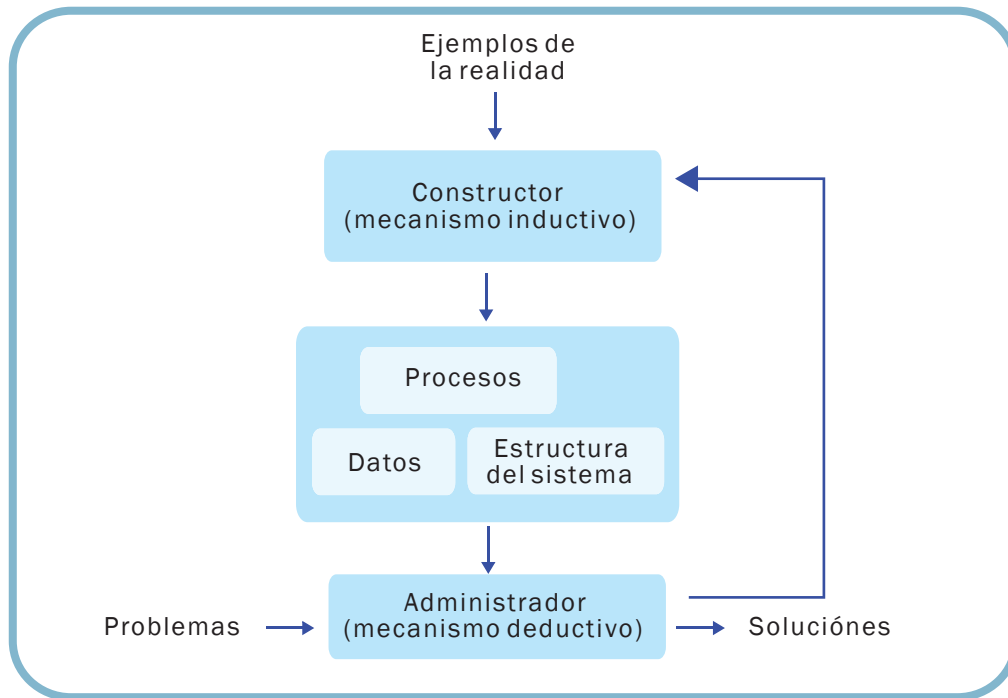


Figura 8. Retroalimentación para actualizar la imagen de la realidad.

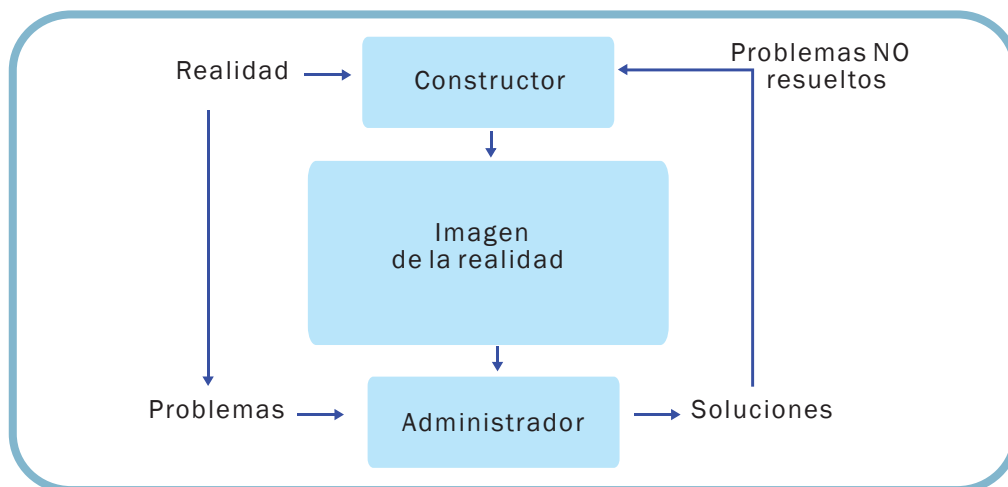


Figura 9. Arquitectura General de un Sistema Evolutivo.

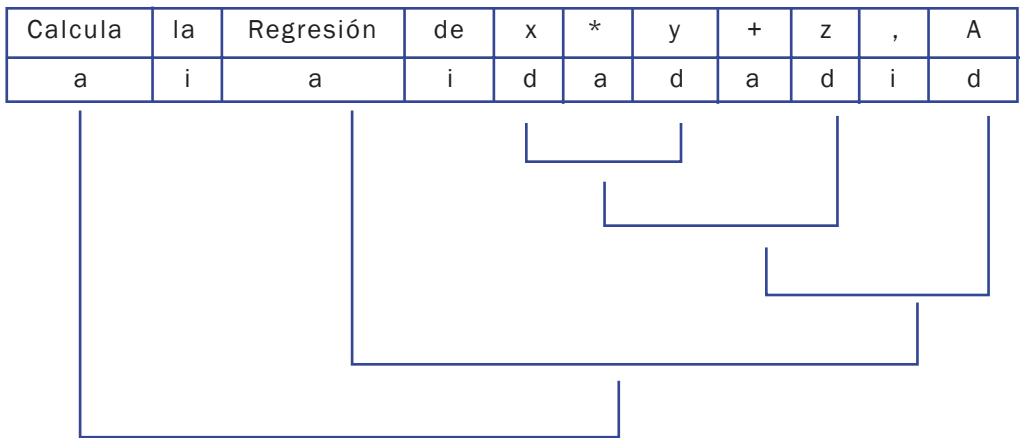
III Arquitectura

En el punto anterior se vio que con el fin de resolver problemas el sistema evolutivo construye una imagen de la realidad, pero no se indicó como se lleva a cabo este proceso, por lo que en este punto se describirá en forma general los principales métodos y algoritmos utilizados por los Sistemas Evolutivos; para lo cual se partirá de que el lenguaje de comunicación con el exterior (o sea el lenguaje con el que se plantean los problemas al Sistema Evolutivo y con el cual éste responde a los requerimientos) y el lenguaje con el que el Sistema Evolutivo construye su imagen de la realidad son el mismo.

En la actualidad ya existen una gran cantidad de herramientas orientadas al manejo lingüístico de los Sistemas Evolutivos y se basan en que es relativamente fácil encontrar los elementos de un Sistema dentro de las oraciones, por ejemplo en la oración:

Calcula la Regresión de $X * Y + Z$, A

Es relativamente fácil detectar los datos, acciones y estructura:



La estructura se indica con las líneas y representa el orden en que se ejecutan las acciones sobre los datos, los datos tienen el tipo d, las acciones el tipo a y las palabras no relevantes llevan i (ignora).

Entonces un Sistema Evolutivo debe tener la capacidad de encontrar los componentes del sistema de información a partir del lenguaje utilizado en el área problema y para lograrlo muchos sistemas evolutivos tienen la arquitectura de la figura 10.

Donde el Constructor/Analizador Léxico es el encargado de encontrar cada una de las unidades léxicas que componen una oración, de que tipo es cada unidad (por ejemplo dato u oración) y la estructura general de la oración conocida como oración canónica (formada por la concatenación de los tipos de las unidades léxicas).

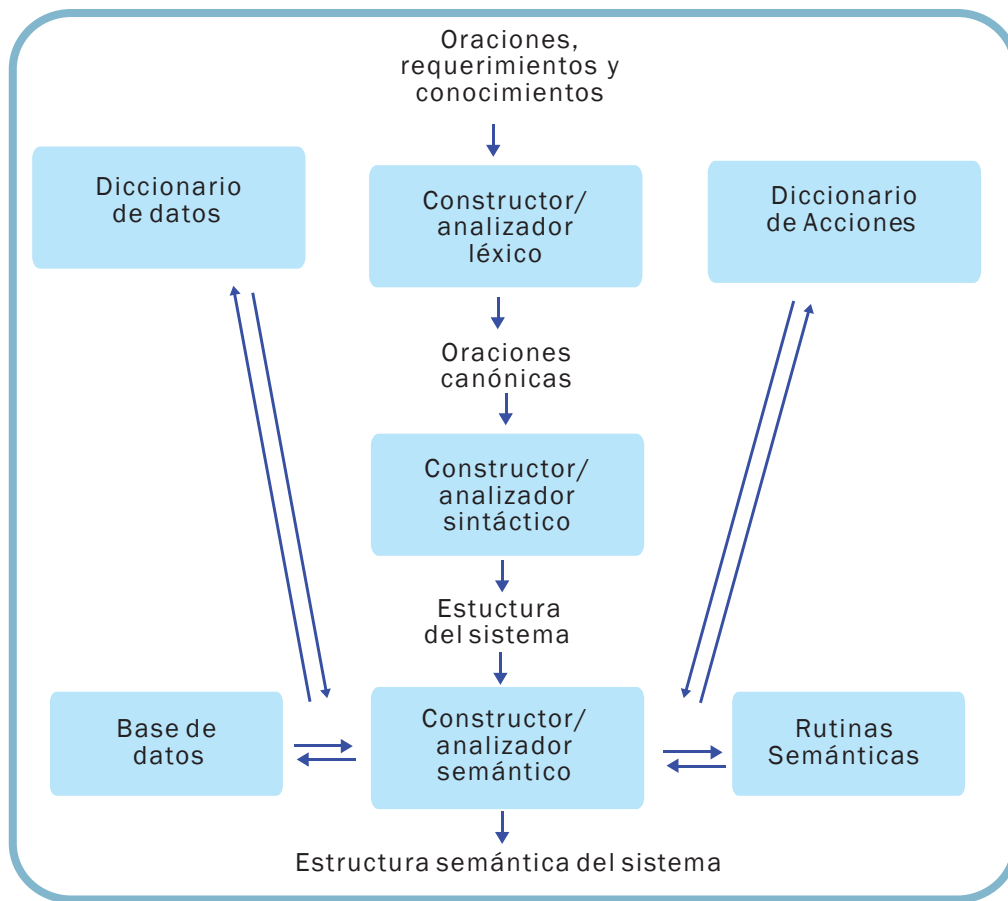


Figura 10. Arquitectura de un Sistema Evolutivo.

A partir de ahí el Constructor/Analizador Sintáctico aplicando métodos de inferencia gramatical (métodos orientados a encontrar la gramática de un lenguaje a partir de ejemplos de oraciones del lenguaje) encuentra la estructura del sistema y finalmente el Constructor/Analizador Semántico, encuentra significado de la oración utilizando mecanismos de diálogo para encontrar el significado de los elementos desconocidos.

A continuación se describen algunos de los principales métodos utilizados por cada uno de los constructores/analizadores.

1 Constructor/Analizador Léxico

El Constructor/Analizador Léxico es un programa que recibe como entrada una oración en algún lenguaje y obtiene las unidades léxicas presentes en la oración, el tipo de cada unidad léxica y la estructura general u oración canónica.

Por ejemplo, dada la oración:

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 Constructor/Analizador 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 para lo cual utiliza entre otros los siguientes métodos “triviales”:

- 1) *Unidades predefinidas*: Es el método más usado y el más antiguo, consiste básicamente en almacenar previamente la tabla de unidades léxicas con las palabras que se utilizan en el sistema indicando de que tipo son. Con la característica de que en un sistema evolutivo esta tabla es creciente (o sea que originalmente la tabla tiene las unidades léxicas mas usadas en el sistema y se cuenta con la capacidad de almacenar nuevas unidades conforme se van detectando).
- 2) *Caracteres especiales*: El mecanismo anterior tiene la limitante de que si se requiere utilizar una palabra no catalogada el sistema tiene que preguntar y esto al principio puede ser tedioso, por lo que, cuando se construyeron los primeros ensambladores y compiladores se utilizó el truco de manejar ciertos caracteres o posiciones específicas dentro del programa para que el sistema asignara las unidades léxicas. Por ejemplo en un ensamblador dado, la instrucción:

```
:Etiqueta      .Mem      #3 ;Coloca 3 en la etiqueta
es fácilmente analizable si se tiene la convención de que:
: Indica Tipo de Etiqueta
. Indica Tipo de Instrucción
# Indica tipo de Dato
; Indica Tipo de Comentario
```

En general esta idea se puede utilizar para catalogar nuevas unidades Léxicas.

- 3) *Mecanismo de Diálogo*: 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.

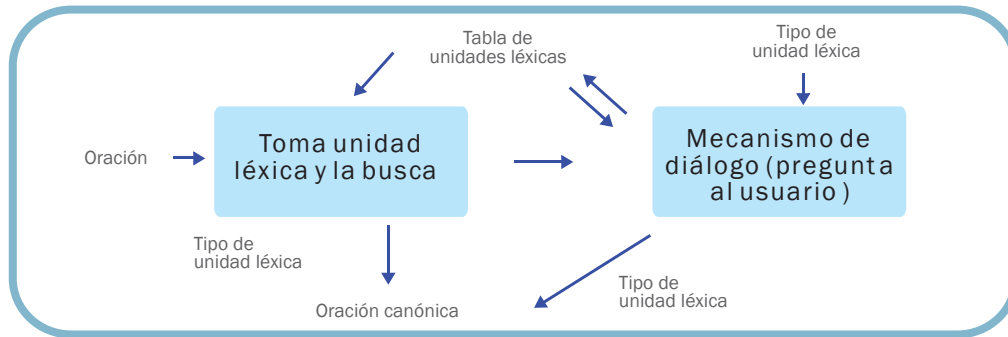


Figura 11. Mecanismo de diálogo.

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

4) *Manejo de Contexto*: Los anteriores métodos tienen el inconveniente de que todo se le tiene que dar al sistema y no utilizan mucha de la información ya obtenida. En el momento que ya se han manejado varias oraciones se pueden empezar a detectar algunos patrones que se pueden utilizar para encontrar el tipo de nuevas unidades léxicas. Por ejemplo, si después de catalogar 10 oraciones se encontró que la estructura:

a i a i d

se repite 4 veces y el sistema recibe la oración

Calcula la media de A

donde ya conoce el tipo de las unidades

calcula (a)

la (i)

de (i)

A (d)

donde a=acción, d=dato, i=ignora; entonces la oración canónica tiene la forma:

a i ? i d

Por lo que el sistema propone que el elemento desconocido

Media es de tipo a.

Los anteriores métodos “triviales” al combinarse son extremadamente poderosos y en su momento con una cantidad relativamente pequeña de ejemplos son capaces de encontrar el conjunto base de palabras manejadas por un área problema.

Existen otros métodos más poderosos pero normalmente requieren del apoyo de los mecanismos Sintáctico y Semántico, estos últimos normalmente no se utilizan en los prototipos de Sistemas Evolutivos.

2 Constructor/Analizador Sintáctico

A partir de los resultados del Constructor/Analizador Léxico el Constructor/Analizador 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 más 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.

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 consiste básicamente en encontrar la Gramática (Estructura) que describe a un lenguaje dado a partir de ejemplos de oraciones del Lenguaje, figura 12.

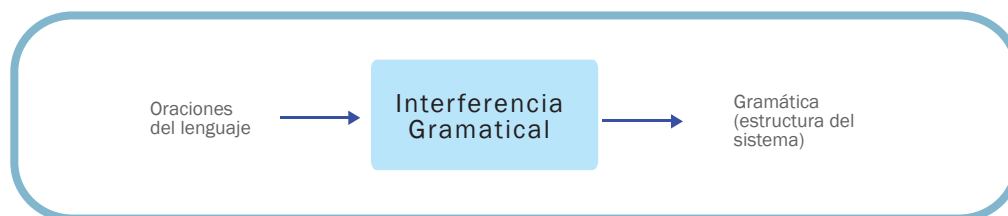


Figura 12. Inferencia gramatical.

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 prácticamente los métodos de inferencia gramatical se basan en las operaciones básicas de:

Factorización, Distribución y Recursividad por lo que a continuación se explicarán estas operaciones.

3 Factorización Lingüística

En su forma más sencilla la factorización lingüística lo que pretende precisamente es encontrar los factores comunes en diferentes oraciones.

Por ejemplo si se tienen las siguientes oraciones canónicas (S indica el primer nivel de factorización).

$$\begin{aligned} S &\rightarrow a i b i a d o f l \\ &\quad a i b i d o f l \\ &\quad a i d a d f \end{aligned}$$

Factorizando la cadena **a i** que es común a las tres oraciones de S

$$\begin{aligned} S &\rightarrow a i X \\ X &\rightarrow b i a d o f l \\ &\quad b i d o f l \\ &\quad d a d f \end{aligned}$$

Factorizando **b i** en las dos primeras oraciones de X queda

$$\begin{aligned} S &\rightarrow a i X \\ X &\rightarrow b i Y l \\ &\quad a d o f \\ Y &\rightarrow a d o f l \\ &\quad d o f \end{aligned}$$

Si se observa, lo único que hace el proceso de factorización es sacar los factores comunes (que se repiten) de las diferentes oraciones.

4 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 más 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.

$$\begin{array}{l} S \rightarrow a X f \\ X \rightarrow \underline{b c} \underline{b c} \underline{b c} \underline{b c} \\ \quad \downarrow \\ S \rightarrow a X f \\ X \rightarrow b c X \end{array}$$

Por ejemplo si se tiene

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

Introduciendo recursividad sobre **a d** queda

$$\begin{array}{l} S \rightarrow a i X f \\ X \rightarrow a d a d a d a d \\ \quad \downarrow \\ S \rightarrow a i X f \\ X \rightarrow a d X \end{array}$$

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 más de lo debido.

En general, el Constructor/Analizador obtiene la estructura de un sistema a partir de las oraciones del lenguaje y utilizando en forma combinada la recursividad, distribución y factorización.

5 Constructor/Analizador Semántico

Esta componente del Sistema Evolutivo es la encargada de asociar el significado a los elementos y estructura del sistema y se basa principalmente en un mecanismo de diálogo y en la construcción de relaciones entre los elementos utilizando diferentes mecanismos de captación de la realidad (en la actualidad utiliza principalmente imágenes y lenguaje escrito).

El mecanismo de Diálogo fue originalmente propuesto por José Luis Díaz Salas en 1987 y posteriormente ha sido complementado por otros investigadores y consiste básicamente en un proceso recursivo en el cual cada que el sistema Evolutivo detecta una palabra u oración para la que no encuentra significado simplemente pregunta y espera una respuesta en el mismo lenguaje.

Por ejemplo si se tiene la oración:

calcula la Media de A

y la computadora no sabe lo que significa la palabra Media, pregunta, por lo que se puede establecer un diálogo del siguiente tipo:

Usuario: *Calcula la media de A*
Computadora: *¿Cómo se obtiene la Media?*
Usuario: *Suma los elementos de A y divídelos entre el número de elementos*

Este mecanismo ha sido modificado con el fin de permitir que se definan Sinónimos o unos elementos en término de otros.

Conclusiones

El área de los Sistemas Evolutivos es muy extensa y en este documento sólo se presentó una introducción al tema ya que desde 1985 se han desarrollado gran cantidad de sistemas de este tipo principalmente a nivel de prototipos, pero ya en la actualidad varias empresas y entidades públicas están construyendo sus propios Sistemas Evolutivos y al menos en un caso se está trabajando en una herramienta de mercado, por lo que ésta es una invitación a que más investigadores e instituciones se sumen a este esfuerzo.

Algunas Publicaciones Relacionadas con Sistemas Evolutivos

Construcción de Sistemas Evolutivos
De la Biblioteca a la Memoria Automatizada
Representación de Conocimiento
Programación Dirigida por Sintaxis
Inferencia Gramatical
Introducción a la Lingüística Matemática
Sistema Evolutivo Constructor de Sistemas Expertos
Sistemas Evolutivos de Lenguaje de Trayectorias
Herramientas Generales Para el desarrollo de juegos y Software Educativo
Herramientas Generales para el Desarrollo de Sistemas (Núcleo de un Sistema Evolutivo)

II.3 Arquitectura Lingüístico-Interactiva para Sistemas Evolutivos

Fernando Galindo Soria³

I Introducción a los Sistemas Evolutivos

Los Sistemas Evolutivos (S.Ev.) 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 (S.Ev.) se comporta como un niño que está aprendiendo y aplicando este aprendizaje a su entorno, ya que de entrada, *el S.Ev. 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 1985 y los primeros productos a nivel de prototipo se presentaron en 1986; 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:

- 1) *Generar esquemas lógicos de base de datos a partir de lenguaje natural.*
- 2) *Reconocimiento de imágenes.*
- 3) *Generación de sistemas expertos.*

³ Fernando Galindo Soria realizó este trabajo cuando se encontraba en la Sección de Posgrado e Investigación de la UPIICSA, IPN.

- 4) *Construcción y explotación de bases de conocimiento.*
- 5) *Control de robots mediante lenguaje natural y lenguajes de trayectoria.*
- 6) *Reconocimiento y corrección de errores ortográficos.*
- 7) *Generación automática de sistemas de información.*
- 8) *Construcción de paisajes en dos dimensiones.*

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.

II Arquitectura General de un Sistema Evolutivo

Cuando se constituye 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 S.Ev. 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.

Para llevar a cabo lo anterior se han desarrollado muchas propuestas y se han construido gran cantidad de sistemas y herramientas. En la actualidad se cuenta con varios enfoques de cómo construir este tipo de sistemas, por lo que en particular presentaremos uno de estos enfoques basados en los conceptos de la Lingüística Matemática y del desarrollo de sistemas interactivos.

En general la arquitectura de un S.Ev. se puede ver constituida por tres grandes módulos interrelacionados (figura 1):

- 1) *La representación de la realidad.*
- 2) *El manejador del S.Ev.*
- 3) *Los mecanismos de interacción con el ambiente.*

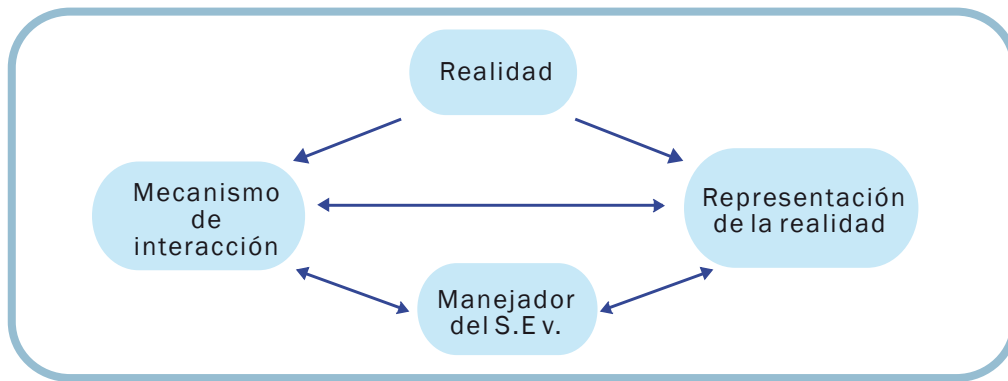


Figura 1. Arquitectura general de un Sistema Evolutivo.

El modelo de Representación de la Realidad nos permite almacenar una imagen de una realidad dada en términos de sus elementos, las relaciones entre éstos o estructura y su significado (figura 2).

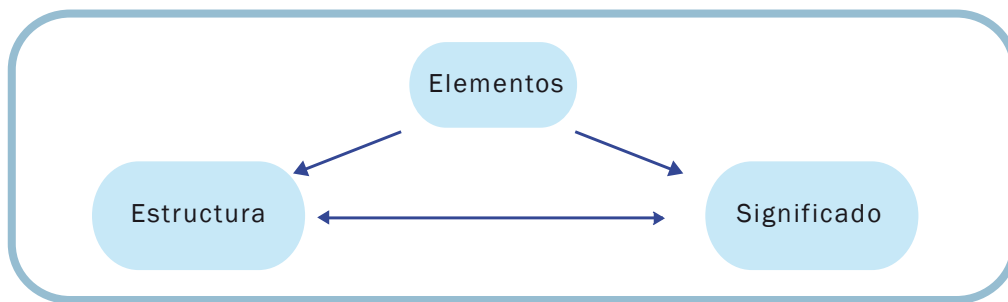


Figura 2. Representación de la realidad de un Sistema Evolutivo.

Por su parte el Manejador del S.Ev. 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.

Para lograr su propósito el manejador consta generalmente de tres módulos:

- 1) *Un Manejador Léxico/Semántico*, responsable de encontrar los elementos que forman parte de una realidad dada y su significado.
- 2) *Un manejador Sintáctico/Semántico*, el cual tiene como función principal encontrar la estructura general o reglas inmersas en esa realidad.
- 3) *Un manejador Semántico/Hermenéutico*, responsable de que el sistema encuentre una interpretación propia de lo que lo rodea y asocie esa interpretación tanto a los elementos como a la estructura.

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.

III Universo Lingüístico Básico

Cualquier persona que haya desarrollado o conozca cómo funciona un compilador o intérprete de algún lenguaje podrá observar que la arquitectura del Manejador del S.Ev. “se parece” a la del compilador; sin embargo, existe una diferencia fundamental, ya que en el caso de un compilador, éste ya cuenta con un lenguaje de programación inamovible definido por la persona que diseñó el compilador y formado por tipos de elementos, reglas sintácticas y significados específicos, por lo que si el sistema no reconoce alguna instrucción, simplemente manda un mensaje de error y normalmente termina el proceso; por su parte el S.Ev. cuenta con la capacidad para encontrar elementos, reglas y significados mediante un mecanismo de tipo interactivo, por lo que, si no reconoce algo, entra en un proceso de diálogo tanto con su exterior como con su interior, hasta que logra aprenderlo e integrarlo en su universo o hasta que se le indica que lo ignore.

Sin embargo, la semejanza no es fortuita, ya que según el Enfoque Lingüístico de la Informática tanto el compilador como el S.Ev. perciben la realidad en términos lingüísticos y así mientras el compilador se comunica con su entorno mediante un lenguaje que le fue predefinido por su constructor, en el caso del S.Ev., éste se comunica mediante un lenguaje que va adquiriendo a partir de la interacción con su entorno.

Uno de los enfoques “operativos” para la construcción de Sistemas Evolutivos parte de la capacidad de construir y mantener una imagen de la realidad de los S.Ev., pero permite que el sistema tenga un conocimiento mínimo inicial (una especie de código genético) sobre su realidad.

A este conocimiento mínimo se le conoce como Universo Lingüístico Básico y está formado generalmente por un conjunto muy pequeño de reglas predefinidas que permiten al sistema controlar su ambiente; estas reglas son el equivalente del lenguaje de máquina o de las funciones básicas de un lenguaje funcional como LISP.

Al integrar un Universo Lingüístico Básico a los sistemas evolutivos se rompe la posibilidad de que éstos entren en un ciclo de aprendizaje infinito, ya que normalmente una regla está sustentada en otras reglas y, en la búsqueda de la regla última, el sistema podría llegar a nivel de partículas elementales y no terminar; sin embargo, con el universo lingüístico básico el S.Ev. cuenta con un conjunto mínimo del cual partir y al cual llegar.

Por ejemplo, en el caso de que se construya un sistema evolutivo para lograr que un robot reconozca instrucciones en español, se puede partir de un Universo Lingüístico Básico formado por instrucciones simples como: que mueva el motor 1, tres pasos a la derecha, el motor 2, cinco a la izquierda, luego el motor 3 ocho a la derecha y finalmente que se detenga.

Ahora bien, a partir de un universo lingüístico formado por instrucciones tan simples como las anteriores, es factible construir un mecanismo que permite interpretar instrucciones en español, haciendo simplemente que cuando el sistema encuentre una palabra que no entiende pregunte su significado en término de las palabras que componen el universo lingüístico y aquellas palabras que no tengan significado sean ignoradas.

En aplicaciones tan simples como la anterior, el problema se reduce a un manejo léxico-semántico, por lo que un alumno de primeros semestres de licenciatura lo puede resolver; sin embargo, a partir de la misma idea se pueden construir sistemas evolutivos que asocien, por ejemplo, oraciones con estructura gramatical diferente, pero con el mismo significado. Por otro lado, en la actualidad existen sistemas evolutivos que no utilizan universos lingüísticos básicos y que encuentran el significado a partir de asociar, por ejemplo, imágenes con palabras. Aún más, existen sistemas evolutivos que ni siquiera utilizan el enfoque lingüístico, ya que el área es nueva y se presta a que diferentes personas propongan diferentes soluciones.

Conclusiones

El área de los sistemas evolutivos es un área relativamente nueva, por lo que es muy grato contar ya con una escuela consolidada sobre el tema, en la cual se desarrollan cotidianamente estos sistemas, ya que existen empresas que han desarrollado este tipo de herramientas para resolver problemas concretos como ORSA, TECCIZ y CELANESE MEXICANA, entre otras.

Por otro lado, esta área ya trascendió las fronteras y se han presentado resultados en foros internacionales realizados en Venezuela, Francia y vía satélite se ha difundido, por parte del Tecnológico de Monterrey, a toda América.

Cada vez estamos más convencidos que los sistemas evolutivos marcan realmente un cambio de paradigma en el desarrollo de la Informática y estamos orgullosos de que los conceptos básicos y primeros productos sean el resultado del esfuerzo de múltiples investigadores mexicanos.

El desarrollo de un área requiere tanto de aspectos técnicos como administrativos, ya que una cosa es que una persona sepa construirla, y otra cosa es lograr que la construcción, aplicación e investigación se vuelva cotidiana en una comunidad y trascienda los límites locales, nacionales e internacionales, hasta que se vuelva una línea de investigación competitiva y relevante en el contexto mundial, por lo que considero que las acciones emprendidas nos llevan en esa dirección y ya podemos hablar de un área con métodos, técnicas y resultados propios y que está pasando a un nivel industrial.

CAPÍTULO III
APLICACIÓN DE LA LINGÜÍSTICA MATEMÁTICA
A LOS SISTEMAS EVOLUTIVOS

III.1 Enfoque Lingüístico

Fernando Galindo Soria¹

Resumen

En este trabajo se presenta el Enfoque Lingüístico como una nueva forma de ver la realidad, en la cual se considera que, cualquier cosa se puede ver como una oración de algún lenguaje. Mediante este enfoque se pueden representar como oraciones de algún lenguaje las imágenes, las reglas de un sistema experto, las trayectorias de un planeta, el movimiento de la mano, la trayectoria que sigue al moverse una pieza de ajedrez, una huella digital, la señal de un electrocardiograma, etc., con lo que, se amplía el concepto de límite que normalmente se restringe a los lenguajes naturales (como el Español, Inglés, Chino o Árabe) y artificiales (como Fortran, Pascal o C), para incluir cualquier cosa.

Como primer punto se comenta que el Enfoque Lingüístico surgió en México a mediados de los 70's y que tiene como antecedentes los trabajos desarrollados por Noam Chomsky a mediados de los 50's para representar la estructura de los lenguajes naturales, la aplicación de estos trabajos a la construcción de compiladores, con lo que se generalizó el concepto de lenguaje para incluir a los lenguajes artificiales de programación y el surgimiento del Reconocimiento Sintáctico de Formas, donde se plantea que la forma de cualquier objeto o el patrón de comportamiento de cualquier proceso se puede ver como una oración de algún lenguaje, con lo que se generalizó aún más el concepto de lenguaje para incluir a cualquier patrón o forma, de donde se llegó finalmente a la idea del Enfoque Lingüístico.

Como siguiente punto se generaliza el concepto de unidad léxica para incluir a cualquier cosa que se pueda percibir o conceptualizar.

Se presentan múltiples tipos de lenguajes, como: lenguajes naturales restringidos, lenguajes de trayectoria, lenguajes para representar sistemas de diagnóstico y toma de decisiones y una ecuación lingüística con la cual se tiene la estructura de cualquier elemento de la naturaleza.

Más adelante se ve que la Gramática (o conjunto de reglas que representan a un lenguaje) es equivalente a un sistema de información, por lo que si se tienen un conjunto de oraciones de un lenguaje se puede encontrar su gramática y de ahí obtener el sistema que trabaja con ese lenguaje.

¹ Fernando Galindo Soria, Instituto Politécnico Nacional, UPIICSA, ESCOM.

Finalmente se muestra que las oraciones tienen una estructura fractal ya que, según el enfoque lingüístico cada una de las unidades léxicas que componen a la oración se pueden ver a su vez como oraciones.

Introducción

En este trabajo se presenta el Enfoque Lingüístico como una nueva forma de ver la realidad, en la cual se considera que, cualquier cosa que se pueda percibir o conceptualizar se puede representar como una oración de algún lenguaje.

Mediante este enfoque se pueden ver como oraciones de algún lenguaje las imágenes, las reglas de un sistema experto, la trayectoria de un planeta, el movimiento de la mano, la trayectoria que sigue al moverse una pieza de ajedrez, una huella digital, la señal de un electrocardiograma, etc., y se amplía el concepto de lenguaje que normalmente se restringía a los lenguajes naturales (como el Español, Inglés, Chino o Árabe) y artificiales (como Fortran, Pascal o C), para incluir cualquier cosa.

Aún mas, en general se define un lenguaje como un conjunto, por lo que cualquier conjunto se puede ver como un lenguaje y los elementos del conjunto como oraciones del lenguaje.

I Antecedentes del Enfoque Lingüístico

El Enfoque Lingüístico surgió como resultado de múltiples trabajos realizados desde mediados de los 70's principalmente en El Colegio de México, la Escuela Superior de Física y Matemáticas (ESFM), el Centro Nacional de Cálculo (CENAC) y la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA), estos tres últimos del Instituto Politécnico Nacional (IPN).

Empero sus primeros antecedentes se encuentran en los trabajos desarrollados por Noam Chomsky para representar la estructura de los lenguajes naturales y plasmados en su libro Syntactic Structures, publicado en 1957, en la aplicación de estos trabajos a la construcción de compiladores y en el surgimiento del Reconocimiento Sintáctico de Formas.

A principios de los 60's los trabajos de Chomsky se aplicaron para describir la gramática del lenguaje Algol propiciando que ya para finales de esa décadas empezaran a surgir libros donde se mostraba como construir un compilador a partir de la gramática de un lenguaje dado, **con lo cual se generalizó el concepto de lenguaje para incluir a los lenguajes artificiales de programación.**

En paralelo con lo anterior y también desde mediados de los 60's se empezó a aplicar la Lingüística Matemática al Reconocimiento de Patrones (o sea al estudio de la forma o patrón de comportamiento asociado con algún objeto o proceso) siendo la base de los métodos de **Reconocimiento Sintáctico de Formas** en los cuales se plantea que **la forma de cualquier objeto o el patrón de comportamiento de cualquier proceso se puede ver como una oración de algún lenguaje**, por ejemplo se ve a una imagen o proceso (voz, movimiento de los planetas, jugada

de ajedrez, etc.) como una oración del lenguaje de imágenes o de procesos, **con lo que se generalizó aún más el concepto de lenguaje para incluir a cualquier patrón o forma.**

A finales de los 60's **Rafael C. González y Michael C. Thomason** escribieron el libro **Syntactic Pattern Recognition** donde ya se presenta en una forma elaborada el concepto de reconocimiento sintáctico de patrones y aparece **claramente la idea de representar patrones como oraciones de algún lenguaje, con lo que se extiende el concepto de lenguaje** para incluir figuras geométricas, dibujos e imágenes en general, además de la forma o patrón de cualquier objeto o proceso, por lo que **podemos considerar que este libro es el precursor del enfoque lingüístico.**

Durante los 70's se continuaron realizando aplicaciones de las herramientas lingüísticas a la construcción de compiladores y al reconocimiento de formas y se extendió su uso a otras áreas de la Informática como la complejidad de algoritmos y la construcción de sistemas operativos, manejadores de bases de datos y editores.

II Resultados Generales

En particular desde mediados de los 70's en el IPN se ha desarrollado el Enfoque Lingüístico y en la actualidad ya se cuenta con una gran cantidad de resultados y aplicaciones, por lo que en los siguientes puntos se presentaran algunos de éstos.

1 Conceptos generales

En general se define un lenguaje como un conjunto, por lo que cualquier conjunto se puede ver como un lenguaje y los elementos del conjunto como oraciones del lenguaje.

Mas específicamente cualquier mecanismo que tenga un conjunto de elementos o palabras (alfabeto) sobre los que se pueda aplicar un conjunto de reglas para relacionarlos (syntaxis) y asociarle un significado (semántica) se puede decir que cuenta con un lenguaje.

En particular una oración está formada por palabras (o unidades léxicas) relacionadas entre si (o syntaxis) y con un cierto significado (o semántica).

2 Percepción y Conceptualización de Unidades Léxicas

Generalmente sólo se aceptan como unidades léxicas a los componentes de las oraciones de los lenguajes naturales, y desde el surgimiento de los lenguajes de programación se ha ampliado el término para abarcarlos también a ellos, sin embargo aún en la actualidad no se incluyen dentro de este concepto por ejemplo a las partes de un gato o a las ramas de un árbol o a las señales que emite una neurona, y el estudio de estos elementos se incluye como problemas independientes dentro de la Semiótica. Por lo que, lo primero que se tiene que hacer es **generalizar el concepto de unidad léxica.**

En el enfoque lingüístico se generaliza el concepto de unidad léxica a cualquier cosa que se pueda percibir o conceptualizar. Por ejemplo la imagen que percibimos de un gato se puede considerar como la oración que representa al gato, y las diferentes partes del gato equivalen a las palabras o unidades léxicas que componen a la oración.

3 Ejemplos de Lenguajes

1) *Lenguaje Natural Restringido*: En este tipo de lenguajes se incluyen los lenguajes imperativos formados por oraciones en las que se plantea algún requerimiento u orden, los lenguajes declarativos que se centran en el planteamiento de hechos o reglas de inferencia, los lenguajes interrogativos mediante los que se plantean preguntas sobre el sistema y sus combinaciones.

Como posibles ejemplos de oraciones de lenguaje se tienen: en diagnóstico médico; la conversación del paciente con el médico, en la nómina, la descripción escrita de los cambios o modificaciones requeridas; en pruebas de personalidad, la oración está formada por una cadena de letras, donde cada letra representa la respuesta a una pregunta de la prueba (normalmente tienen un máximo de cinco posibles respuestas) y la cadena representa todas las respuestas dadas por una persona, de donde una oración típica se parece a una cadena binaria.

2) *Lenguajes de Trayectoria*: Estamos acostumbrados a manejar el concepto de lenguaje como sinónimo de lenguaje natural escrito o hablado y si nos fuerzan mucho podemos aceptar la idea de lenguaje de señas o movimientos. Sin embargo, atrás de todos estos tipos de lenguajes podemos encontrar un concepto más general que los incluye y generaliza conocido como lenguaje de trayectoria en el cual se incluyen todos los lenguajes visuales (gráficas), de movimiento, (jugadas en un tablero de ajedrez, movimiento de un robot, etc.) y cualquier otro lenguaje que se pueda representar por una trayectoria (sonido, recorrido por una ciudad, etc.).

Un Lenguaje de Trayectoria se puede conceptualizar como un conjunto de oraciones que representan la trayectoria espacial o temporal entre dos puntos para lo cual en lugar de lexemas o fonemas utiliza como unidades básicas símbolos que representan trayectorias.

Prácticamente todos los elementos de los lenguajes tradicionales (natural escrito y hablado, señas, etc.) se pueden ver como una combinación de trayectorias ya que al final de cuentas lo que se representa mediante una letra es el resultado de la trayectoria que sigue la pluma de un punto a otro, o lo que se escucha como un fonema es la abstracción de la trayectoria que sigue una onda sinusoidal y existen múltiples fenómenos que captamos como resultado de una trayectoria en alguna dimensión entera o fractal (el movimiento de los planetas, el movimiento de un pie, el paso del tiempo, el crecimiento de una célula, etc.), por lo que en su momento a todos estos fenómenos se les puede asociar un lenguaje de trayectoria y podemos postular que cuando se capta algún fenómeno en primera instancia estamos captando oraciones de algún lenguaje de trayectoria las cuales al ser absorbidas y procesadas por nuestros sentidos son integradas mediante un patrón general al cual se le asocia algún tipo de significado.

3) *Sistemas de diagnóstico y toma de decisiones*: Una de las características de los sistemas de diagnóstico y toma de decisiones (DSS, reconocedores de patrones, etc.) en general y de los sistemas expertos en particular se encuentra en que el lenguaje del usuario consta principalmente de oraciones de la forma:

SÍNTOMA : DIAGNÓSTICO : ACCIONES o TRATAMIENTO

donde este tipo de oraciones nos permiten plantear por ejemplo, la base de un conjunto de reglas de inferencia o el patrón general de un cierto tipo de figuras.

Por ejemplo si se tiene la oración:

Paciente femenino de 15 años con 38 grados de temperatura y

s1	s2	s3
<u>dolor en el pecho</u> , se le diagnosticó <u>laringitis</u> y se le recetó <u>antibióticos</u> ,		
s4	d1	t1
<u>antihistamínicos</u> y <u>reposo</u>		
t2	t3	

se detecta que el paciente tiene los síntomas: s1 s2 s3 s4, el diagnostico: d1 y requiere los tratamientos: t1 t2 t3

4) *Ecuación fundamental de la naturaleza*: Desde mediados de los 70's se ha utilizado la lingüística y en particular los sistemas-L para representar la estructura de árboles y construir programas de graficación, siguiendo con esa línea y combinando los conceptos de fractales y el Enfoque Lingüístico a finales de los 80's era relativamente fácil representar mediante gramáticas la estructura de múltiples objetos de la naturaleza como árboles, nubes y montañas, sin embargo cada tipo de objeto se representaba con una gramática diferente a partir de la premisa de que cada objeto tenía una estructura diferente de los otros. Mas adelante se encontró que todos esos objetos se podían representar con una única ecuación lingüística de la forma $S \rightarrow a^*S^*$ con lo cual prácticamente **se tiene en una sola ecuación la estructura de cualquier elemento de la naturaleza**.

4 Del Lenguaje al Sistema

El concepto de lenguaje es muy amplio y es importante saber elegir el que se adecue más a nuestro problema. Ya que se eligió un lenguaje X, el siguiente paso consiste en captar muchas oraciones (grabando, observándolo, sintiéndolo, etc.).

Ahora bien, si se tuviera que tener la lista de todas las oraciones de un lenguaje no terminaríamos, por lo que comúnmente en lugar de la lista de oraciones se utiliza una gramática o conjunto de reglas que representan al lenguaje.

La gramática que representa al lenguaje es equivalente a un sistema de información, por lo que si se tienen un conjunto de oraciones de un lenguaje se puede encontrar su gramática y de ahí obtener el sistema que trabaja con ese lenguaje.

5 Estructura Fractal de las Oraciones

Un resultado que se encontró junto con Juan Martín González Vázquez en 1984, fue que las oraciones tienen una estructura fractal, ya que, según el enfoque lingüístico cada una de las unidades léxicas que componen a la oración se pueden ver a su vez como oraciones.

Por ejemplo en la oración siguiente

El perro mordió al gato

Cada una de las palabras puede ser una oración y entonces la palabra

perro

es por si sola una oración compuesta por sus propias unidades léxicas, que dependiendo de la aplicación pueden ser, por ejemplo:

- 1) **pe rro**, para un analizador silábico
- 2) **p e r r o**, para un programa de ordenamiento de palabras.
- 3) A su vez cada letra se puede ver como una oración de otro lenguaje, por ejemplo la letra:

p

será una oración donde las unidades léxicas pueden ser los diferentes trazos que la forman.

Conclusiones

En este documento se mostró como el estudio de los lenguajes se ha ampliado desde la idea original del tratamiento de los lenguajes naturales, su ampliación para incluir a los lenguajes artificiales de programación y su generalización para incluir a cualquier patrón o forma, de donde se llegó finalmente a la idea del **Enfoque Lingüístico que plantea que cualquier cosa se puede ver como una oración de algún lenguaje**. Además se mostraron algunas de sus aplicaciones con el fin de que se vea que es un concepto general y que se encuentra en el núcleo de las herramientas que nos permiten estudiar la realidad.

Bibliografía

- [1] Przemyslaw Prosinkewicz, Aristid Lindenmayer, James Hanan. Developmental Models of Herbaceous Plants for Computer Imagery Purposes, *Computer Graphics*, Vol 22(4), Agosto 1988.
- [2] Sofia Bueno Peralta y Antonio Simancas López. Generador de Arboles Fractales en Memorias del III Congreso Nacional sobre Informática y Computación, Jalapa, Ver. México, Octubre 1990.
- [3] Heinz-Otto Peitgen and Dietmar Saupe (editores). *The Science of Fractal Images*, ed. Springer-Verlag, 1988.
- [4] Femando Galindo Soria, *Sistemas Evolutivos: Nuevo Paradigma de la Infomática en las Memorias de la XVII Conferencia Latinoamericana de Informática*, Caracas Venezuela, julio de 1991.
- [5] Femando Galindo Soria, *Sistemas Evolutivos en Boletín de Política Informática*, México, Septiembre de 1986.
- [6] Femando Galindo Soria. *Aplicaciones de la Lingüística Matemática y los Fractales a la Generación de Imágenes en Memorias Simposium Nacional de Computación*, México, noviembre de 1991.
- [7] Rafael C. González y Michael C. Thomason, *Syntactic Pattern Recognition*, ed. Addison-Wesley.
- [8] Emmon Bach, *Teoría Sintáctica*, ed. Anagrama.
- [9] Salomaa, *Formal Languages*, ed. Academic Press.
- [10] Herbert A. Simon, *Las Ciencias de lo Artificial*, ed. ATE.
- [11] Noam Chomsky, *Estructuras Sintácticas*, ed. Siglo XXI.
- [12] Hopcroft y Ullman, *Formal Languages and their Relation to Autómata*, ed. Addison-Wesley.
- [13] Aho y Ullman, *Principles of Compiler Design*, ed. Addison-Wesley.

III.2 Programación Dirigida por Sintaxis (un método de desarrollo de sistemas basado en el lenguaje del usuario)

Fernando Galindo Soria²

Resumen

La Programación Dirigida por Sintaxis es un método de desarrollo basado en la Lingüística Matemática y orientado a facilitar la comunicación hombre-máquina.

Este método tiene la característica de permitir desarrollar el sistema de información a partir del lenguaje del usuario, por lo que, el manejo de requerimientos se realiza en un “lenguaje natural restringido” parecido al que utiliza el usuario en su comunicación dentro del área problema.

La idea original partió de los métodos de construcción de compiladores con los cuales a partir de la gramática de un lenguaje se puede construir un traductor capaz de reconocer instrucciones en ese lenguaje, por lo que originalmente se construyeron reconocedores que tenían como entrada la gramática de un lenguaje muy restringido (tipo FORTRAN, BASIC ó PASCAL) y que interpretaba instrucciones de éste lenguaje. Más adelante se amplió el concepto de lenguaje y se construyeron traductores de lenguajes orientado a base de datos, sistemas operativos y validación de datos entre otros, para lo cual únicamente se cambió la gramática, ya que el reconocedor siguió siendo el mismo, por lo que se vio que ahora el problema principal era encontrar la gramática representativa de un lenguaje dado, ya que contando con la gramática el proceso de reconocimiento se tenía resuelto (y en realidad existen muchísimos métodos de reconocimiento de un lenguaje a partir de su gramática en los libros de construcción de compiladores), por lo que se buscaron y desarrollaron métodos de inferencia gramatical con los cuales se puede encontrar la gramática de un lenguaje a partir de ejemplos de éste. (Ejemplos de estos métodos aparecen en trabajos sobre reconocimiento sintáctico de patrones), más adelante se reordenaron éstos métodos para que quedaran en una forma lógica, en la cual primero se estudia el área y se encuentran ejemplos significativos del lenguaje (Análisis), más adelante se obtiene una gramática con atributos que representa el lenguaje (Diseño) y finalmente se construye el reconocedor del lenguaje (Implantación).

² Fernando Galindo Soria, enero 1988

Introducción

En la actualidad se considera que el proceso de razonamiento se desarrolla principalmente mediante métodos deductivos e inductivos; en los **métodos deductivos** se aplica un conjunto de **reglas preestablecidas** para encontrar la solución de un problema dado y por su parte los **métodos inductivos** buscan **obtener precisamente reglas generales o patrones** a partir de casos particulares, de donde se ve que son métodos complementarios.

Sin embargo, en la actualidad, prácticamente todas las formas de programación que se utilizan se orientan al manejo de **sistemas de tipo deductivo** (o sea que aplican un conjunto de reglas preestablecidas para resolver un problema) y esto ocurre independientemente de la aplicación (nomina, manejador de base de datos, sistema experto, simulador de juegos, etc.) y del lenguaje o herramienta automatizada que se utilice (COBOL, FORTRAN, Pascal, LISP, PROLOG, Lotus, etc.), por lo que el informático o programador tiene que desarrollar la parte inductiva del problema a mano (lo cual ha propiciado el surgimiento de métodos para encontrar reglas generales para resolver problemas como son: el Desarrollo de Sistemas, Algorítmica, Ingeniería de Software, Ingeniería de Conocimientos, etc.).

Por esto surge la idea de desarrollar y en su caso automatizar métodos de **Programación de tipo Inductivo** mediante los cuales sea relativamente fácil encontrar el conjunto de reglas generales de un problema a partir de casos particulares.

En éste documento se presenta un método de programación de tipo inductivo basado en la Teoría de Lenguajes y en los métodos de Reconocimiento de Patrones (o formas) y orientados a facilitar la comunicación entre el usuario y la computadora en un lenguaje lo más natural posible y al manejo por parte de la computadora y de formas ó patrones más que de datos o conocimientos específicos.

Este método tiene la característica de permitir desarrollar el sistema de información a partir del lenguaje del usuario. Por lo que, el manejo de requerimientos se realiza en un “lenguaje natural restringido” parecido al que utiliza el usuario en su comunicación dentro del área problema. A pesar de que tradicionalmente en la literatura se presenta el problema de manejo del lenguaje natural como un problema de alta complejidad y que por tal motivo sólo puede ser atacado por personas de muy alto nivel y no es factible su manejo por la generalidad de los informáticos, sin embargo, se ha detectado que éste planteamiento es erróneo y que por el contrario, atacar los problemas informáticos a base del “lenguaje natural restringido” no sólo es más sencillo que los enfoques tradicionales sino que además permite resolver los problemas de una forma más completa, ya que al encadenarse con los métodos de reconocimiento sintáctico patrones se tiene un doble mecanismo en el cual se parte de múltiples ejemplos del lenguaje del usuario, se puede encontrar una gramática en la cual se encuentra precisamente representada la forma del lenguaje, y a partir de esta gramática es relativamente fácil construir por ejemplo las instrucciones de un sistema de información o el conjunto de reglas de inferencia de un sistema experto, de donde el sistema experto, el sistema de información se transforma en la parte deductiva de un sistema mucho más poderoso.

La idea de éste método surge de los métodos de construcción de compiladores con los cuales a partir de la gramática de un lenguaje se puede construir un traductor capaz de reconocer instrucciones de ese lenguaje, por lo que originalmente se construyeron reconocedores que tenían como entrada la gramática de un lenguaje muy restringido (tipo FORTRAN, BASIC ó Pascal) y que interpretan instrucciones de éste lenguaje. Más adelante se amplió el concepto de lenguaje y se construyeron traductores de lenguajes orientados a base de datos, sistemas operativos y validación de datos, entre otros. Para lo cual únicamente se cambió la gramática, ya que el reconocedor siguió siendo el mismo, por lo que se vio que ahora el problema principal era el de encontrar la gramática representativa de un lenguaje dado, ya que contando con la gramática el proceso de reconocimiento se tenía resuelto (y en realidad existen muchísimos métodos de reconocimiento de un lenguaje a partir de su gramática en los libros de construcción de compiladores) por lo que se buscaron y desarrollaron métodos de inferencia gramatical, con los cuales se puede encontrar la gramática de un lenguaje a partir de ejemplos de éste. (ejemplos de éstos métodos aparecen en trabajos sobre reconocimiento sintáctico de patrones), más adelante se ordenaron estos métodos para que quedaran en una forma lógica y al método resultante se le llamó Programación Dirigida por Sintaxis.

En la actualidad éste método se ha utilizado para desarrollar sistemas en áreas tan disímiles como construcción de compiladores, sistemas operativos, editores, bases de datos, comunicación de datos, seguridad, etc.; lenguajes para paquetes estadísticos, de validación de datos, de control de paquetes de graficación y simuladores de robots; y aplicando el concepto de lenguaje en áreas no tradicionales se han utilizado para enseñar a la computadora a reconocer reglas ortográficas, análisis de contenido, reconocimiento de imágenes, aprendizaje de juegos (viendo las imágenes y las jugadas como ejemplos de oraciones y para desarrollar modelos neurales y secuencias lógicas de actividades) por lo que como se observa, el potencial de ésta herramienta es muy grande, ya que en contra de lo que se maneja tradicionalmente, es relativamente sencillo construir sistemas capaces de reconocer un “lenguaje natural restringido”, por lo que ésta herramienta ha trascendido el área académica se utiliza actualmente, tanto en empresas privadas como en diversas áreas del sector público, con el fin de desarrollar sus sistemas con una orientación mayor hacia el usuario, por lo que, uno de los objetivos de éste trabajo es el de difundir éste tipo de métodos y lograr que un grupo mayor de la comunidad Inteligencia Artificial (I.A.) se involucre en su utilización y principalmente en su estudio, ya que es un área nueva y existen múltiples temas abiertos de investigación, como por ejemplo:

- 1) *Ampliación del concepto de lenguaje.*
- 2) *Inferencia Gramatical.*
- 3) *Automatización de las diferentes etapas del método.*
- 4) *Construcción de Sistemas Evolutivos.*

Con éste documento se pretende entre otras cosas:

- 1) *Mostrar un método de programación no tradicional.*
- 2) *Lograr que los sistemas de información tengan comunicación más orientada al usuario.*
- 3) *Mostrar que los métodos de construcción de compiladores no son exclusivos de los especialistas en esa área, sino que lo mismo se puede aplicar para construir un compilador,*

un sistema manejador de base de datos, un sistema operativo, una nómina ó sistema de inventarios, con lo cual se ve en un momento dado el hecho de tener diferentes aplicaciones no implica que no pueda usar la misma herramienta.

4) *Presentar un proceso integrado para resolver problemas* y que actualmente no se maneja de ésta forma (por ejemplo en los cursos de compiladores es común que se indique como obtener el lenguaje ó a partir de éste cómo obtener la gramática).

5) *Mostrar la gran importancia que tiene la teoría de gramáticas* (en general los sistemas formales) como herramientas para el desarrollo de sistemas.

6) *Mostrar la gran importancia que tiene la Teoría de Reconocimiento de Formas* (en particular el Reconocimiento Sintáctico de Patrones) como herramientas para el desarrollo de sistemas y mostrar como su integración con la Teoría de Gramáticas proporciona una herramienta aplicable en diversas aplicaciones.

I Conceptos Generales

Como se ha comentado anteriormente éste método se basa principalmente en herramientas de Lingüística Matemática y de Reconocimiento Sintáctico de Patrones y en particular en las Gramáticas Libres de Contexto (o tipo 2 de la Jerarquía de Chomsky) por lo que como primer punto se presentará en forma general éste concepto y más adelante se mostrará su ámbito de aplicación.

1 Definición de Gramáticas Libre de Contexto

Una Gramática Libre de Contexto es una herramienta, con la cual se puede describir la estructura de las oraciones de una lengua, ésta herramienta consiste básicamente en cuatro componentes:

$G = \langle V_n, V_t, S, P \rangle$ donde:

1) V_n es un conjunto de elementos o variables no terminales que sirven para representar los cambios de un estado a otro, normalmente las variables no terminales se denotan con letras mayúsculas.

2) V_t es un conjunto de elementos o variables terminales que equivalen a las señales u órdenes sobre el sistema, normalmente se denotan con letra minúscula.

3) Se tiene un elemento no terminal que indica el estado inicial o el axioma del sistema y se denota como S .

4) P es un conjunto de reglas con las cuales se describe la estructura del problema y son de la forma:

$A \rightarrow \alpha$

donde A es algún estado e indica que esa regla sólo se aplica si se encuentra el problema en el estado A y α es una cadena de estados y señales.

P es un conjunto de relaciones o reglas de producción con dominio y contradominio en V^* ($V = V_n \cup V_t$). V^* es el conjunto de todas las posibles cadenas formadas por la concatenación de elementos de V . Si $\alpha \in V^*$ se dice que α es una cadena de elementos.

Ejemplo: El conjunto de reglas:

$$\begin{aligned} S &\rightarrow v A \\ A &\rightarrow + v A \\ A &\rightarrow ; \end{aligned}$$

Indican que si se está en el estado S entonces se debe verificar que llegue la señal v , si llega v entonces se pasa al estado A . En el caso del estado A se tienen dos posibles caminos: si llega la señal $+$ se toma uno y si llega la señal $;$ se toma el otro. Por simplicidad cuando existen varios caminos se puede denotar como:

$$A \rightarrow + v A \mid ;$$

donde el símbolo \mid indica camino alternativo.

Si no llega alguna de las señales esperadas entonces se detecta que ese no era un camino válido y se regresa al estado anterior a buscar otro camino, si ya no existe ningún camino posible entonces la señal es errónea.

En éste ejemplo en particular las cadenas de señales válidas es de la forma:

$$\begin{aligned} &v ; \\ &v + v ; \\ &v + v + v ; \\ &\dots \\ &v + v + \dots + v ; \end{aligned}$$

O sea ésta gramática espera señales que indiquen sumas de variables.

2 Equivalencia Estructural entre Gramáticas y Sistemas de Información

Si partimos de que un lenguaje es un mecanismo con el que se puede representar y transmitir el conocimiento y observamos el lenguaje que se utiliza en algún área problema podemos encontrar que en éste se encuentran en forma natural los componentes de un sistema de información, **ya que podemos encontrar un conjunto de acciones que se refieren a ciertos objetos en un orden dado**, donde las acciones del lenguaje se pueden representar como acciones en el sistema; a partir de los objetos se pueden obtener las entidades, atributos y valores de la estructura de datos y finalmente a partir del orden en que aparecen las acciones y objetos en la oración se puede encontrar el orden o estructura de control del Sistema de Información.

En particular, mediante la gramática que representa el lenguaje se puede representar la estructura del Sistema de Información. Esta posibilidad de representar un sistema como una gramática permite por ejemplo obtener la gramática del Sistema de Información y estudiar la estructura del sistema y en su momento encontrar una nueva gramática que realice lo mismo (gramática equivalente) y a partir de ésta proponer un mejor sistema.

Por lo que a continuación se verá que cualquier sistema, programa o estructura de datos es estructuralmente (también se le dice débilmente) equivalente a una gramática y como esto nos permite utilizar las gramáticas como un mecanismo para representar sistemas (como los diagramas de Warnier, de Flujo, de Flujo de Datos, etc.)

2.1 Equivalencia de Gramáticas y Programas

Como primer punto, se verá que la estructura de un programa se puede representar con una gramática y viceversa.

La equivalencia entre una gramática y un programa se puede observar si se toma un programa típico.

```
Programa típico
  Rutina A
  Ejecuta B
  Si C entonces D sino E
  Mientras F ejecuta G
  Llama a D
  Fin rutina
```

ya que en éste se pueden encontrar cuatro estructuras de control básicas:

- 1) *Sentencias.*
- 2) *Interacciones.*
- 3) *Decisiones.*
- 4) *Recursión.*

y cada una de estas puede representarlas en abstracto mediante una gramática:

- 1) *Sentencias:* $A \rightarrow B$
- 2) *Iteraciones:* $A \rightarrow E^*$
- 3) *Decisiones:* $A \rightarrow \{C \mid D\}$
- 4) *Recursiones:* $A \rightarrow d A$

Por lo que una rutina de la forma:

```
Rutina A
  B
  D ó E
  (G)*
Fin rutina
```

se puede representar mediante la siguiente producción:

```
A → B {C | D} G*
o de otra forma
A → B X Y
X → D | E
Y → G*
```

Por otro lado, a partir de la gramática se puede encontrar el programa, ya que:

1) $S \rightarrow D$

equivale a

```
Rutina S
  llama a D
Fin rutina
```

2) $S \rightarrow A B C$

equivale a

```
Rutina S
  llama a A
  llama a B
  llama a C
Fin rutina
```

3) $S \rightarrow a A$

equivale a

```
Rutina S
  Si a entonces llama a A
Fin rutina
```


$$4) S \rightarrow a A \mid b B$$

equivale a

Rutina S

Si a entonces llama a A

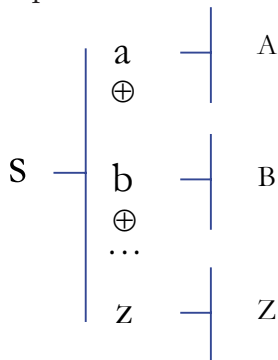
sino Si b entonces llama a B

Fin si

Fin rutina

$$5) S \rightarrow a A \mid b B \mid c C \mid \dots \mid z Z$$

equivale a



Cualquier programa es equivalente estructuralmente a una gramática por lo que se puede tomar un programa, encontrar su gramática y detectar:

- 1) El tipo de lenguaje que tiene.
- 2) Problemas estructurales.
- 3) Oraciones que no reconoce.
- 4) etc., etc.

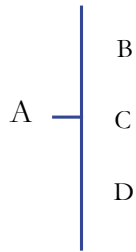
Se puede tomar una gramática y construir un programa que la representa, con lo que se tendría el Sistema de Información.

2.2 Equivalencia entre Gramáticas y Diagramas de Warnier

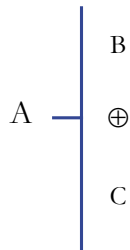
Los diagramas de Warnier son herramientas para representar sistemas, ya que absorben las características de los diagramas Top/Down y de la programación estructurada con la ventaja de ser sencillos de manejar y con ellos se puede representar fácilmente, por ejemplo, Sistemas de Información, Estructuras de Datos y programas de tipo administrativo.

En estos diagramas se permite:

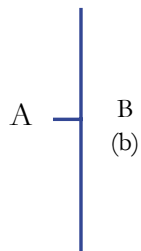
- 1) *La secuencia.* La rutina A se compone de B, C y D



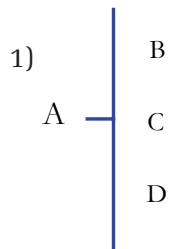
2) *La decisión.* En la rutina A se ejecuta B ó C



3) *La iteración.* En la rutina A se repite el proceso B, b veces

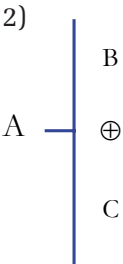


Dado que es muy fácil obtener los programas y el sistema de información del diagrama de War-nier, este es de aplicación general, por lo que veremos que es equivalente a una gramática.



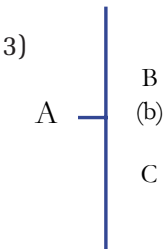
equivale a:

$A \rightarrow B C D$



equivale a:

$$A \rightarrow B \mid C$$



equivale a:

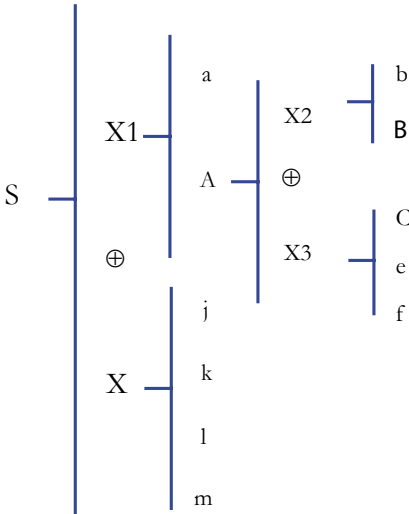
$$A \rightarrow \{B\}^b C$$

como no es de uso común el exponente se puede sustituir por una gramática recursiva:

$$\begin{aligned} A &\rightarrow B X \\ X &\rightarrow B X \mid C \end{aligned}$$

El proceso de equivalencia es el mismo por lo que por ejemplo de la gramática, se puede obtener un Diagrama Warnier:

$$\begin{aligned} S &\rightarrow a A \mid X \\ A &\rightarrow b B \mid C e f \\ X &\rightarrow j k l m \end{aligned}$$



2 Método de Programación dirigida por Sintaxis

El hecho de que sea relativamente fácil representar un sistema de información como una gramática y que por el otro lado la gramática represente los patrones o reglas generales de algún lenguaje plantea la posibilidad de contar con un mecanismo que nos permita obtener a partir del lenguaje la estructura de un sistema de información y de ahí obtener el sistema.

A continuación describiremos el método de Programación Dirigida por Sintaxis y en los siguientes puntos se presentará a detalle cada paso con el fin de mostrar su uso y algunas de sus posibilidades.

El método en general consta de una fase de Análisis, en la cual se estudia el área significativa y se encuentran ejemplos significativos del lenguaje; otra de Diseño en la cual se encuentra una gramática con atributos que representa los patrones del comportamiento del lenguaje y la estructura del sistema y finalmente una fase de Implantación, donde se desarrollan las herramientas para reconocer el lenguaje.

Detallando un poco, el método consta de las siguientes partes:

1) *Análisis:*

- i) Detección de un área de aplicación o campo problema.
- ii) Estudio del lenguaje del área y detección de ejemplos significativos del lenguaje.

2) *Diseño:*

- i) Detección de las unidades léxicas del enunciado.
- ii) Obtención de una Gramática Generativa Canónica, mediante la sustitución de cada una de las unidades léxicas en las oraciones por un símbolo que indica su tipo de unidad léxica e introduciendo un axioma que genere las oraciones.
- iii) Obtención de una Gramática Generalizada mediante Técnicas de Inferencia Gramatical.
- iv) Obtención de la Gramática con Atributos mediante la inclusión de rutinas semánticas.
- v) Detección de problemas estructurales y obtención de una gramática determinística (si existe) o de una gramática con pocos problemas.

3) *Implantación:*

- i) Construcción de una Tabla Gramatical en la que se representa la estructura del sistema a partir de la gramática.
- ii) Construcción o uso de algún Reconocedor de Lenguaje que recibe por un lado, la gramática de lenguaje (que representa los patrones de comportamiento del sistema) y por otro los requerimientos del usuario en un lenguaje natural, orientado a la aplicación y generando respuestas a los requerimientos.

III Estudio del Lenguaje y Detección de Ejemplos Significativos

Dado que se tiene un área de aplicación o algún problema a resolver, el primer punto importante consiste en la detección del lenguaje involucrado.

Tradicionalmente el concepto de lenguaje está muy restringido y por lo común o se piensa en instrucciones muy restringidas (tipo FORTRAN, Pascal, PROLOG, JCL, QUERY, etc.) o si bien va en términos de “Lenguaje Natural Restringido”, con lo que se quiere indicar algún subconjunto del Español escrito, sin embargo el concepto de lenguaje es mucho más amplio, ya que cualquier mecanismo en el que se encuentre un conjunto de elementos (alfabeto) sobre los que se puede aplicar un conjunto de reglas para relacionarlas (sintaxis) y asociarle un significado (semántica) se puede decir que se cuenta con un lenguaje, en su momento puede existir lenguajes de múltiples tipos: natural escrito, simbólico, hablado, visual, etc.; por lo que cuando se desarrolla un sistema mediante éste método, el universo de posibles mecanismos de comunicación se amplió enormemente y en su momento es un factor a tomar en cuenta, ya que sería poco adecuado restringirse a lenguajes tradicionales (menú, ensamblador, pseudocódigo, natural restringido, etc.) si existe algún otro lenguaje más adecuado.

- 1) En *diagnóstico médico*, la conversación del paciente con el médico.
- 2) En la *nómina*, la descripción escrita de los cambios o modificaciones requeridas.
- 3) En *visión*, la imagen a reconocer, la cual por ejemplo, se puede transformar en una oración tradicional mediante la aplicación de las técnicas de las primeras vecinas y número de forma.
- 4) *Juegos de Tablero*, la trayectoria de la pieza, en éste caso la idea es que la trayectoria de una pieza en el tablero se puede ver como una imagen visual y por tanto aplicarle las herramientas de 3)
- 5) *Características Especiales*, en éste caso el concepto de lenguaje pasa a otro nivel, ya que lo que se pretende es representar conceptos no tangibles como arriba, abajo, al lado, etc. En éste caso se requiere la confluencia de varios mecanismos de captación de información que la integran en una sola oración, por ejemplo (mecanismo visual, mecanismo especial y mecanismo escrito).
- 6) *Atributos Generales*, partiendo de la idea anterior del uso de varios mecanismos de captación que integran en una sola oración se pueden captar características como: más claro, más oscuro, verde, azul, curvo, etc.
- 7) *Características Temporales*, se puede integrar en una sola opción, eventos ocurridos en diferentes tiempos, (introduciendo un operador temporal⬆) y captar por ejemplo cambios lógicos en un sistema (por ejemplo el crecimiento de una célula se rige por esos cambios lógicos, ya que no se transforma en forma brusca).

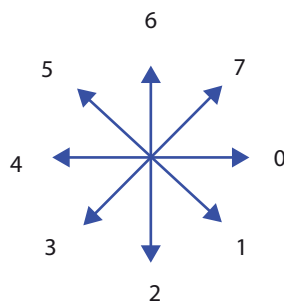
8) En *Pruebas de Personalidad*, éste es un ejemplo en el cual el concepto de lenguaje es de una simplicidad extrema y su fuerza es muy grande, ya que en éste caso la oración está formada únicamente por una cadena de letras, donde cada letra representa la respuesta a una pregunta de la prueba (normalmente tienen un máximo de 5 posibles respuestas) y la cadena representa todas las posibles respuestas dadas por una persona, de donde una oración típica se parece a una cadena binaria.

Como se observa de los ejemplos anteriores el concepto de lenguaje y es importante saber elegir el que se adecue más a nuestro problema. Ya que se eligió un lenguaje X el siguiente paso consiste en captar muchos ejemplos de oraciones (grabando, observándolo, sintiéndolo, etc.) como ejemplos distintivos o que muestran alguna característica gramatical que los distinga.

IV Detección de las Unidades Léxicas

En ésta etapa es necesario analizar las oraciones con el fin de detectar las unidades léxicas (unidades mínimas con significado en el lenguaje) y clasificarlas en tipo de unidades léxicas, por ejemplo:

1) En la *información visual* se puede tener la siguiente convención:



de donde la figura



se puede representar como : 0 0 0 6 6 6 4 4 3 2 2

y las unidades léxicas se obtienen a partir del recorrido de la figura digitalizada.

2) En un lenguaje se consulta de una base de datos se puede tener:

Dame	los	nombres	de	las	personas
acción	artículo	atributo	preposición		entidad
a	b	c	d	b	e
mayores	de	edad	30 años		
operador			valor		
f	d	c	g		
Dime	el	total	de	camiones	en
a	b	a	d	g	d
S. L. Potosi					
g					
Escribe	el	CP	de	la	colonia
a	b	g	d	b	c
Portales					
g					

Unidades Léxicas

a	b	c	d	...	
Dame	los	nombres	de		
Dime	las	edad	en		
Escribe	el				
	la				

Entidad		...			
Nombre	Tipo Transporte	...			
	camión				
	avión				

5. Obtención de la Gramática Generalizada mediante Inferencia Gramatical

Ya que se han detectado los componentes del lenguaje (Unidades Léxicas) el siguiente paso consiste en encontrar una Gramática con la cual se obtiene la estructura sintáctica del lenguaje, para lo cual se aplicará el Método de Inferencia Gramatical.

Esta etapa es precisamente la más fuerte dentro del método, ya que permite encontrar los patrones o formas generales de un lenguaje, a partir de ejemplos particulares, con lo que se encadenan las áreas de Lingüística Matemática y Reconocimiento de Formas y se obtiene un mecanismo de programación de tipo inductivo.

1 Inferencia Gramatical

La Inferencia Gramatical consiste en encontrar una Gramática a partir de un conjunto de ejemplos del lenguaje que se quiere representar.

Dados 2 conjuntos $R+$ y $R-$ donde:

$$R+ \supseteq L(G) \text{ y } R- \supseteq L'(G)$$

es decir $R+$ es un subconjunto del lenguaje generado por una Gramática G y $R-$ es un subconjunto de las cadenas que no se deben generar a partir de G , entonces la idea es encontrar la gramática G a partir de la información contenida en $R+$ y $R-$.

En particular se verán algunos métodos con los cuales se puede obtener una gramática a partir de $R+$ (conjunto de ejemplos de oraciones válidas en el lenguaje).

$$\{\alpha\} \rightarrow \text{ALGORITMO} \rightarrow G$$

donde $L(G) = \{\alpha\}$

Como primer paso de éste método se obtiene una gramática canónica sustituyendo cada unidad léxica en las oraciones ejemplo por su tipo de unidad léxica e integrando todas las oraciones resultantes en una gramática generada por un axioma S (que se introduce explícitamente).

Dado que se tiene una Gramática Canónica prácticamente todos los métodos de Inferencia Gramatical se basan en una combinación de operaciones de Factorización e introducción de Recursividad.

La factorización es una operación sobre las gramáticas parecida a la factorización algebraica, por ejemplo si se tiene:

$$\begin{aligned} S &\rightarrow a b c d \mid a b m l \quad (\text{se puede factorizar y queda}) \\ S &\rightarrow a b \{c d \mid m l\} \text{ de donde se obtiene} \\ S &\rightarrow a b X \\ X &\rightarrow c d \mid m l \end{aligned}$$

La factorización tiene la característica de transformar la gramática, por otra gramática débilmente equivalente (o sea que la nueva gramática tiene otra estructura, pero maneja exactamente las mismas oraciones que la vieja) con la cual se absorbe en una sola estructura varios ejemplos particulares.

El proceso de introducción de recursividad nos permite generalizar y obtener a partir de ejemplos concretos estructuras generales (se considera que este mecanismo es el que permite al ser humano manejar una lengua sin necesidad de tener todas las oraciones en el cerebro) sin embargo, por esta misma característica se corre el peligro de generalizar tanto que se llegue a aceptar oraciones no válidas en el lenguaje, como ejemplo de recursividad se tiene:

$$A \rightarrow a B \mid a a B \mid a a a B$$

En éste caso se propone que el lenguaje admite cadenas indefinidas de a's y se propone:

$$A \rightarrow a A \mid B$$

como gramática recursiva.

Ejemplos:

1) *Factorizar cadenas*

$$S \rightarrow a b a a b c \mid a b a b b a \mid a b b b b a$$

es equivalente a:

$$S \rightarrow a b \{a a b c \mid a b b a \mid b b b a\}$$

equivalente a :

$$S \rightarrow a b X$$

$$X \rightarrow a a b c \mid a b b a \mid b b b a$$

equivalente a :

$$S \rightarrow a b X$$

$$X \rightarrow a a b c \mid \{a \mid b\} b b a$$

equivalente a:

$$S \rightarrow a b X$$

$$X \rightarrow a a b c \mid Y b b a$$

$$Y \rightarrow a \mid b$$

2) *Factorización empotrada* (método de Solimonoff modificado)

$$S \rightarrow a \alpha_1 B \mid a \alpha_2 B \mid \dots \mid a \alpha_n B$$

produce

$$S \rightarrow a X B$$

$$X \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$$

3) *Introducir recursividad* (izquierda o derecha) a partir de:

$$S \rightarrow a a b b c \mid a a b b c \mid a a b b b b c$$

se obtiene mediante recursividad

$$\begin{aligned} S &\rightarrow a a X \\ X &\rightarrow b X \mid c \end{aligned}$$

4) *Introducir recursividad empotrada a partir de :*

$$S \rightarrow a a b c c \mid a a a b c c c \mid a a a a b c c c c$$

se obtiene

$$\begin{aligned} S &\rightarrow a a X c c \\ X &\rightarrow a X c \mid b \end{aligned}$$

5) *Inferencia Gramatical*

$$S \rightarrow a b a a b b d \mid a b c a b b d \mid a b c a b b b d$$

factorizando: a b

$$\begin{aligned} S &\rightarrow a b A \\ A &\rightarrow a a b b d \mid c a b b d \mid c a b b b d \end{aligned}$$

factorizando: a a y c a

$$\begin{aligned} S &\rightarrow a b A \\ A &\rightarrow a a B \mid c a B \\ B &\rightarrow b b d \mid b b b d \end{aligned}$$

Introduciendo recursividad

$$\begin{aligned} S &\rightarrow a b A \\ A &\rightarrow a a B \mid c a B \\ B &\rightarrow b B \mid d \end{aligned}$$

6) *Factorizar*

$$\begin{aligned} S &\rightarrow a b a d a d a \mid a b c d b e f d c a \mid \\ &\quad a b c j c j c d b e f d c f d f d g f d f d f \mid a c e f c g \mid \\ &\quad j a h b e i g j \mid a b c a d b c a \mid \\ &\quad a b g d b c g \end{aligned}$$

Primero se ordena alfabéticamente las oraciones:

a b a d g d g
a b c a d b c g
a b c d b e f d c g
a b c j c j c d b e f d c f d f d g f d f d g
a c e f c g
j a h b e i g j

De ahí se factoriza y queda:

$S \rightarrow a A \mid j a h b e y g j$
 $A \rightarrow b B \mid c e f c g$
 $B \rightarrow a d g d g \mid c C \mid g d b c g$
 $C \rightarrow a d b c g \mid d b e f d c g \mid j c j c d b e f d c f d f g f d f d$

7) *Aplicando la recursión a las oraciones anteriores:*

$B1 \rightarrow a d g d g$

resulta

$B1 \rightarrow a B2$
 $B2 \rightarrow d g B2 \mid \lambda$

entonces:

$B \rightarrow a B2 \mid c C \mid g d b c g$
 $B2 \rightarrow d g B2 \mid \lambda$
 $C1 \rightarrow j c j c d b e f d c f d f d g f d f d g$

al introducir

$X \rightarrow c \mid g \mid \lambda$

podemos tener

$C1 \rightarrow C2 d b e C3$
 $C2 \rightarrow j c C2 \mid \lambda$
 $C3 \rightarrow f d X C3 \mid \lambda$

entonces

$$\begin{aligned}
C &\rightarrow a d b c g \mid d b e f d c g \mid C2 d b e C3 \\
C2 &\rightarrow j c C2 \mid \lambda \\
C3 &\rightarrow f d X C3 \mid \lambda \\
X &\rightarrow c \mid g \mid \lambda
\end{aligned}$$

2 Método del Pivote

Busca cadenas dentro de cadenas.

- 1) Ordenar los ejemplos por el número de elementos.

$$\begin{aligned}
&a * a \\
&(a * a) * a \\
&a * (a * a) \\
&(a * a) * (a * a) \\
&(a * (a * a) * a) \\
&((a * a) * a) * a \\
&(a * (a * a) * a) \\
&(a * (a * a) * a)
\end{aligned}$$

- 2) Busca subcadenas dentro de las cadenas y se toma el más sencillo como Pivote.

VI Manejo de la Semántica

Mediante el léxico y la sintaxis únicamente se presentan los elementos básicos de un lenguaje y las relaciones permitidas entre estos elementos, sin embargo no se presenta nada acerca del significado de las palabras y de las estructuras, esto es estudiado por la Semántica.

El problema del significado se puede contemplar como una cebolla compuesta de múltiples capas, en la primera se encuentran las palabras, en la segunda las estructuras sintácticamente válidas y en las siguientes lo que se podría llamar niveles de significado. En el Lenguaje Natural una expresión válida sintácticamente puede tener múltiples significados, dependiendo del que habla y el que escucha, de la situación Social, Económica, Psicológica, etc. de los habitantes, del tiempo y lugar donde se realiza la comunicación, etc.

Ahora bien, hasta ahora se ha visto que con una gramática se puede representar la estructura de un problema, sin embargo aparte de la gramática se tiene que encontrar las acciones o rutinas semánticas y los datos sobre los que se opera para tener las tres partes que representa a un problema.

Si se observa una oración típica por ejemplo:

Dame la edad y dirección de Juan Pérez

Se nota que se tiene una acción **Dame** la cual indica una señal para que se ejecute una rutina semántica; por su parte **edad** y **dirección** indican los datos que se requieren sobre el ente Juan Pérez, de donde, a partir de una Oración puedo encontrar las acciones que se necesitan representar como rutinas semánticas, los datos que se están manejando y el orden de ejecución o estructura del problema.

Existen múltiples herramientas encaminados al manejo de la semántica y en su momento se han desarrollado escuelas completas alrededor de éste punto, en particular en éste documento se contarán sólo dos tipos de herramientas: las Gramáticas Generativo-Transformacionales y las Gramáticas con atributos.

1 Gramáticas Generativo-Transformacionales

Una Gramática Generativo-Transformacional consta de tres componentes:

- 1) *Componente Léxico.*
- 2) *Componente Generativa.*
- 3) *Componente Transformacional.*

La componente Generativa es lo que se ha estudiado en los capítulos anteriores.

Ejemplo:

$$S \rightarrow v + S \mid v$$

La componente Léxica nos permite caracterizar a los elementos léxicos mediante sus atributos. Dentro de la lingüística transformacional se considera que las oraciones tienen dos niveles.

El nivel superficial que se refiere a la estructura de la oración y el nivel profundo que refiere al significado de la misma, por ejemplo:

- 1) En A1 “*El perro corre en la calle*” y A2 “*En la calle corre el perro*”

Se observa igualdad semántica y diferencia completa sintácticamente, es decir, significado de A1 = significado de A2.

Cuando se agregan reglas transformacionales a la gramática que se tenía entonces se tiene una gramática Generativa-Transformacional.

2 Gramáticas con Atributos

Las gramáticas con atributos son simplemente gramáticas en las que se permite intercalar en las producciones llamadas a rutinas semánticas (una rutina semántica es una rutina que lleva acabo alguna acción) las cuales se representan como S_n donde n es el número de la rutina.

En general una gramática con atributos:

$$GA = \langle V_n, V_t, V_s, S, P \rangle$$

es una gramática con producciones de la forma:

$$A \rightarrow \alpha$$

donde

$$A \in V_n$$

$$\alpha \in \{V_n \cup V_t \cup V_s\}^*$$

V_n es el conjunto de variables no terminales.

V_t es el conjunto de variables terminales.

V_s es el conjunto de variables semánticas o rutinas semánticas.

$A \rightarrow \alpha$ son las producciones con atributos

Ejemplo:

$$S \rightarrow v \ S1 \ A$$

$$A \rightarrow + \ S2 \ A \mid ; \ S3$$

con lo cual se indica que si se está en el estado S y llega la señal v , entonces se manda ejecutar la rutina semántica $S1$ y se ve si llega la señal A .

En pseudocódigo la gramática quedaría como:

```
Rutina S
  si señal = v entonces
    ejecuta S1
  sino
    regresa error
  llama a A
Finrutina
```

```
Rutina A
  si señal = + entonces
    ejecuta S2;
    llama a A
  sino
    si señal = ; entonces
      ejecuta S3
    sino
      regresa error
Finrutina
```

Como se puede observar las rutinas S y A únicamente representan puntos de control y llamados a rutinas semánticas S1, S2 y S3 que son las que ejercen las operaciones sobre los datos. Entonces dada la gramática con múltiples producciones, estas se podrían representar por rutinas administradoras que únicamente llaman a otras rutinas y finalmente mandan ejecutar las rutinas semánticas.

VII Normalización de la Estructura de Control

Cuando se construye un sistema de información a partir de la gramática, es posible que el sistema tenga problemas de funcionamiento. Lo anterior es porque la gramática representa la estructura de control de programas y si esta estructura tiene problemas, el programa los tendrá.

Por lo anterior surge la necesidad de Normalizar la gramática con el fin de que la estructura sea la adecuada.

El primer paso de normalización busca que la gramática no tenga islas o variables no usables y por otro lado que siempre genere un lenguaje diferente del vacío.

En el segundo paso de normalización se pretende que la gramática sea del tipo libre de contexto determinístico con el fin de que el sistema no tenga indeterminaciones o ambigüedades.

Para lograr la normalización se aplica una serie de algoritmos orientados a tener gramáticas equivalentes que cumplan con los criterios buscados, por lo que a continuación se verán los principales teoremas sobre normalización de gramáticas (los algoritmos y demostraciones aparecen en múltiples libros sobre Lenguajes Formales y por falta de espacio sólo se presentan algunos en éste documento).

1 Primer paso de normalización Gramáticas Depuradas

Un problema que se presenta con las gramáticas libres de contexto surge cuando están mal construidas por lo que, es necesario verificar que no se tengan gramáticas con producciones sobrantes o que no generen algún lenguaje.

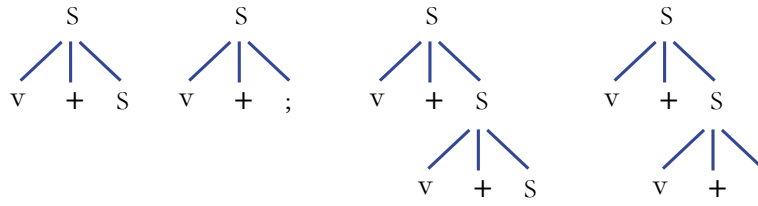
A continuación se verán algunos teoremas sobre depuración de gramáticas basados principalmente en el concepto de árbol de derivación, donde un árbol de derivación es la representación gráfica del proceso de aplicar las producciones para obtener a partir del axioma una cierta cadena.

Ejemplo:

Dada la gramática:

$$S \rightarrow v + S \mid v ;$$

Se tienen entre otros muchos los siguientes árboles:

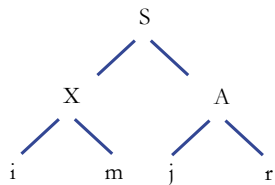


Teorema. Dada una gramática libre de contexto se puede detectar si $L(G) = \rightarrow \emptyset$

Algoritmo. 1, Genere todos los árboles de derivación de longitud menor o igual a n ; donde la cardinalidad de V_n es n . 2, Si alguno de los árboles tienen en las hojas puro elemento terminal entonces $L(G) \neq \emptyset$ (a los nodos terminales se les conoce como hojas)

Ejemplo:

1)



A partir de S se obtiene $imjr$ que es una cadena terminal, por lo que el lenguaje generado por la gramática es diferente del vacío.

Ahora se presentan teoremas para quitar “islas”:

$$A \Rightarrow \alpha \in Vt^*$$

$$S \Rightarrow \alpha A \beta$$

son los dos problemas a quitar.

En realidad se tendría que quitar a A (es una isla) porque no sirve de nada.

Ejemplo:

Sea la gramática:

$$S \rightarrow B m$$

$$S \rightarrow A b$$

$$A \rightarrow Z k$$

$$Z \rightarrow m$$

al menos una de las producciones produce una cadena terminal: $Z \rightarrow m$, de lo contrario las cadenas no se terminan nunca y nuestra gramática no estaría definida totalmente.

Teorema. Dada una gramática libre de contexto: $G = \langle V_n, V_t, S, P \rangle$ se puede encontrar una gramática equivalente $G' = \langle V_n', V_t, S, P \rangle$ tal que si $A \in V_n'$ entonces existe $\alpha \in V_t^*$ tal que $A \Rightarrow^* \alpha$

Corolario. Si $S \in V_n'$ entonces $L(G) \neq \emptyset$

Teorema. Dada una gramática libre de contexto $G = \langle V_n, V_t, S, P \rangle$ es posible encontrar una gramática equivalente G' tal que si $A \in P'$, entonces $S \xrightarrow{*} w_1 A w_2$ donde $w_1, w_2 \in V^*$

2 Segundo paso de normalización Gramáticas Libres de Contexto Determinísticas

Las gramáticas libres de contexto pueden ser:

- 1) *Determinísticas*, para cada señal que se recibe existe un solo camino para ir de un punto a otro.
- 2) *No determinísticas*, para la misma señal se pueden tomar diferentes caminos y llegar a puntos distintos.
- 3) *Ambiguas*, para una señal dada se puede llegar al mismo punto siguiendose diferentes caminos.

Ejemplo: Dadas las producciones siguientes:

$$S \rightarrow aB \mid aC$$

a partir de S y con la señal a se puede ir a B o a C.

Para que una gramática libre de contexto sea determinística debe cumplir con las restricciones siguientes:

- 1) Que no tenga producciones recursivas izquierdas o sea de la forma:

$$A \rightarrow A\alpha \quad \text{donde } \alpha \in V^*$$

- 2) Si $A \rightarrow \beta \mid \lambda$ donde $\beta, \gamma \in V^*$, $\beta = x_1 x_2 \dots x_n$ y $\gamma = y_1 y_2 \dots y_m$ entonces $x_1 \neq y_1$

- 3) Si $A \rightarrow B \mid C$ y $C \xrightarrow{*} \lambda$ entonces $B \not\xrightarrow{*} \lambda$

- 4) Si $A \rightarrow BC$ y $B \rightarrow \lambda$ entonces $B \not\xrightarrow{*} C$

Por lo que en esta parte del método lo que se busca es encontrar una gramática en la cual no se presenten estos problemas mediante la aplicación de algoritmos para quitar recursividad izquierda, quitar cadenas vacías y factorizar elementos comunes (estos algoritmos aparecen en libros de Lenguajes Formales principalmente), ya que al contar con una gramática determinística podemos obtener directamente un sistema de información en el cual existe un solo camino para cada señal recibida, con lo que se evita el indeterminismo o la ambigüedad.

Por otro lado existen gramáticas a partir de las cuales no se pueden encontrar una gramática equivalente determinística ya que es común que los algoritmos para quitar cadenas vacías y factorizar elementos comunes se neutralizan mutuamente.

Ejemplo:

Si en $S \rightarrow a B \mid a$
se factoriza por la izquierda queda:
 $S \rightarrow a X$
 $X \rightarrow B \mid \lambda$

En estos casos se tienen dos enfoques posibles: o se admite que la gramática es posiblemente no determinística y se constituye el reconocedor tomando en cuenta eso o se modifica el lenguaje para quitar el problema; por ejemplo introduciendo un ϵ y proponiendo:

$S \rightarrow a B \mid a \epsilon$;

Ejemplo: Gramática que genera sumas y restas de variables y números:

1) $G = \langle V_n, V_t, S, P \rangle$ donde $V_n = \{S, O\}$, $V_t = \{v, n, +, -\}$, $P = \{S \rightarrow v O S \mid n O S, O \rightarrow + \mid -\}$

2) Factorizando la producción $S \rightarrow v O S \mid n O S$ se obtiene la producción equivalente: $S \rightarrow \{v \mid n\} \{O S\}$

y de ahí las producciones equivalentes

$S \rightarrow A O S$
 $A \rightarrow v \mid n$

Una gramática equivalente a G es

$G_1 = \langle V_{n1}, V_t, S, P \rangle$ donde $V_{n1} = \{S, O, A\}$, $V_t = \{v, n, +, -\}$,
 $P = \{S \rightarrow A O S, A \rightarrow v \mid n, O \rightarrow + \mid -\}$

Otra gramática equivalente es:

$G_2 = \langle V_{n2}, V_t, S, P_2 \rangle$ donde $V_{n2} = \{S, O, A, B\}$, $V_t = \{v, n, +, -\}$, $P_2 = \{S \rightarrow A B, B \rightarrow O S, A \rightarrow v \mid n, O \rightarrow + \mid -\}$

VIII Implantación del Sistema

Como se comentó anteriormente una gramática es equivalente a un diagrama de Warnier o a un programa por lo que realmente la etapa de implantación se puede reducir a tomar la gramática y construir un programa conocido como compilador de compiladores al cual se le da la gramática y genera el sistema) que reconoce el lenguaje definido por la gramática, sin embargo este método sólo es válido para visualizar el proceso ya que en este caso la gramática queda inmersa en el programa, con lo que cualquier modificación a la gramática (o al lenguaje del usuario)

requiere modificar el programa, por lo que se han desarrollado otros métodos en los cuales la gramática se representa mediante alguna tabla y se construye un programa con el cual se recibe por un lado los requerimientos del usuario en su lenguaje, y por el otro la tabla gramatical y reconoce y ejecuta las oraciones que se le dieron (existen múltiples métodos para realizar este proceso en los libros de compiladores).

Este último enfoque tiene la ventaja de que únicamente se requiere obtener (comprar, desarrollar) una sola vez el programa de reconocimiento y si cambia la gramática únicamente se tiene que cambiar la tabla gramatical con lo que generalizando la idea, prácticamente toda la estructura de control de un sistema puede quedar representada mediante una tabla y ahorrarse gran cantidad de programación.

Conclusiones

Como puede observarse, este método está orientado al desarrollo de software de cualquier tipo y básicamente integra en un sólo proceso la parte inductiva y deductiva de la programación, ya que, a partir de un conjunto de ejemplos del lenguaje manejado en un área de conocimiento se puede encontrar la forma general de este lenguaje (en el documento se manejaron las gramáticas como mecanismo de representación de las formas pero realmente se puede desarrollar lo mismo con otros sistemas formales, como por ejemplo, con los autómatas) y a partir de esta generar la parte deductiva que en este caso fue representada mediante las ideas de los compiladores pero sin embargo se ha generalizado y entre otras muchas aplicaciones se ha encontrado que si:

- 1) En el estudio del lenguaje se **detectan objetos** (estos se representan en el sistema como entidades, atributos y datos) y acciones (representados como rutinas dentro del sistema) encadenados mediante algún tipo de oración, entonces las rutinas semánticas pueden ir generando una secuencia llamados a datos y acciones con lo que se tiene un generador de sistemas como parte deductiva.
- 2) En el estudio del lenguaje se detecta que este consiste en la descripción de un *conjunto de síntomas* que permiten llegar a un diagnóstico (médico, automotriz, etc.) entonces el método de inferencia gramatical lo que permite obtener es precisamente el conjunto de *reglas de inferencia* de un sistema experto y la parte deductiva es precisamente el sistema experto. Por otro lado, prácticamente todo el proceso se puede automatizar con lo que en su momento las reglas de inferencia del sistema experto se obtienen a partir de ejemplos del usuario en una forma automática.
- 3) En el estudio del lenguaje se detecta que está formada por imágenes o patrones y nuevamente si este proceso se automatiza se cuenta con un sistema que “capta” de la realidad la forma de las imágenes y aprende a reconocerlas.

Por otro lado, otro campo de aplicación muy prolífico se da en el mantenimiento, revisión y reestructuración de sistemas en operación ya que, a partir del sistema se puede obtener una gramática con atributos que representa la estructura del sistema y mediante algoritmos de equivalencia detectar y corregir problemas estructurales.

Si este proceso se automatiza, entonces se puede contar con un mecanismo que sustituye **toda** la estructura de un sistema (que en un momento puede contar con muchas rutinas, como en el caso de un sistema manejador de base de datos o un sistema experto) por una gramática (que se representa como una tabla gramatical) la cual se encadena con un reconocedor y desaparecen prácticamente las rutinas de control.

Si en general el método de Programación Dirigida por Sintaxis se automatiza, tanto en su parte inductiva como deductiva, y se le permite actualizarse continuamente con ejemplos nuevos y con la validez o no validez de sus resultados entonces se cuenta con un **Sistema Evolutivo**.

III.3 Sistemas Evolutivos de Reescritura

Fernando Galindo Soria³

Resumen

Tal vez una de las herramientas más poderosas y desconocidas de la Matemática actual sean las reglas de reescritura, su fuerza es tan grande que valdría la pena que su uso se introdujera sin mucho formalismo desde el nivel de educación primaria y secundaria, sin embargo y a pesar de que su campo de aplicación es enorme, la potencia de esta herramienta se ha diluido dentro del concepto de gramática generativa. Por lo que en este documento se presentarán las reglas de reescritura como una herramienta independiente de las gramáticas y se mostrará su campo de aplicación.

Como primer punto se dará una breve introducción al tema y se verá como un sistema de reescritura se puede representar fácilmente mediante un sistema de información integrado por un programa de control y un archivo donde se almacenan las reglas de reescritura. Con esta arquitectura el conocimiento queda independiente del programa y el proceso de mantenimiento y actualización del sistema se vuelve simple, ya que si surgen nuevas reglas solo se requiere almacenarlas en el archivo y el programa no se modifica. Aún más el sistema puede empezar a construir la base de conocimiento desde cero, ya que el archivo puede estar vacío y el sistema en forma interactiva lo puede empezar a llenar, con lo que se tiene la base de un Sistema Evolutivo.

A continuación se mostrará que problemas aparentemente diferentes como el reconocimiento de imágenes, señales, sistemas expertos, tutores, traductores, lenguaje natural elemental, y prácticamente cualquier problema que se pueda representar como una cadena de bits o caracteres, son casos particulares de un problema más general atacable con reglas de reescritura.

Finalmente se verá que cualquier programa o algoritmo X se puede representar como un conjunto de reglas de reescritura, que demostrar un teorema y hacer un programa son actividades equivalentes y que las reglas de reescrituras permiten representar directamente a las gramáticas tipo cero.

Palabras claves: Reglas de Reescritura, Sistemas Evolutivos, Gramáticas y Autómatas, Sistemas expertos, Lenguaje Natural, Reconocimiento de Imágenes, Voz y Señales.

³ Fernando Galindo Soria escribió este trabajo en 199_

Introducción:

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 más de la Matemática, tan importante como la suma y la resta, ya que abre a los alumnos todo un nuevo universo.

I 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:

$$X \rightarrow Y$$

Donde $X \in A^*$, $Y \in B^*$ (A^* y B^* representan respectivamente las cerradura de A y B). 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:

COLUMNA A	COLUMNA B
A_1	B_1
A_2	B_2
...	...
A_n	B_n

```

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”
}

```

donde el archivo representa al conjunto de reglas de reescritura:

$$\begin{aligned}
 A_1 &\rightarrow B_1 \\
 A_2 &\rightarrow B_2 \\
 &\dots \\
 A_n &\rightarrow B_n
 \end{aligned}$$

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

II 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 Traduccioni
  sino escribe “no conozco el texto” Textox
}

```


Como se podrá 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 más, 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
}

```

Como se podrá ver 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 a construir la base de conocimiento y ‘aprender’ desde cero, en tiempo real y fácilmente.**

Aunque se tienen que tomar en cuenta algunas consideraciones prácticas, por ejemplo que sólo se permita “enseñar” al sistema a personas con clave de autorización, estos motivos no afectan ni la idea ni la estructura del sistema.

Si seguimos con la misma idea el programa se puede generalizar aun mas y aplicar para atacar otros problemas aparentemente diferentes, como es la **construcción de sistemas expertos, tutores automatizados y en general al tratamiento restringido de lenguaje natural.**

Por ejemplo si se analizan algunos sistemas expertos se observa que es común que los programas en su forma más simple tengan una estructura de cascadas de IF’s de la forma:

```

Programa()
{
  lee Sx
  si Sx = S1 entonces D1 sino
  si Sx = S2 entonces D2 sino
    ""      ""
  si Sx = Sn entonces Dn sino
  escribe "no reconoce los síntomas"
}

```

Donde el programa lee los síntomas **Sx** que da el usuario y los compara con los primeros síntomas almacenados **S1**, si corresponden entonces envía el diagnóstico **D1**, sino los compara con los síntomas **S2** y así sucesivamente.

El problema de este enfoque es que si se quieren aumentar las reglas o modificar al sistema se tiene que reestructurar el programa. Ahora bien, usando prácticamente el mismo sistema evolutivo desarrollado para la traducción es relativamente fácil construir un programa que generaliza al anterior.

Para lo cual sustituiremos la cascada de IF's por un sistema formado por un archivo de reglas de reescritura de la forma:

Si → **Di**, donde se almacenan los síntomas y diagnósticos.

Síntoma	Diagnóstico
S1	D1
S2	D2
...	...
Sn	Dn

Y por un pequeño programa que realiza la sustitución de reglas.

```

Programa()
{
  i=1
  lee Sx
  mientras ((Sx != Si) y (no fin de archivo))i++
  si (no fin de archivo) escribe Di
  sino //entra al dialogo
    escribe "desconozco Sx, dame su diagnostico"
    lee Dx
    almacena en el archivo Sx , Dx
}

```

Nuevamente con este sistema se puede tener el archivo vacío y conforme se realiza el proceso de diálogo las reglas del sistema experto se van integrando en forma natural, con lo que se tiene un sistema evolutivo que genera las reglas de inferencia de un sistema experto. Lo único que cambia entre el traductor y el sistema experto es que en lugar de escribir:

desconozco texto, dame su traducción

se escribe

desconozco Sx, dame su diagnóstico

Aun más, si se cambian las reglas, con el mismo programa se puede tener un sistema de reconocimiento de lenguaje natural elemental. Por ejemplo, en lugar de poner los síntomas y diagnósticos de un sistema experto se pueden poner las preguntas y respuestas de un sistema tutor. En este caso se puede tener una lista de preguntas y respuestas como las siguientes:

Pregunta	Respuesta
¿quién descubrió América?	Colon
¿cuál es la capital de Francia?	París
...	...
¿en que continente esta Rusia?	en Europa

Entonces **el mismo programa** funciona como un sistema tutor, donde el alumno pregunta y el sistema responde, y **en lugar de síntomas y diagnósticos** el sistema **tiene preguntas y respuestas**, lo cual es interesante ya que **el mismo sistema se puede ver como un sistema experto o como un tutor automatizado**. Y lo único que cambia es que en lugar de escribir:

desconozco Sx, dame su diagnóstico

escribiría

no conozco la respuesta, dámela

III Generalización para manejo de Señales

Como se puede ver una gran cantidad de problemas aparentemente diferentes se pueden reducir en primera instancia al mismo sistema, sin embargo no son los únicos casos que se pueden manejar con esta herramienta, ya que prácticamente se puede atacar cualquier problema que se pueda representar como una cadena de bits o caracteres.

Para poder ver lo anterior se trabajará primero con algo muy simple, ya que lo importante no es resolver el problema en toda su complejidad, sino **mostrar una de las cosas mas hermosas de la Informática, que el reconocimiento de imágenes, texto, traductores, sistemas expertos, señales biofísica, voz, etc., que normalmente se ven como cuestiones completamente diferentes son en realidad casos particulares de un problema general.**

Como primer ejemplo se verá un sistema que entrega una imagen a partir de su nombre. Este sistema consta de un archivo con dos columnas, en la primera columna se encuentra el nombre de la imagen y en la segunda su imagen correspondiente (almacenada por ejemplo como cadena de bits), además se cuenta con un programa construido de tal manera que si alguien da el nombre de la imagen **N_x** regresa su imagen correspondiente **I_x** y si no la tiene entra en un proceso de diálogo donde la pide y almacena.

Nombre	Imagen
N1	I1
N2	I2
...	...
Nn	In

```
Programa()
{
  i=1
  lee Nx
  mientras ((Nx != Ni) y (no fin de nombres))i++
  si (no fin de nombres) escribe Ii
  sino //entra al dialogo
    escribe "no conozco la imagen dámela"
    lee Ix
    almacena Ix y Nx
}
```

Como se puede ver este sistema es prácticamente idéntico al que se ha estado manejando desde el principio para texto con la diferencia de que ahora un lado de la regla de reescritura no es texto.

A partir de una idea tan simple es fácil construir un **programa que pronuncie palabras a partir de texto escrito**, en este caso en la primera columna se tiene el texto y en la segunda el sonido asociado, en caso de que la computadora no encontrara la cadena escrita puede entrar en un proceso de dialogo donde se grabaría el texto y su sonido.

A continuación se puede guardar en la primera columna una serie de imágenes y en la segunda sus nombres, entonces si se presenta una nueva imagen, el sistema puede compararla con las que ya tiene, si la encuentra regresa su nombre, sino la pide y almacena la imagen y el nombre.

Imagen	Nombre
I1	N1
I2	N2
...	...
In	Nn

```

Programa()
{
  i=1
  lee Ix
  mientras ((Ix != Ii) y (no fin de imágenes))i++
  si (no fin de imágenes) escribe Ni
  sino //entra al dialogo
    escribe "no conozco la imagen dame su nombre"
    lee Nx
    almacena Ix y Nx
}

```

Con lo que se tiene un sistema elemental de reconocimiento de imágenes usando el mismo que se utilizó para tratamiento de texto, donde en lugar de manejar cadenas de caracteres, se manejan imágenes, con lo que la programación específica se puede complicar, porque no es lo mismo manejar registros de unos cuantos caracteres a manejar arreglos de miles de palabras, sin embargo la estructura general se conserva.

Aunque buscar imágenes idénticas se podría considerar como una “trampa”, este sistema es un buen ejemplo para empezar con el tratamiento de imágenes, ya que el manejo de cadenas idénticas es común en computación, por ejemplo en un compilador las palabras claves son iguales a palabras ya almacenadas.

A partir de lo anterior se pueden atacar problemas de mayor complejidad, por ejemplo modificando el sistema para que encuentre la distancia entre la imagen de entrada y la almacenada y en su caso nos indique si son idénticas o su grado de semejanza o diferencia, con lo que se tiene las bases de un sistema de reconocimiento de formas.

El tratamiento de señales en general, se vuelve prácticamente idéntico, ya que en principio, lo que se tiene son archivos donde se guardan las señales y se comparan unas con otras.

Al almacenar la información **se ha construido una mina de información** sobre la cual se pueden aplicar una gran cantidad de técnicas y métodos de reconocimiento de formas, sistemas evolutivos, lingüística matemática e inferencia gramatical, redes neuronales y otras herramientas para el manejo de información.

IV Generalización a cualquier Sistema

Al proceso de sustituir **A_i** por **B_i** se le conoce como derivación directa y se denota como:

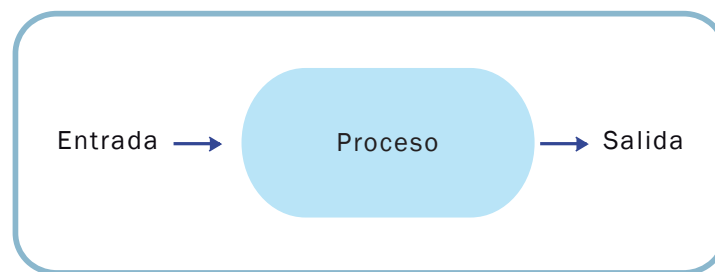
A_i ⇒ B_i

Si **A = B**

o **A ⇒ B₁ ⇒ B₂ ⇒ B₃ ⇒ ... ⇒ B**

entonces se dice que **A deriva B** y se denota como **A ⇒ B**

En general cualquier sistema informático se puede ver como un proceso de derivación, ya que sí se tiene:



El sistema sustituye la entrada en la salida (a lo mejor realizando millones de operaciones), o sea:

Entrada $\overset{*}{\Rightarrow}$ Salida

Y la secuencia de pasos que llevan de la entrada a la salida se puede visualizar como:

Entrada ⇒ a₁ ⇒ a₂ ⇒ ... ⇒ Salida

Donde para cada paso de la forma **a_i ⇒ a_j** existe una regla de reescritura de la forma **a_i → a_j**, por lo que el algoritmo o programa se puede ver como un sistema de reescritura.

De la misma forma un teorema se puede ver como

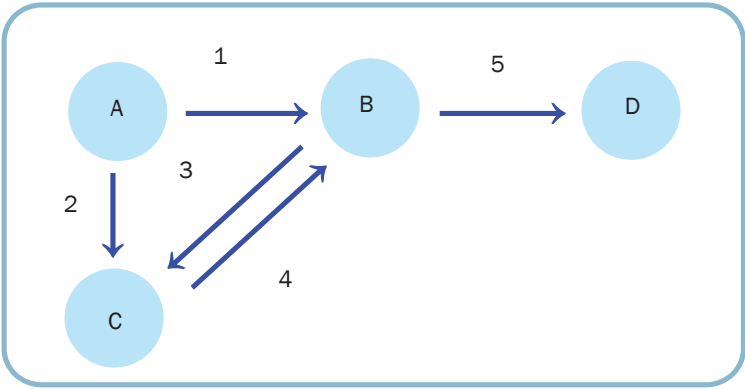
A $\overset{*}{\Rightarrow}$ B

Y la demostración del teorema se puede ver como

A ⇒ a₁ ⇒ a₂ ⇒ ... ⇒ B

O sea que, **desde el punto de vista de los sistemas de reescritura demostrar un teorema y hacer un programa son equivalentes**, lo cual es interesante, ya que normalmente se considera que es más fácil programar que demostrar teoremas, por lo que, si se adquiere conciencia de que son procesos equivalentes, se podría facilitar el desarrollo de la capacidad de demostración de teoremas a partir de la capacidad de programar.

Aún mas, esta herramienta se puede aplicar dentro del entorno de los sistemas axiomáticos (gramáticas, autómatas, Sistemas de Post, etc.) Por ejemplo si se tiene el siguiente diagrama de estados.



La gramática regular equivalente tiene reglas de reescritura de la forma:	El autómata finito equivalente se representa con las herramientas tradicionales como:	El autómata finito equivalente tiene reglas de reescritura de la forma:
$\begin{array}{l} A \rightarrow 1 B \\ A \rightarrow 2 C \\ B \rightarrow 3 C \\ B \rightarrow 5 D \\ C \rightarrow 4 B \end{array}$	$\begin{array}{l} \delta(A, 1) = B \\ \delta(A, 2) = C \\ \delta(B, 3) = C \\ \delta(B, 4) = D \\ \delta(C, 5) = B \end{array}$	$\begin{array}{l} A1 \rightarrow B \\ A2 \rightarrow C \\ B3 \rightarrow C \\ B4 \rightarrow D \\ C5 \rightarrow B \end{array}$

En este ejemplo se ve la fuerza de las reglas de reescritura, ya que, con un cambio notacional se reduce drásticamente la complejidad del manejo de los autómatas.

Por ejemplo, si se quiere demostrar que **A245** \Rightarrow **D** o sea que ha partir de **A** se llega a **D** con las señales **2, 4 y 5** con la notación tradicional quedaría:

$$\delta(A, 245) = \delta(\delta(A, 2), 45) = \delta(\delta(\delta(A, 2), 4)5) = \delta(\delta(C, 4)5) = \delta(B, 5) = D$$

Que desde el “punto de vista matemático” es poesía, pero desde el punto de vista aplicativo complica la vida (como un número romano se puede ver más elegante que uno arábigo, pero es menos práctico).

Usando reglas de reescritura queda:

$$A245 \Rightarrow C45 \Rightarrow B5 \Rightarrow D$$

Que como se puede ver es mucho más simple. Por lo que sin entrar en mas detalle quiero comentar que, los teoremas de equivalencia entre gramáticas y autómatas se vuelven más simples usando reglas de reescritura en lugar de la notación tradicional.

Las reglas de reescritura tienen un rango de aplicación enorme y están en el núcleo de solución de muchos problemas aparentemente complejos. Lo anterior no es gratis, ya que si se recuerda la jerarquía de Chomsky, las gramáticas generativas tipo cero son las más generales y equivalen a las máquinas de Turing que pueden representar el algoritmo de solución de cualquier problema computable.

Ahora bien a pesar de su fuerza la máquina de Turing normalmente no se aplica a la solución de problemas reales por la dificultad notacional y operativa que presenta.

Sin embargo las gramáticas tipo 0 tienen reglas de la forma $a \rightarrow b$ donde a y b son cadenas de elementos de algún conjunto. O sea que las gramáticas tipo cero están formadas por reglas de reescritura.

Por lo que no es raro que un sistema de reescritura sea tan fuerte ya que representa a una gramática tipo cero y por equivalencia a una máquina de Turing.

Conclusiones

En este trabajo se mostró la fuerza de las reglas de reescritura, como se pueden manejar en forma independiente de las gramáticas y aplicar a áreas tan diferentes como el reconocimiento de imágenes, tratamiento de voz, traducción de lenguajes, sistemas expertos, reconocimiento de señales y muchas otras.

Se vio a estas áreas como casos particulares del mismo problema, por lo que se esta perdiendo una cantidad enorme de energía al atacarlos como problemas independientes, ya que, muchas de las herramientas que se han encontrado para atacar un tipo de problema se podrían usar para atacar los otros y en su momento se deben desarrollar herramientas generales a las que les sean indistintas las aplicaciones particulares y aún más que llegue un momento en el que **ya no se vea cada problema en forma particular sino que se ataque el problema general.**

Finalmente se vio que las gramáticas tipo cero se representan mediante reglas de reescritura, de donde se concluyo que se puede construir un sistema con la fuerza de una máquina de Turing y la facilidad de un programa de unas cuantas líneas.

Bibliografía

- [1] Chomsky, Noam, “Estructuras Sintácticas”, ed. Siglo XXI.
- [2] Bach, Emmon, “Teoría Sintáctica”, ed Anagrama.
- [3] Salomaa, “Formal Languages”, Ed. Academic Press.
- [4] Galindo Soria, Fernando, ”Sistemas Evolutivos” en Boletín de Política Informática. México, Septiembre de 1986.
- [5] Galindo Soria, Fernando, ”Sistemas Evolutivos: Nuevo Paradigma de la Informática” en Memorias XVII Conferencia Latinoamericana de Informática. Caracas, Venezuela, julio de 1991.
- [6] Gonzalez, Rafael C., Thomason, Michael C., ”Syntactic Pattern Recognition”, ed. Addison-Wesley.

III.4 Algunas Propiedades Matemáticas de los Sistemas Lingüísticos

Fernando Galindo Soria⁴

I Desarrollo Histórico

1 De la Lingüística Matemática a los Sistemas dirigidos por Sintaxis

A pesar de que el área de la Lingüística Matemática es relativamente joven, ya que empezó a consolidarse a mediados de los 50's, su campo de aplicación en la Informática ha ido creciendo rápidamente.

Uno de sus primeros logros se presentó cuando se usó para describir la gramática del lenguaje Algol durante los años 60's propiciando que ya para finales de esa década empezaran a surgir libros donde se mostraba como construir un compilador a partir de la gramática de un lenguaje dado, y que durante los 70's se volviera cotidiana la construcción de compiladores e intérpretes bajo este enfoque.

Lo anterior ocasionó que en prácticamente todos los cursos de compiladores y de Programación de Sistemas se enseñaran una gran cantidad de métodos para analizar tanto léxica como sintácticamente las oraciones de algún lenguaje de programación.

En particular en el caso del análisis sintáctico se desarrolló una gran variedad de métodos que permiten detectar si una oración es sintácticamente correcta, verificando si cumple o no con las reglas gramaticales del lenguaje de programación.

Conforme fue madurando el área, se detectó que existían muchos problemas en los cuales el usuario se comunicaba mediante algún lenguaje con la computadora, desde los problemas más complejos de reconocimiento de lenguaje natural hasta la comunicación en algún lenguaje de control con el sistema operativo o algún lenguaje de descripción de datos con un manejador de base de datos.

⁴ Fernando Galindo Soria escribió este trabajo en septiembre 1992 en IPN UPIICSA Licenciatura en Ciencias de la Informática Sección de Posgrado e Investigación. A todo este universo de problemas se les agrupó con el nombre genérico de sistemas dirigidos por sintaxis, y es así que en la actualidad es común encontrar por ejemplo editores dirigidos por sintaxis.

En particular se detectó que algunos de estos problemas se volvían un caso específico de la construcción de intérpretes, ya que es relativamente fácil, usando las técnicas de compiladores que se reconozca por ejemplo las instrucciones de un lenguaje de control, y se ejecuten.

2 Inferencia Gramatical y Programación dirigida por Sintaxis

Ya para mediados de los 70's era común tratar de construir interpretes de múltiples tipos de lenguajes, sin embargo, casi desde el principio se encontró un problema que luego se volvió cotidiano, ya que, según la técnica empleada en la construcción de compiladores, para poder construir el compilador o intérprete de un lenguaje dado, se requiere contar con su gramática y en ningún libro de compiladores decía como encontrar la gramática de un lenguaje.

En el caso específico de los lenguajes de programación y algunos otros, la gramática la daba el diseñador del lenguaje y reflejaba las características y restricciones que se querían imponer al sistema, sin embargo, en muchos otros casos no se contaba con la gramática, sino con múltiples ejemplos de oraciones del lenguaje y con criterios y reglas empíricas, por lo que encontrar la gramática de un lenguaje dado a partir de un conjunto de oraciones se volvió un gran reto.

En paralelo con lo anterior y también desde mediados de los 60's se empezó a aplicar la Lingüística Matemática al Reconocimiento de Patrones y en particular al Reconocimiento de Imágenes y ya desde esa época se empezó a desarrollar el Reconocimiento Sintáctico de Patrones, en el cual se aplica la Lingüística Matemática para reconocer imágenes o patrones específicos viéndolos como “oraciones” de algún “Lenguaje de Patrones o Imágenes”.

Ahora bien, en el caso del Reconocimiento de Patrones es común contar con un gran número de oraciones que representan imágenes o patrones particulares, sin embargo, por lo común no se cuenta con la gramática del lenguaje, por lo que, desde mediados de los 60's se comenzaron a desarrollar un conjunto de métodos y técnicas orientados a la obtención de la gramática de un lenguaje a partir de ejemplos de oraciones de este lenguaje. A todo este conjunto de herramientas se les englobó con el nombre genérico de Inferencia Gramatical.

A principios de los 80's se empezaron a combinar la Inferencia Gramatical y la construcción de compiladores con el fin de resolver los problemas de tratamiento de diferentes tipos de lenguajes y gracias a esa interrelación ya se contaba con un conjunto de herramientas con las cuales:

- 1) *A partir de un conjunto de ejemplos* de un lenguaje se puede encontrar la gramática que describe el lenguaje.
- 2) *A partir de la gramática* se puede construir un compilador o intérprete capaz de reconocer las oraciones del lenguaje.

Ya para 1983 se tenían integradas estas herramientas en un método conocido como Programación Dirigida por Sintaxis, en el cual, se muestra un proceso para desarrollar sistemas a partir de ejemplos del lenguaje con el que se quieren dar órdenes al sistema.

3 Enfoque Lingüístico de la Informática

En un principio estas herramientas se aplicaban para encontrar la gramática y construir el compilador de lenguajes muy concretos, tipo PASCAL, FORTRAN, lenguajes de consultas y por otro lado se siguieron aplicando al Reconocimiento de Patrones.

Sin embargo, conforme avanzó el área se detectó que existían muchos otros problemas donde se podía aplicar este método, únicamente con la condición de que los problemas a atacar fueran susceptibles de representarse mediante oraciones de algún lenguaje.

Pero por otro lado, el mismo concepto de lenguaje se fue ampliando, ya que, si en un principio se utilizaron estas herramientas para manejar lenguajes como FORTRAN y PASCAL, al mismo tiempo se utilizaban para representar imágenes y patrones en general.

Actualmente el campo de aplicación de estas herramientas ha crecido enormemente y se postula que cualquier problema susceptible de ser atacado por medios automatizados es susceptible de representarse mediante oraciones de algún lenguaje llegándose a plantear así el Enfoque Lingüístico de la Informática, en el cual se considera que cualquier cosa se puede ver como una oración de algún lenguaje X.

Ahora bien, si se tuviera que tener la lista de todas las oraciones de un lenguaje no terminaríamos, por lo que comúnmente en lugar de la lista se utiliza una gramática o conjunto de reglas que representan la estructura del lenguaje.

Por lo que el principal problema cuando se tiene que manejar objetos de los cuales no se tiene la gramática es precisamente encontrar ésta.

4 Operaciones Lingüísticas

En la actualidad ya existen una gran cantidad de herramientas de Inferencia Gramatical orientadas a encontrar la gramática de múltiples tipos de lenguaje (visuales, auditivos, de trayectorias, etc.) Sin embargo, los primeros métodos presentados, tendían a ser complejos, particulares y difíciles de programar, por lo que, se empezaron a desarrollar nuevos métodos, para diferentes tipos de problemas.

A través de esta búsqueda de métodos y herramientas, ya para principios de los 80's se empezó a detectar que muchos métodos eran parecidos y solo eran un caso particular de métodos más generales.

En la actualidad se ha llegado a que existe un grupo de operaciones lingüísticas que al combinarse entre sí cubren la gran mayoría de los problemas de inferencia gramatical y por otro lado se ha detectado que estas operaciones son similares a ciertos procesos algebraicos y analíticos. En particular se cuenta con una gran cantidad de herramientas basadas en las operaciones lingüísticas de:

- 1) *Factorización.*
- 2) *Conmutatividad.*
- 3) *Distribución.*
- 4) *Recursividad.*

La Factorización y Distribución Lingüísticas son equivalentes a las algebraicas con la diferencia de que en este caso se factorizan o distribuyen componentes de una oración.

La Recursividad es, tal vez, la herramienta lingüística más poderosa ya que permite encontrar reglas generales o patrones a partir de casos particulares.

Estas herramientas de la inferencia gramatical se utilizan cotidianamente desde hace varios años, tanto para desarrollo de sistemas en forma manual mediante la Programación Dirigida por Sintaxis, como en la construcción de Sistemas Evolutivos.

Es precisamente durante el desarrollo de estas aplicaciones que se ha detectado un conjunto de propiedades de tipo matemático, presentes en la Factorización, Distribución y Recursividad Lingüística.

II Factorización Lingüística

1 Introducción

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 oraciones canónicas).

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 sola 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:

$$\underline{o1} \text{ r1 } o2 + \underline{o1} \text{ r2 } o3$$

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

A las oraciones

$$o1 \text{ r1 } o1 \text{ y } o1 \text{ r2 } o3$$

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 factores comunes en una expresión algebraica y sacarlos de la expresión, como se ve a continuación:

$$\begin{aligned} a \ b + a \ c &= a(b + c) \\ 3 \ x + 3 \ y &= 3(x + y) \\ a(b * c) + a(e / f) &= a(b * c + e / f) \end{aligned}$$

Aplicando lo anterior a las oraciones canónicas entonces tenemos que:

$$o1 \text{ r1 } o2 + o1 \text{ r2 } o3 = o1(r1 \ o2 + r2 \ o3)$$

o sea que el párrafo $o1 \text{ r1 } o2 + o1 \text{ r2 } o3$ es equivalente al párrafo $o1(r1 \ o2 + r2 \ o3)$ si sustituimos las etiquetas por los fragmentos que representan tenemos entonces que:

$$\begin{array}{ccccccc} \text{Juan} & \text{es hermano de} & \text{Pedro} & \text{y} & \text{estudia en} & \text{UPIICSA} \\ o1 & \text{r1} & o2 & + & r2 & o3 \end{array}$$

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:

$S \rightarrow a b c d e \mid$
 $a b m e x y z \mid$
 $a b m g$

conocida como Gramática Canónica donde “ $S \rightarrow$ ” se puede ver como el nombre de una rutina y el símbolo “ \mid ” como un separador entre los 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 a b c d e
del tipo a b m e x y z
o del tipo a b m g

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 “ \mid ” equivalente al “ $+$ ” del álgebra tradicional.

$S \rightarrow \underline{a b} c d e \mid$
 $\underline{a b} m e x y z \mid$
 $\underline{a b} m g$

factorizando **a b** queda:

$$S \rightarrow \underline{a b} (c d e \mid \\ m e x y z \mid \\ m g)$$

factorizando **m** queda:

$$S \rightarrow a b (c d e \mid \\ m (e x y z \mid \\ g))$$

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, por ejemplo:

Introduciendo la variable no terminal X

$$S \rightarrow a b X \\ X \rightarrow c d e \mid \\ m (e x y z \mid \\ g)$$

Introduciendo la variable no terminal Y

$$S \rightarrow a b X \\ X \rightarrow c d e \mid \\ m Y \\ Y \rightarrow e x y z \mid \\ g$$

que se lee:

El programa S genera a b y llama a la rutina X

La rutina X genera la cadena:

c d e

o la cadena m, y llama a la rutina Y.

La rutina Y genera la cadena:

e x y z

o la cadena:

g

si analizamos este programa podemos ver que:

$$S \Rightarrow a b X \Rightarrow a b m Y \Rightarrow a b m e x y z$$

Donde “ \Rightarrow ” significa “se sustituye por” de donde S se sustituye por a b X y así sucesivamente de donde llegamos que al final:

S genera a b c d e,

a b m e x y z
ó a b m g

que es la lista de oraciones originales

De donde se tiene que la gramática canónica

$S \rightarrow a b c d e \mid$
a b m e x y z \mid
a b m g

y la gramática

$S \rightarrow a b X$
 $X \rightarrow c d e \mid$
m Y
 $Y \rightarrow e x y z \mid$
g

obtenida a partir de la primera mediante la factorización lingüística generan las mismas oraciones.

Sin embargo, estructuralmente esas dos gramáticas no son iguales ya que en el primer caso se tiene sólo un nivel y en el segundo la gramática incluye tres niveles.

Cuando dos gramáticas generan el mismo lenguaje pero con diferente estructura se dice que son débilmente equivalentes.

3 De la Gramática al Programa

Ahora bien si analizamos un poco a fondo las dos gramáticas y las vemos como dos programas podemos observar que la gramática canónica se comporta como un programa con tres opciones en la primera opción el programa ejecuta las instrucciones.

a b c ...

en la segunda opción ejecuta

a b m e ...

y en la tercera

a b m g

Por lo que existe una repetición de instrucciones comunes y lo más lógico es que el programa ejecute al principio

a b

y después las opciones

c...

m e...

m g...

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

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

Programa S

1) a b c d e

2) a b m e x y z

3) a b m g

fin del programa

y la segunda se comporta como:

Programa S

a b

1) c d e

2) m

1) e x y z

2) g

fin del programa

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

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 con la colaboración de Javier Ortiz (en esa época coordinador de la Maestría en Computación del CENIDET en Cuernavaca, Morelos) 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 lo demás.

A partir de ahí integrar todas las oraciones canónicas en una Gramática Canónica, mediante factorización agrupar los síntomas comunes y ponerlos 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
dolor en el pecho, se le diagnóstico faringitis y se le
recetó antibióticos, antihistaminicos y reposo

la oración canónica equivalente sería:

s1 s2 s3 s4 d1 t1 t2 t3

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:

$$S \rightarrow \begin{matrix} s_1 & s_2 & s_3 & s_4 & d_1 & t_1 & t_2 & t_3 & | \\ & s_1 & s_2 & s_5 & d_2 & t_1 & t_4 & & | \\ & & s_1 & s_4 & s_6 & s_7 & d_3 & t_5 & \end{matrix}$$

factorizando s1

$$\begin{array}{l} S \rightarrow s_1 X \\ X \rightarrow s_2 s_3 s_4 d_1 t_1 t_2 t_3 \mid \\ \quad s_2 s_5 d_2 t_1 t_4 \mid \\ \quad s_4 s_6 s_7 d_3 t_5 \end{array}$$

factorizando s2

$$\begin{aligned} S &\rightarrow s1 X \\ X &\rightarrow s2 Y \mid \\ &\quad s4 s6 s7 d3 t5 \\ Y &\rightarrow s3 s4 d1 t1 t2 t3 \mid \\ &\quad s5 d2 t1 t4 \end{aligned}$$

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.

III Conmutatividad Lingüística

En el ejemplo los síntomas, tratamientos y diagnósticos no necesariamente están ordenados o aparecen en el mismo orden en todas las oraciones por lo que ya que se encontraron las oraciones canónicas el siguiente paso consiste en ordenar los elementos de la oración. Por ejemplo si se tiene:

s5 s1 d2 s2 t4

al ordenarla queda:

s1 s2 s5 d2 t1 t4

Lo anterior no necesariamente es aplicable a cualquier oración ya que es mucho más común que se trabaje con oraciones en las cuales no se permite la conmutatividad con lo que tenemos dos tipos de oraciones.

Las oraciones conmutables como en el caso de los Sistemas expertos y de las oraciones donde los diferentes elementos son del mismo tipo (por ejemplo una lista de atributos como: alto, fuerte, estudioso).

Y las oraciones no conmutables como en el caso general de cualquier expresión en Español, por ejemplo:

El perro mordió al gato

no es lo mismo que:

Mordió el gato al perro.

Por lo que la propiedad de la Conmutatividad Lingüística se puede aprovechar por ejemplo en los sistemas expertos pero no es de aplicación generalizada, ya que sólo se puede aplicar en general a elementos del mismo tipo y que se encuentren contiguos.

Por ejemplo la expresión algebraica:

$$a + b + c$$

es conmutable y equivalente a:

$$b + a + c$$

$$a + c + b$$

etc.

Por otro lado $a + b * c$ es conmutable bajo +:

$$b * c + a$$

o bajo *:

$$a + c * b$$

Pero no es conmutable en forma mezclada

$$a * c + b$$

Por otro lado la expresión $a - b / c$, no es conmutable bajo ningún caso ya que se obtendrían cosas como:

$$b / c - a$$

$$a / b - c$$

En el caso de la expresión lingüística se presenta exactamente el mismo problema con la diferencia de que en este caso la cantidad de tipo de elementos y de operadores es mucho mayor y puede cambiar de un lenguaje a otro por lo que requiere de un análisis específico para cada caso.

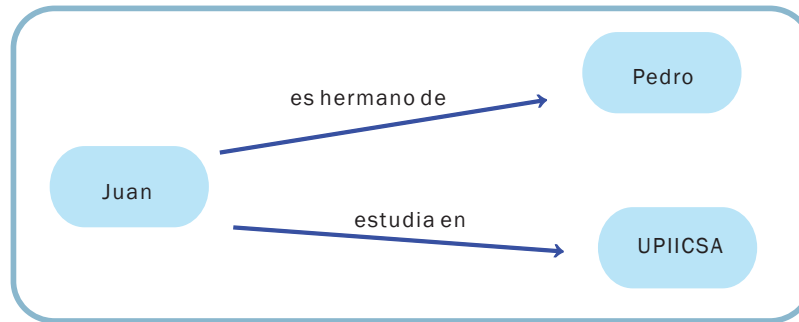
IV Distribución Lingüística

1 Antecedentes

En 1990 durante la construcción de un Sistema Evolutivo para Representación de Conocimiento desarrollado por Jesús Manuel Olivares Ceja, se presentó un problema relacionado con la generación de redes semánticas a partir de oraciones declarativas en lenguaje natural, ya que cuando las oraciones son por ejemplo:

Juan es hermano de Pedro, Juan estudia en UPIICSA

es relativamente fácil construir la red semántica:



Pero cuando las oraciones son como en:

Juan es inteligente, estudioso, trabajador y alegre

o sea de la forma

$$o1 \ r1 \ (a1 + a2 + a3 + a4)$$

donde $a1, \dots, a4$ son los atributos de Juan, no es factible generar directamente la red semántica, por lo que, es necesario transformarlas en oraciones del tipo **o r o**

2 Distribución lingüística

Para lograr lo anterior se desarrolló y aplicó una técnica conocida como Distribución Lingüística, que es la operación inversa de la factorización lingüística, por lo que:

$$o1(r1 \ o2 + r2 \ o3) = o1 \ r1 \ o2 + o1 \ r2 \ o3$$

De donde, si se tiene la oración:

Juan es inteligente, estudioso, trabajador y alegre

su oración canónica es:

$$o1 \ r1(a1 + a2 + a3 + a4)$$

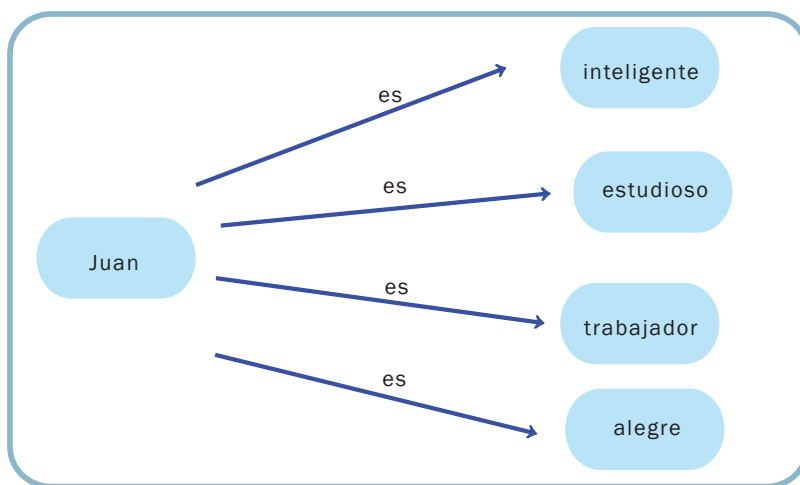
al aplicar la distribución lingüística queda:

$$o1 \ r1 \ a1 + o1 \ r1 \ a2 + o1 \ r1 \ a3 + o1 \ r1 \ a4$$

de donde se tienen las oraciones:

Juan es inteligente,
 Juan es estudioso,
 Juan es trabajador,
 Juan es alegre.

y de estas oraciones obtenidas la red semántica que se genera es:



La distribución lingüística se puede aplicar a la descomposición de oraciones relativamente complejas, por ejemplo:

$$\begin{array}{ccccccc} \text{Juan y Pedro} & \text{estudian en} & \text{UPIICSA y} & \text{trabajan en} & \text{el Metro} \\ o1 & + & o2 & r1 & o3 & + & r2 & o4 \end{array}$$

cuya oración canónica es:

$$(o1 + o2)(r1 o3 + r2 o4)$$

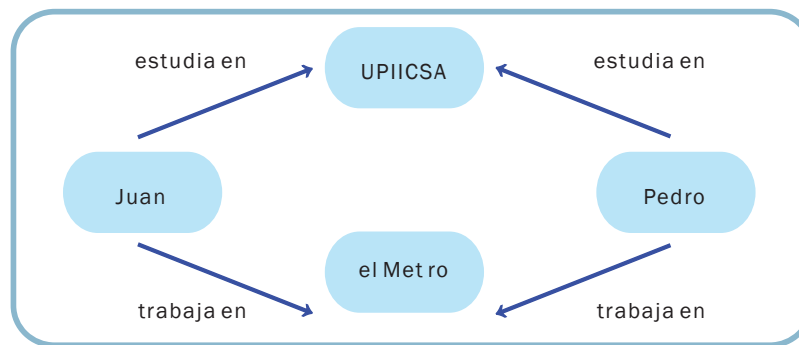
aplicando la distribución se obtiene:

$$\begin{array}{l} o1(r1 o3 + r2 o4) + o2 (r1 o3 + r2 o4) \\ o1 r1 o3 + o1 r2 o4 + o2 r1 o3 + o2 r2 o4 \end{array}$$

Que son equivalentes a las oraciones:

Juan estudia en UPIICSA y
 Juan trabaja en el metro y
 Pedro estudia en UPIICSA y
 Pedro trabaja en el Metro

De estas oraciones obtenidas se puede obtener directamente su red semántica.



V Propiedades Algebraicas de los Sistemas Lingüísticos

Las operaciones de Factorización, Conmutatividad y Distribución Lingüísticas son operaciones netamente algebraicas, por lo que, en este apartado se comentan algunas características generales de este sistema algebraico.

En primer lugar, es necesario comentar que dentro de una gramática se encuentran involucrados al menos dos operadores.

El primer operador es el operador de Concatenación que permite construir cadenas de caracteres (u oraciones) a partir de elementos simples.

Por ejemplo si se tiene el siguiente conjunto de elementos:

$$V_T = \{a, b, c, d\}$$

La operación de concatenación permite formar cadenas como:

a b c
b a b a
b c d a b c
a c

La operación de concatenación se puede ver como la multiplicación en algunos otros sistemas algebraicos.

El segundo operador presente en una gramática es el que se lee “ó” y se presenta cuando en una gramática se tiene más de una opción.

Por ejemplo la gramática

$S \rightarrow a b c d \mid$
a b d \mid
b c a

Nos dice que S se puede sustituir por:

a b c d ó a b d ó b c a

El operador “|” se puede ver como el operador de suma de algún otro sistema algebraico.

La concatenación y el operador “|” no son los únicos elementos algebraicos presentes en un sistema lingüístico ya que también se cuenta con un elemento que funciona como cadena vacía o λ .

La cadena vacía es una cadena de caracteres sin caracteres (se dice que $|\lambda| = 0$) y tiene las propiedades siguientes:

Dada una cadena de caracteres m:

$$m\lambda = \lambda m = m$$

o sea que λ se comporta como la unidad bajo la multiplicación.

Si integramos los operadores de concatenación, “|” y la cadena vacía entonces la gramática se comporta como un anillo algebraico y es susceptible de ser estudiado desde el punto de vista de la teoría algebraica.

VI Recursividad Lingüística

1 Introducción a la Recursividad

La Recursividad es tan importante que ameritaría un trabajo por si sola, por lo que en este documento solo presentaremos una breve introducción a su interrelacion con la Lingüística Matemática y el Análisis Matemático con el fin de visualizar su fuerza.

En general se considera que un sistema es recursivo cuando se llama a si mismo, por ejemplo la rutina que gráfica árboles:

```
Rutina Arbol (x0, y0, t, a)
  x1 = x0 + t * cos (a)
  y1 = y0 + t * sen (a)
  dibuja tronco (x0,y0, x1, y1)
  Arbol (x1, y1, t/2, a - 30)
  Arbol (x1, y1, t/2, a + 30)
fin Rutina
```

es una rutina recursiva porque se llama a si misma

2 Lingüística Matemática y Recursividad

Dentro de la lingüística matemática se han encontrado una gran variedad de casos en los que la mejor representación de un lenguaje es en términos de una gramática recursiva y aún más, en algún momento se ha postulado que el mecanismo “natural” de adquisición del lenguaje en los seres vivos es un mecanismo que genera estructuras recursivas, ya que por ejemplo, si se tiene el siguiente conjunto de oraciones:

Estudia y triunfaras
 a b
 Estudia, trabaja y triunfaras
 a a b
 Estudia, trabaja, ahorra y triunfaras
 a a a b

Al representarlas mediante su gramática canónica se obtiene:

a b
 a a b
 a a a b

se observa que entramos en un ciclo de repetición en el cual se pueden poner tantos elementos como se quiera, y podemos proponer que una oración del tipo

a a a ... a b

es sintácticamente válida dentro del lenguaje.

Cuando se tiene un conjunto de oraciones donde un elemento se puede repetir en forma indefinida es conveniente sustituir el conjunto de oraciones por una gramática recursiva, ya que la representación es mucho más compacta y general.

Por ejemplo si se tiene:

$S \rightarrow a b \mid$
 $a a b \mid$
 $a a a b$
 $\dots \mid$
 $a a a \dots a b$

Una gramática recursiva asociada es:

$S \rightarrow a S \mid b$

Ya que sustituyendo **S** por **a S** o por **b** se pueden tener cadenas como las siguientes:

$$S \Rightarrow a \quad S \Rightarrow a a \quad S \Rightarrow a a b$$

$$S \Rightarrow a \quad S \Rightarrow a a \quad S \Rightarrow a a a \quad S \Rightarrow a a a b$$

y en general cualquier número de **a** seguidos por una **b**.

Entonces la gramática recursiva genera todas las cadenas originales y muchas más.

3 Gramática Recursiva y Generalización

Por lo común la gramática recursiva es una generalización de las oraciones que representa, es decir, que si se tiene un conjunto de oraciones donde se detecta una estructura recursiva, ésta no solo genera el conjunto de oraciones sino es capaz de generar muchas otras que no estaban contempladas.

Esta propiedad de generalización de las gramáticas recursivas las hace extremadamente poderosas ya que permite encontrar a partir de unos cuantos ejemplos la estructura de un lenguaje.

Sin embargo al generalizar se puede llegar a proponer una gramática que genere también estructuras que no sean válidas en el lenguaje, o sea que la gramática puede ser tan general que produzca cosas sin sentido o contradictorias.

A pesar de lo anterior la fuerza de la Recursividad es tan enorme que se utiliza cotidianamente para atacar una gran cantidad de problemas de lingüística y únicamente se debe ser consciente de sus peligros y no usarla a ciegas.

4 Método de Generación de Gramáticas Recursivas

En muchos casos la recursividad se ha introducido en forma intuitiva a los sistemas, sin embargo, ya existen métodos que permiten obtener una gramática recursiva a partir de un conjunto de oraciones del lenguaje.

Por ejemplo si se tiene la gramática:

$$\begin{array}{l} S \rightarrow a b \mid \\ \quad a a b \mid \\ \quad a a a b \end{array}$$

la gramática recursiva que la generaliza es:

$$S \rightarrow a S \mid b$$

Ahora bien, si se observa el ejemplo se puede notar que la recursividad se introduce cuando se detecta que una cadena **a** se repite en forma monotona **a a a** alrededor o tendiendo a un punto **b**. El punto **b**, al cual tiende la cadena se le conoce también como límite o núcleo.

Por ejemplo en la oración

a b c a b c a b c a b c d

la cadena a b c presenta un comportamiento monótono que termina en d.

Dada una oración repetitiva para generar la gramática recursiva se siguen los siguientes pasos:

- 1) *Se detecta un comportamiento monótono.*
- 2) *Se busca el elemento repetitivo.*
- 3) *Se detecta a que punto tiende.*
- 4) *Se genera la gramática recursiva.*

Por ejemplo, en la cadena:

3 8 3 8 3 8 3 8 3 8 m

El elemento repetitivo es 3 8 el sistema tiende a m, por lo que la gramática es:

$$S \rightarrow 3\ 8\ S \mid m$$

o sea que toda la cadena repetitiva se sustituye por el elemento repetitivo 3 8 seguido de un llamado recursivo S, por otro lado el núcleo del sistema m se pone como otra opción de la gramática.

Por ejemplo la cadena:

m n o p m n o p m n o p q

tiene como cadena repetitiva a m n o p y tiende a q

Por lo que la gramática queda como:

$$S \rightarrow m\ n\ o\ p\ S \mid q$$

Un caso interesante se presenta cuando el comportamiento monótono es alrededor de un punto, como por ejemplo en la oración,

((((a))))

donde el número de paréntesis izquierdos es el mismo que derechos y giran alrededor de a.

La letra a funciona como núcleo del proceso recursivo.

Para generar la producción recursiva se sustituye el núcleo por la variable recursiva, de donde la gramática queda:

$$S \rightarrow (S) \mid a$$

5 Enfoque Analítico de los Sistemas Lingüísticos

Como se puede observar según este enfoque el concepto de recursividad es similar al concepto de límite en el Análisis Matemático ya que en los dos casos se tiene un conjunto de elementos que tienden a o giran alrededor de un núcleo o límite.

Las similitudes no se quedan en ese punto, ya que, el proceso recursivo es un proceso que se puede continuar indefinidamente y requiere de un atractor, límite, núcleo o criterio de terminación para detenerse, de donde, tal vez, el Límite del Análisis Matemático y la Recursividad son dos componentes de un fenómeno más general.

Aparentemente lo anterior es muy factible, ya que se tienen ejemplos de procesos recursivos que “tienden a un límite” como es el caso de algunos fractales que en el límite tienden a ocupar el espacio delimitado por otros fractales.

Conclusiones

En este trabajo se presentó una breve introducción a la Lingüística Matemática y a sus propiedades Matemáticas. En particular se presentaron las operaciones de Factorización, Conmutatividad y Distribución Lingüística y se vio su contraparte algebraica, llegándose a proponer la construcción de un Álgebra Lingüística soportada por las operaciones de concatenación, por “|” y por el neutro multiplicativo o cadena vacía λ , por lo que, específicamente se planteó que estamos ante la presencia de un Anillo Algebraico, y es factible ver al Álgebra Lingüística, como un nuevo campo de acción y en particular empezar a aplicarle los resultados encontrados durante el desarrollo de la Escuela Algebraica.

Por otro lado, se dio una introducción a la operación de Recursividad Lingüística y se vio su similitud con el concepto de Límite del Análisis Matemático, por lo que, nuevamente se presentó una interrelación entre dos campos aparentemente disímiles, y se plantea que de esta interrelación se puede enriquecer tanto el Análisis como la Lingüística.

Las operaciones algebraicas y analíticas de la Lingüística Matemática no están separadas ya que en la mayoría de los problemas de Inferencia Gramatical se aplican conjuntamente, por lo que, otro campo de estudio se encuentra en la interrelación del Álgebra y el Análisis en el estudio de problemas que involucren por ejemplo la Factorización y la Recursividad Lingüística.

Finalmente no quiero perder la oportunidad de mencionar que he escuchado comentarios acerca de que los pueblos prehispánicos manejaban los conceptos de Límite y Recursividad dentro de un mismo campo, por lo que, tal vez, estemos en el umbral de una puerta que ha permanecido cerrada por 500 años.

III.5 Sistemas Evolutivos de Lenguajes de Trayectoria

Fernando Galindo Soria⁵

Resumen

En este trabajo se presenta un proceso para construir Sistemas Evolutivos basados en lenguajes de trayectoria, donde un Lenguaje de Trayectoria se puede ver como un conjunto de oraciones que representan la trayectoria entre dos puntos.

Este concepto generaliza la idea de lenguaje, ya que prácticamente todos los elementos de los lenguajes tradicionales (escrito, hablado) se pueden tratar como una secuencia de trayectorias (por ejemplo, una letra es el resultado de la trayectoria que sigue la pluma de un punto a otro) y por otro lado existen una multitud de fenómenos que vemos como una trayectoria, por ejemplo, el movimiento de un brazo humano o mecánico, el recorrido por una ciudad, un movimiento de ajedrez, el movimiento de los planetas, la danza de las abejas, la estructura de una casa, la relación entre los componentes químicos de una sustancia, etc.

I Conceptos Generales

1 Sistemas Evolutivos

Cuando se involucra la Informática en algún área de estudio, las aplicaciones y los problemas a resolver al principio son relativamente sencillos y han sido atacados y resueltos de muchas formas previamente, sin embargo, conforme pasa el tiempo los problemas se van complicando y son cada vez más difíciles de resolver, hasta que llega un punto en el que la relación de la Informática-Área de Conocimiento se enfrenta a problemas nuevos y para los cuales no existen soluciones previas, esta situación se empieza a presentar cada vez más en las áreas con mayor tiempo de interacción con la Informática o en las áreas en las que no existen algoritmos o métodos de solución de problemas ya establecidos. Lo anterior ha ocasionado que se cuestione cada vez más los métodos tradicionales de la informática, en los cuales el informático (desarrollador de sistemas, Ingeniero de Software, Ingeniero de Conocimientos, etc.) analiza un problema y propone una solución en términos de un conjunto de reglas que al aplicarse permiten resolver el problema (método deductivo de programación), ya que cada vez es más difícil encontrar precisamente esas reglas, por lo que, desde hace varios años se han desarrollado nuevos enfo-

⁵ Fernando Galindo Soria, VI Reunión Nacional de Inteligencia Artificial, Memorias, Querétaro, junio 1989 Editorial Limusa, IPN DGIT UPICSA CENIDET México, D.F. Cuernavaca, Morelos.

ques basados en la idea de contar con herramientas que permitan resolver los problemas de la informática sin necesidad de tener en forma explícita el método de solución.

Dentro de estas herramientas se cuenta principalmente con los Sistemas Evolutivos, capaces de interactuar con su ambiente y crear imágenes de ese ambiente, detectando e integrando en forma automática en la imagen los mecanismos y reglas generales que permiten resolver los problemas que se le plantean.

En este trabajo se plantea la construcción de un Sistema Evolutivo orientado a reconocer lenguajes de trayectoria y a encontrar los patrones o reglas generales representadas por las trayectorias específicas.

2 Familias Lingüísticas

Tradicionalmente en el área de la informática el concepto de lenguaje esta muy restringido y por lo común o se piensa en lenguajes muy específicas (tipo FORTAN, PASCAL. PROLOG. JCL, QUERY, etc.) o si bien nos va, en términos de “lenguaje natural restringido”, con lo que se quiere indicar algún subconjunto del Español escrito, sin embargo el concepto de lenguaje es mucho más amplio, ya que cualquier mecanismo en el que se encuentre un conjunto de elementos (alfabeto) sobre los que se pueda aplicar un conjunto de reglas para relacionarlos (sintaxis) y asociarle un significado (semántica) se puede decir que cuenta con un lenguaje y en su momento pueden existir lenguajes de múltiples tipos: natural, escrito, simbólico, hablado, visual, etc., con lo que, el universo de posibles mecanismos de comunicación es enorme y no es conveniente restringirse únicamente a lenguajes tradicionales (menú, ensamblador, pseudocódigo, natural restringido, etc.) si existe algún otro lenguaje más adecuado.

Dentro del proceso de desarrollo de los sistemas evolutivos se ha detectado que en su momento la construcción del sistema se puede facilitar si se toma en cuenta las características del lenguaje que se maneja en el campo problema.


En particular, se han encontrado dos grandes Familias Lingüísticas, con la característica de que prácticamente cualquier problema utiliza alguna de estas familias o una combinación de ellas, en la primera familia tenemos lo que tradicionalmente se engloba como lenguajes e incluye los Lenguajes Imperativos formados por oraciones en las que se plantea algún requerimiento sobre el sistema, Lenguajes Declarativos, los cuales se centran en el planteamiento de hechos o reglas de inferencia y los Lenguajes Interrogativos mediante los que se plantean preguntas sobre el sistema.

La segunda familia lingüística engloba lo que se conoce como Lenguajes de Trayectoria en los cuales se incluyen todos los lenguajes visuales (gráficas, movimiento, jugadas en un tablero de ajedrez, movimiento de un robot, etc.) y cualquier otro lenguaje que se pueda representar por una trayectoria (sonido, recorrido por una ciudad, etc.).

3 Lenguajes de Trayectoria

Normalmente estamos acostumbrados a manejar el concepto de lenguaje como sinónimo de lenguaje natural escrito o hablado y si nos fuerzan mucho podemos aceptar la idea de lenguaje de señas o movimientos, sin embargo, atrás de todos estos tipos de lenguajes podemos encontrar un concepto más general que los incluye y generaliza conocido como Lenguajes de Trayectoria.

Un lenguaje de trayectoria se puede conceptualizar como un conjunto de oraciones que representan la trayectoria espacial o temporal entre dos puntos para lo cual en lugar de lexemas o fonemas utiliza como unidades básicas símbolos que representan trayectorias.

Esta idea generaliza el concepto de lenguaje, ya que prácticamente todos los elementos de los lenguajes tradicionales (natural, escrito y hablado, señas, etc.) se pueden ver como una combinación de trayectorias ya que al final de cuentas lo que se representa mediante una letra es el resultado de la trayectoria que sigue la pluma de un punto a otro o lo que se escucha como un fonema es la abstracción de la trayectoria que sigue una onda sinusoidal y por otro lado existen múltiples fenómenos que captamos como resultado de una trayectoria en una, dos o más dimensiones (el movimiento de los planetas, el movimiento de un pie, el paso del tiempo, el crecimiento de una célula, etc.), por lo que en su momento a todos estos fenómenos se les puede asociar un lenguaje de trayectoria y podemos postular que prácticamente cuando se capta algún fenómeno en primera instancia estamos captando oraciones de algún lenguaje de trayectoria las cuales al ser absorbidas y procesadas por nuestros sentidos son integradas mediante un patrón general al cual se le asocia algún tipo de significado, por ejemplo, al ver la letra **A** en principio captamos una trayectoria a la cual se le asocia un símbolo (**A**, abstracto) y un significado (vocal **A** mayúscula, primera letra del alfabeto, etc.); sin embargo algún otro símbolo, como por ejemplo,  no tiene asociado un símbolo abstracto y mucho menos un significado, por lo que simplemente lo vemos como una línea quebrada (o sea que lo vemos como una trayectoria), así mismo, cuando escuchamos un sonido, lo podemos conceptualizar como algo con significado, si es un sonido conocido o podemos decir simplemente que fue un ruido, sin embargo en cualquier caso lo captamos como una trayectoria sinusoidal.

En este trabajo se postula que si se logra encontrar el lenguaje de trayectoria asociado con algún área problema, entonces es factible construir un mecanismo que permita reconocer los patrones de comportamiento y solución de esa área problema. Por ejemplo, en el caso de lenguaje natural escrito las letras se pueden distinguir entre sí porque tienen un patrón de trayectoria diferente unas de otras, en el caso de los sonidos se tienen también trayectorias diferentes para sonidos diferentes, en el caso del movimiento de un brazo, el movimiento de la cola de un perro o el baile de una abeja se tienen nuevamente patrones de comportamiento con significados muy precisos.

4 Ejemplos de Lenguajes de Trayectoria

Estamos tan acostumbrados a integrar patrones generales a partir de fenómenos específicos que nos es difícil conceptualizar en primera instancia la representación de estos fenómenos como una trayectoria, por lo que, a continuación se presentan unos ejemplos de fenómenos asociados con lenguajes de trayectoria:

- 1) *En visión*. La imagen a reconocer, la cual por ejemplo, se puede transformar en una oración tradicional mediante la aplicación de las técnicas de primeros vecinos y números de forma.
- 2) *Juegos de tableros*. En este caso la trayectoria de la pieza en el tablero se puede ver como una imagen visual y por tanto aplicarle las herramientas de visión.
- 3) *Características espaciales*. En este caso se pretende representar conceptos no tangibles como arriba, abajo, al lado, etc. Por lo que se requiere la confluencia de varios mecanismos de captación de información (visual, espacial, escrita, etc.) y su integración en una sola oración.
- 4) *Características Temporales*. Se pueden integrar en una sola oración eventos ocurridos en diferentes tiempos (introduciendo un operador temporal \uparrow) y captar por ejemplo cambios lógicos en un sistema (por ejemplo, una secuencia de dibujos animados).
- 5) *Trayectorias Temporales*. El movimiento de un objeto se puede representar como una trayectoria en el tiempo en la cual a cada posición del objeto conforme pasa el tiempo corresponde un punto en la trayectoria (por ejemplo el movimiento de los planetas representa trayectorias temporales).
- 6) *Estructuras*. La estructura de un objeto se puede “visualizar” como una trayectoria (por ejemplo la estructura de un cubo o de una casa) y en su momento asociar la estructura con las propiedades del objeto, por ejemplo la estructura de una cadena de química orgánica esta relacionada con las propiedades de esa sustancia.
- 7) *Movimiento de un cuerpo*. Por ejemplo el movimiento de una mano o brazo (de algún animal o robot) se puede representar como la trayectoria que sigue la mano o el brazo al moverse. Por lo que, por ejemplo, mediante un lenguaje de trayectoria se pueden programar directamente los movimientos de un robot.
- 8) *Recorridos geográficos y espaciales*. Por ejemplo, el recorrido de una persona por una ciudad se puede ver como la trayectoria que sigue la persona para ir de un punto a otro y en su momento es indistinto decir que la persona va de Tacuba a la Alameda o dar la trayectoria que representa ese recorrido. Como otro ejemplo de recorrido espacial se tiene el baile con el que las abejas comunican la distancia y dirección donde se puede recolectar miel.

II Construcción de Sistemas Evolutivos de Lenguajes de Trayectoria

1 Arquitectura del Sistema

En general un sistema evolutivo consta de un constructor y un ejecutor integrados mediante un mecanismo de diálogo. El constructor es el sistema responsable de interactuar con el ambiente para construir una imagen de éste y encontrar las reglas generales que permiten resolver problemas en ese ambiente; por su parte el ejecutor interactúa con el ambiente para resolver

problemas específicos aplicando los mecanismos de solución detectados por el constructor. Estos dos sistemas se integran a su vez mediante un mecanismo de diálogo que permite pasar el control del constructor al ejecutor y viceversa.

El ejecutor en general es equivalente a un intérprete o reconocedor sintáctico de imágenes y por su parte el constructor esta formado por tres grandes componentes: constructor léxico, constructor sintáctico y constructor semántico.

- 1) *Constructor léxico*. Es el componente encargado de detectar las diferentes unidades léxicas involucradas en el lenguaje, su significado y su relación con las otras unidades léxicas dentro de la oración (oración canónica).
- 2) *Constructor sintáctico*. Es el encargado de encontrar los patrones de comportamiento, estructura general o sintaxis del lenguaje.
- 3) *Constructor semántico*. Permite asociar su significado a los diferentes componentes de la oración, a las diferentes oraciones y a las estructuras del lenguaje.

2 Constructor Léxico

El constructor léxico permite analizar las oraciones con el fin de detectar las unidades léxicas (unidades mínimas con significado en el lenguaje) e integrar la clasificación de los tipos de unidades léxicas.

En el caso de los lenguajes de trayectorias uno de los mecanismos más sencillos para asociar las unidades léxicas es lo que se conoce como número de forma.

- 1) Por ejemplo, en el caso de la información visual una de las *convenciones más usadas* se muestra en la figura 1.

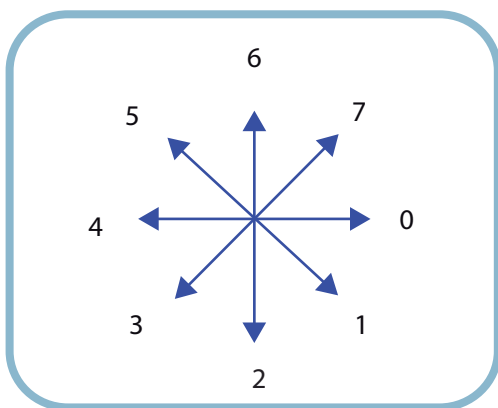


Figura 1. Códigos usados en el número de forma

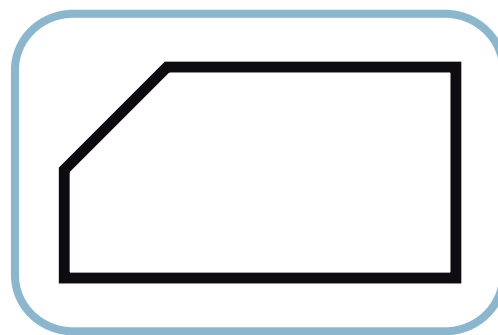


Figura 2. Un número de forma y su imagen.

De donde la imagen de la figura 2 tiene asociada la oración de trayectoria en términos del número de forma.

2) *La trayectoria que sigue un brazo de robot, el movimiento de una pieza de ajedrez o una persona en una ciudad* se pueden representar también mediante su número de forma.

3) Existen *trayectorias no lineales para las que el número de forma no es suficiente* (por ejemplo, una figura estrellada) por lo que, otros componentes léxicos pueden representar relaciones estructurales o espaciales.

Por ejemplo, una figura formada por un triángulo sobre un cuadrado puede representarse mediante el número de forma del triángulo relacionado con el del cuadrado mediante una relación que indique triángulo sobre cuadrado.

Así también, se pueden introducir relaciones espaciales como: arriba, abajo, izquierda, derecha, dentro, fuera, mayor, menor, etc., y relaciones temporales como: antes, después, al mismo tiempo, etc.

A partir de la detección de las unidades léxicas el constructor léxico permite encontrar lo que se conoce como gramática canónica, formada por las oraciones canónicas, o sea el conjunto de oraciones que se obtienen al sustituir los elementos de una oración dada por las unidades léxicas que los representan.

Por ejemplo, si a partir de cuatro figuras diferentes se obtienen los siguientes números de forma:

00066644322
060066444222
00066653331
171757535313

Entonces la gramática canónica asociada es:

$S \rightarrow$ 00066644322 |
060066444222 |
00066653331 |
171757535313

3 Constructor Sintáctico

El constructor léxico, únicamente permite encontrar los componentes de una oración pero no la gramática o estructura general del sistema por lo que se han desarrollado un conjunto de herramientas que permiten encontrar patrones, formas o estructuras generales a partir de ejemplos o casos particulares, estos métodos se engloban en lo que se conoce como inferencia gramatical.

Se tiene gran cantidad de métodos de inferencia gramatical y la mayoría se basan en las operaciones lingüísticas de factorización y recursividad, sin embargo existe una serie de problemas que no se pueden atacar directamente con los métodos tradicionales de factorización o recursividad

o que se prestan a una generalización de estos mismos métodos. En particular a muchos de los lenguajes de trayectoria, no se les puede aplicar los métodos tradicionales por lo que a continuación se presenta un método orientado a los lenguajes de trayectoria.

3.1 Factorización Multiempotrada mediante Introducción de Atributos (método de Sandra Camacho Villanueva)

El manejo de los lenguajes de trayectoria presenta el problema de encontrar la gramática de una trayectoria o forma que representa un movimiento. Como caso particular, se tiene por ejemplo, el problema de encontrar la gramática de las figuras cerradas de dos dimensiones (que es uno de los problemas “más sencillos” del reconocimiento de imágenes).

En este caso, lo primero que se hace es encontrar las oraciones para representar a las imágenes, para lo cual se aplican los conceptos de primeros vecinos y de número de forma como se vio en el constructor léxico.

El problema se pone interesante cuando se intenta encontrar la gramática de algunas figuras regulares como cuadrados, triángulos o rectángulos ya que por ejemplo si se dan las imágenes de la figura 3, se obtienen las oraciones 000222444666 y 0000222244444666.

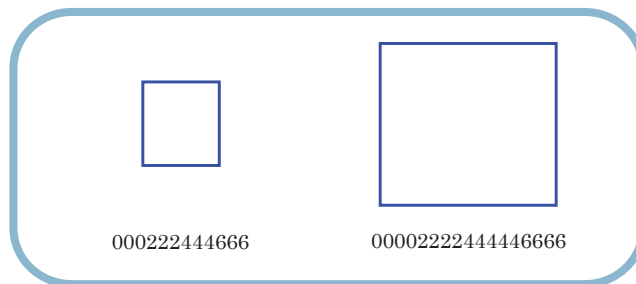


Figura 3. Imágenes y su número de forma.

Para las cuales a pesar de su sencillez y regularidad no se puede encontrar una gramática libre de contexto que las genere. Lo anterior ocurre porque el patrón de comportamiento es multiempotrado (son más de dos cadenas las que tienen un comportamiento empotrado) ya que se tienen cuatro componentes relacionadas entre sí (cada componente corresponde a un lado y la relación nos muestra que:

$$\#0 = \#2 = \#4 = \#6$$

y las gramáticas libres de contexto no tienen “memoria” para recordar tantos comportamientos relacionados.

La forma como se resolvió este problema fue introduciendo el concepto de atributo y descomponiendo cada subcadena en dos atributos: dirección y magnitud de donde la oración:

000222444666

queda como:

0(3)2(3)4(3)6(3)

y la oración

0000222244446666

queda como

0(4)2(4)4(4)6(4)

Como siguiente paso, se construye una matriz de relación entre las subcadenas donde se detecta la relación entre las magnitudes:

	0	2	4	6
0	=	=	=	=
2		=	=	=
4			=	=
6				=

En este caso tanto para el primer ejemplo como para el segundo se detectó que la magnitud entre todas las componentes tiene que ser la misma, si se continúa dando cuadrado se sigue detectando la misma regla por lo que se propone la siguiente gramática con atributos para el cuadrado:

$$S \rightarrow 0(n)2(n)4(n)6(n)$$

donde n indica el número de elementos y en este caso el número de elementos en cada dirección 0 2 4 6 es el mismo.

Otro ejemplo es el siguiente:

<div><div><div></div><div>0022244666 0(2)2(3)4(2)6(3)</div></div><div><div></div><div>00002222224446666666 0(3)2(6)4(3)6(6)</div></div></div>										
	0	2	4	6			0	2	4	6
0	=	<	=	<		0	=	<	=	<
2		=	>	=		2		=	>	=
4			=	<		4			=	<
6				=		6				=

De donde se puede concluir que:

#0 = #4

#2 = #6

#0 < #6

O sea que el número de ceros debe ser el mismo que el de cuatros y el de dos debe ser el mismo que el de seis, pero el numero de ceros es debe ser menor del numero de seis de donde la gramática del rectángulo es:

$S \rightarrow 0(n)2(m)4(n)6(m)$

4 Constructor Semántico

Ya que se tiene la gramática que representa ciertos tipos de trayectorias es posible distinguir entre los elementos representados por una u otra trayectoria, sin embargo eso no significa que se sepa lo que significa alguno de los elementos.

El constructor semántico es el componente del sistema evolutivo encargado de integrar el significado a los diferentes componentes del lenguaje, para lo que se utilizan tres herramientas combinadas: un mecanismo de diálogo, integración de diferentes mecanismos de captación de la realidad y sustitución de significados por significados más elementales.

4.1 Mecanismo de Diálogo (desarrollado por José Luis Díaz Salas)

Permite asignar significado en forma directa a los diferentes componentes de un lenguaje, ya que lo que hace es que, cada que encuentra una componente que no comprende, pregunta por su significado y espera una respuesta que se toma en forma recursiva como una nueva entrada al sistema evolutivo.

En general el mecanismo de diálogo tiene la forma:

```
Sistema Diálogo.
Lee oración
Analiza componente (trata de detectar las unidades léxicas)
Si no entiende
    entonces
        Llama a Diálogo
Ejecuta significado
Fin Diálogo
```

Otra herramienta fundamental para asignar significado es la capacidad de manejar diferentes mecanismos de captación de información en paralelo e integrarlo en una sola oración de tal manera que el significado de cada componente se puede explicar de diferentes formas.

Así, si se dibuja un triángulo y se le escribe en Español el letrero “este es un triángulo” el sistema debe integrar las dos oraciones (visual y escrita) en una sola oración o red semántica con dos componentes, donde una explica a la otra.

Como un caso particular, dada una trayectoria se le puede asociar un significado específico a cada uno de los componentes de la trayectoria con lo que, sí por ejemplo, se dibuja la estructura de una casa y se quiere hacer diseño estructural asistido por computadora se pueden integrar rutinas semánticas a cada una de las trayectorias asociadas con cada componente (trabes, castillos, zapatas, etc.) de tal manera que conforme el sistema evolutivo recorre una estructura particular va ejecutando las rutinas semánticas asociadas con cada punto de la trayectoria y al final puede entregar los valores de diseño (como un ejemplo particular se tienen las redes de transición aumentadas o ATN, que son autómatas finitos con significado asociado a las trayectorias y nodos del autómata).

La tercera componente para integración de la semántica es una generalización del mecanismo de diálogo y fue desarrollada por Gustavo Martínez Cortés y Eduardo Camacho Flores y consiste básicamente en dar el significado de una oración mediante una oración o combinación de oraciones más sencillas.

Conclusiones

Mediante los lenguajes de trayectoria se generaliza el concepto de lenguaje ya que engloba como casos particulares a los lenguajes naturales escritos y hablados, los movimientos de una pieza de ajedrez, el recorrido en una ciudad, los movimientos de un robot, la trayectoria en el tiempo de un objeto, etc., por lo que, múltiples problemas de la Inteligencia Artificial (IA), como el reconocimiento de formas, lenguaje natural, juegos, reconocimiento de imágenes, programación de robot, y de otras áreas de la Informática como el diseño asistido por computadora pueden tratarse mediante lenguajes de trayectorias. En todos los casos anteriores el problema general se centra en el lenguaje de trayectoria, por lo que, en este documento se presentó el proceso para desarrollar un sistema evolutivo capaz de construir una imagen de la realidad en términos de trayectoria incluyendo los elementos léxicos, la gramática o estructura del sistema y el manejo del significado asociado en cada problema a la trayectoria.

III.6 Aplicación de la Lingüística Matemática a la Generación de Paisajes

Fernando Galindo Soria⁶

Resumen

Dentro del área de tratamiento de imágenes existe una gran cantidad de problemas en los que se requiere manejar objetos de la naturaleza como: árboles, nubes, estrellas y montañas, ya sea, para generar paisajes o construir animaciones de crecimiento de árboles o nubes entre otros.

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 problema y ya se cuenta con una gran cantidad de herramientas para resolverlos. Ahora bien, comúnmente se utilizan técnicas distintas para resolver los diferentes problemas, el método que permite generar las montañas no se aplica para generar árboles y por un lado se tienen métodos para generar nubes, por otro estrellas o árboles y por otro peces o caracoles, siendo cada método diferente y específico y la construcción de un paisaje involucra el uso de múltiples métodos diferentes.

Por lo que, en este trabajo se presenta una herramienta general basada en la Lingüística Matemática y los fractales y en particular en una ampliación semántica de las Gramáticas Generativas, específicamente se presenta una ecuación lingüística de la forma $S \rightarrow a^*S^*$, con la cual se puede representar la estructura de múltiples elementos de la naturaleza, incluyendo árboles, nubes, estrellas, montañas, caracoles y ríos y que proponemos como una de las ecuaciones fundamentales de la naturaleza.

Dado que una regla lingüística o producción es fácil de representar mediante un programa de computo, se mostrará como representar la ecuación $S \rightarrow a^*S^*$ mediante un pequeño programa, al que cambiándole el valor de un parámetro puede generar por ejemplo un bosque de helechos, o una nube en forma de perro, finalmente se muestra como con la misma rutina y simplemente cambiándole parámetros se pueden construir múltiples tipos de paisajes incluyendo: montañas nevadas, paisajes marinos, playas y muchos otros.

⁶ Fernando Galindo Soria escribió este trabajo en la Escuela Superior de Computo (ESCOM-IPN).

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 problema y específicamente ya se cuenta con una gran cantidad de herramientas que permiten generar montañas, nubes, plantas y otros elementos de un paisaje.

Ahora bien, comúnmente se utilizan técnicas distintas para resolver los diferentes problemas y es así que por un lado se tienen métodos para generar nubes y por otro estrellas o árboles, siendo cada método diferente y específico.

Por ejemplo, en el caso de las plantas es común encontrar métodos en los cuales a partir de un elementos inicial y mediante un algoritmo recursivo se producen ejemplos de plantas muy naturales, por otro lado, desde los años 70's ya se encontraban aplicaciones de los Sistemas-L a la generación de plantas con muy buenos resultados.

Tanto los sistemas-L como los métodos basados en un Iniciador-Generador se han aplicado a construir además de plantas también ríos, rayos y otros elementos de la naturaleza que tienen una estructura dendrítica (formada por múltiples ramas que se entrecruzan como una red neuronal o de vasos capilares).

Por otro lado, en el caso de nubes o montañas se ha tenido también un gran desarrollo y así existen métodos para generar montañas mediante particiones sucesivas de una curva o mediante el expediente de generar en 3D la imagen de un objeto y colocando en el eje Z la densidad del objeto (Por ejemplo tomar la fotografía de una galaxia y graficar en Z la densidad de estrellas).



Sin embargo estos métodos por lo común son independientes unos de otros y el que produce por ejemplo las montañas no se aplica para generar árboles, por lo que la construcción de un paisaje involucra el uso de múltiples métodos diferentes.

En este trabajo se presenta una herramienta basada en la Lingüística Matemática y en particular en una ampliación semántica de las Gramáticas Generativas, con la cual es relativamente fácil **representar la estructura de múltiples elementos de la naturaleza**, por lo que prácticamente la misma rutina genera árboles, nubes, montañas, caracoles, estrellas y muchos otros componentes de la naturaleza.

I Una ecuación Fundamental

Las gramáticas generativas surgieron en 1957 a partir de los Trabajos de Noam Chomsky y desde principios de los 60's se han aplicado para representar la estructura de los múltiples lenguajes de programación que existen y poder construir los compiladores o intérpretes de estos lenguajes.

Por otro lado también desde los 60's ha sido la base de los métodos de reconocimiento sintáctico de patrones en los cuales se ve a una imagen o fenómeno que se repite cotidianamente (voz, movimiento de los planetas, jugada de ajedrez, etc.) como una 'oración' del lenguaje de imágenes o del fenómeno repetitivo, a partir de ahí se encuentra la regla gramatical o producción que representa la estructura general de este lenguaje y se asume entonces que por ejemplo, la regla gramatical de un conjunto de imágenes representa su patrón general o rasgos característicos.

Lo anterior permite desarrollar una herramienta que, a partir de la gramática de una familia de imágenes, es capaz de reconocer si una imagen en particular pertenece a la familia o no (como un compilador es capaz de reconocer si una oración pertenece o no a un lenguaje dado). Por otro lado, si se invierte el orden de entradas y salidas se puede tener un sistema capaz de generar imágenes específicas a partir de una gramática dada.

Un enfoque particular de lo anterior se presentó durante los 80's, al aplicar la Lingüística Matemática al reconocimiento y generación de componentes de la naturaleza como árboles, montañas y nubes con el fin de construir Sistemas Evolutivos de la Naturaleza (en donde un sistema evolutivo es un mecanismo capaz de encontrar, construir y mantener actualizada una representación del medio que lo rodea y usar esa representación para mantenerse y resolver problemas dentro de ese medio).

Específicamente el propósito consistió en lograr que un Sistema Evolutivo tomara la imagen de una familia de objeto de la naturaleza (árbol, montaña, etc.) encontrarla la regla gramatical o producción que representaba a la familia de objetos y fuera capaz de reconocer si un nuevo objeto pertenece o no a la familia. Como otro resultado además se encontró que esta era una forma muy simple de compactar objetos y de generar nuevas imágenes de la naturaleza.

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 más 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 tenía una producción general para árboles (para estructuras dendríticas) 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 más general aun, que las englobaba a todas.

Encontrándose que prácticamente la estructura gramatical de cualquier elemento de la naturaleza se puede ver como un caso particular de una regla general a la que llamamos **Ecuación Fundamental** y es de la forma:

$$S \rightarrow e^* S^*$$

Donde **S** significa Sistema, **e** es cualquier elemento del sistema (una rama un tronco, una ladera etc.), **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.

Ejemplos particulares de la ecuación fundamental son:

$$S \rightarrow e$$

$$S \rightarrow e S$$

$$S \rightarrow e S S$$

$$S \rightarrow e S S \dots S$$

$$S \rightarrow e S e e S e \dots S$$

II Árboles, Estrellas y Caracoles.

En particular la ecuación

$$S \rightarrow e$$

que se lee como **el sistema S llama a e**, representa la estructura de un programa que llama a un elemento, por ejemplo la rutina que gráfica un tronco:

```
Sistema
{
  tronco
}
```

o más específicamente:

```
Sistema (x0,y0,long,w)
{
  tronco(x0,y0,long,w)
}
```

donde x_0, y_0 representan el punto inicial del tronco, $long$ su tamaño y w el ángulo con el que crece. Como otro ejemplo podemos ver que la ecuación.

$S \rightarrow e S$

es equivalente a un sistema que genera un elemento y se llama recursivamente, con lo que, vuelve a generar otro elemento y así sucesivamente.

```
Sistema
{
  elemento
  Sistema
}
```

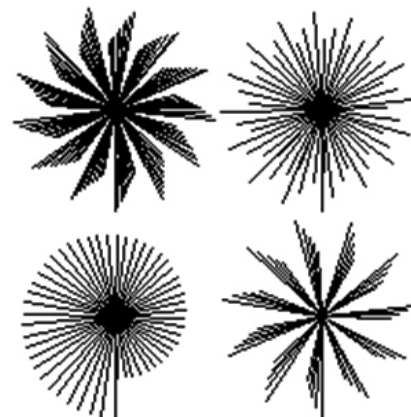
Esta ecuación engloba por ejemplo a la familia de las estrellas ya que la rutina

```
Sistema(x0,y0,long,w)
{
  elemento(x0,y0,long,w)
  sistema(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 una estrella.

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 el llamado recursivo únicamente representa la posibilidad de que surja una nue-



va rama o el sistema siga creciendo, sin embargo si esto se genera en una computadora es importante 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 numero prefijado de llamados.

En muchos casos es preferible una estructura iterativa en lugar de la recursiva, sin embargo existen problemas en los cuales la representación más simple es mediante procesos recursivos.

En el caso de rutinas que dibujan árboles, la estructura iterativa ya no es funcional y en cambio la estructura recursiva es muy simple, por ejemplo la ecuación:

$$S \rightarrow e S S$$

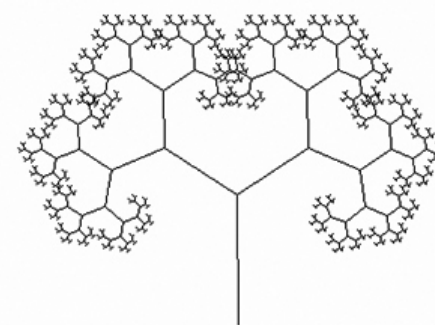
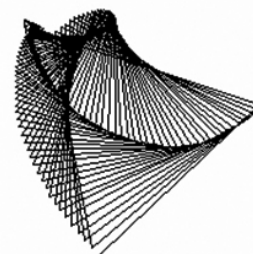
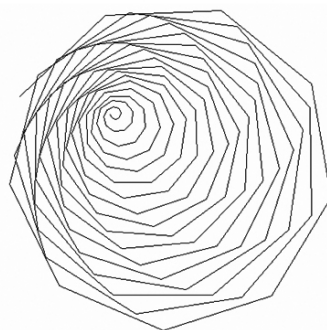
representa una familia de arboles de 2 ramas y el programa es de la forma:

```
main()
{
    árbol(x0, y0, long, ángulo, nivel);
}

Arbol(x,y,long,wg,nivel)           /*S*/
{
    if (nivel > 0) /* condición de terminación */
    {
        rama(x,y,long,wg,x1,y1); /*e*/
        Arbol(x1,y1,long/lon1,wg+w1,nivel-1); /*S */
        Arbol(x1, y1, long/lon2,wg + w2, nivel -1); /*S*/
    }
}
```

Donde lon1, lon2, w1 y w2 son parámetros que indican los cambios de tamaño y ángulo de las ramas.

Ahora bien, si se quiere 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:



$$S \rightarrow e \text{ SSS}$$

$$S \rightarrow e \text{ SSSS}$$

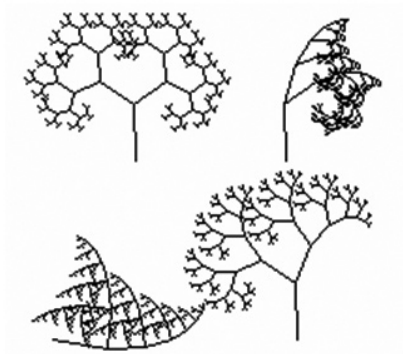
$$\dots$$

$$S \rightarrow e \text{ SS.....S}$$

y en general la estructura de cualquier árbol es de la forma

$$S \rightarrow e \text{ S}^*$$

y los programas son muy simples.



III Un Programa Generalizado

Construir un programa para cada tipo de árbol puede llegar a ser tedioso porque al final todos los programas son prácticamente idénticos y es más fácil pensar en un programa general al cual simplemente se le indique cuantas ramas se quieren y se llame a la rutina tantas veces como se necesite.

Por ejemplo, la rutina:

```

árbol(x,y,long,wg,nivel)
{
  if (nivel>0)
  {
    rama(x,y,long,wg,xl,yl)
    ind=1
    while ind <= n
      árbol(x1,y1,long/lon[ind],wg+w[ind++],nivel--)
    }
  }
}

```

representa precisamente a la ecuación

$$S \rightarrow e \text{ S}^*$$

Y dibuja árboles con n ramas (donde n es una variable que representa el número de ramas).

Otra forma alternativa de representación sería que en lugar de pasarle el número n de llamadas a la rutina, se le pasará una lista de la forma e S e e S S S e S con tantas e y S como se quiera.

La ventaja de este enfoque es que el sistema se puede generalizar para representar llamados a árboles “raros” como por ejemplo

$$S \rightarrow e \text{ S e}$$

$$S \rightarrow e \text{ S S e}$$

$$S \rightarrow e \text{ e S e S S}$$

$$S \rightarrow e S S e e S e$$

y cualquier combinación de elementos y llamadas, para lo cual únicamente se necesita preguntar si el comando es una **e** o una **S** como se ve a continuación:

```
gramática[] = "eSeeSS"

main()
{
  árbol(x0,y0,l0,w0,nivel)
}
árbol(x,y,long,wg,nivel)
{
  si (nivel>0)
  {
    i=1
    mientras (gramática[i] != fin de renglón)
    {
      si (gramática[i]='e') linea(x,y,l,w,x1,y1)
      si (gramática[i]='S')
        árbol(x1,y1,long/lon[i],wg+w[i],nivel--)
      i++
    }
  }
}
```

El anterior es un Sistema Generador de Árboles y representa a la estructura

$$S \rightarrow e^* S^*$$

si en el arreglo llamado gramática se almacena

- $S \rightarrow e$, el sistema genera una línea.
- $S \rightarrow e S$, el sistema genera caracoles o estrellas
- $S \rightarrow e S S$, el sistema genera árboles con dos ramas
- $S \rightarrow e S S S$, el sistema genera árboles con tres ramas

y así sucesivamente.

De donde resulta que el mismo programa que gráfica líneas, gráfica caracoles y árboles y todos son un caso particular de la estructura

$$S \rightarrow e^* S^*$$

Al anterior sistema se le pueden hacer múltiples modificaciones como:

- 1) Sustituir la rutina línea por una rutina que genere troncos de diferentes anchos, largos, colores, texturas, en forma de arco, etc.
- 2) Parametrizar el máximo de elementos como li , wi , y pasar diferentes valores para cada caso o aún más introducir secuencias aleatorias basadas en la distribución estadística de los elementos de una planta y en sus probabilidades de crecimiento, muerte o reproducción.
- 3) Introducir más de un tipo de elemento y generar con líneas, arcos, troncos círculos, triángulos, esferas u otros elementos.
- 4) Introducir múltiples estructuras y permitir que unas se llamen a otras, por ejemplo, tener la estructura de árbol, hoja y flor y que en cierto momento árbol llame a hoja o flor.

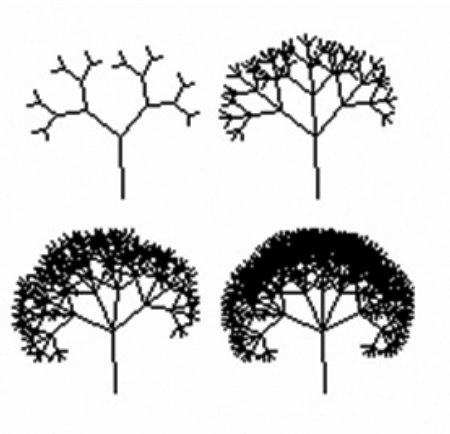
Con lo que un programa de este tipo es capaz de generar plantas de múltiples tipos únicamente cambiando la gramática que representa la estructura de la planta. Y para lograr lo anterior simplemente se requiere generalizar el programa para que pueda llamar al i -ésimo elemento (tronco, línea, arco, etc.) y a la i -ésima estructura o producción de la gramática.

En general la estructura de cualquiera de los sistemas anteriores es de la forma

$$Si \rightarrow ej^* Sk^*$$

donde ej es cualquier elemento del sistema y Sk cualquier producción de la gramática que representa al sistema.

Lo anterior nos proporciona una **herramienta generalizada para el desarrollo de sistemas** y en particular de árboles **a partir de la Lingüística Matemática**, ya que únicamente es necesario encontrar la gramática del sistema que se quiera producir y proporcionárselo al generador para que este obtenga el sistema específico con la estructura que se desee.



IV Árboles, Nubes, Montañas y Paisajes

Sin embargo el asunto no queda ahí ya que no se ha dicho prácticamente nada de los parámetros del sistema (En este caso x, y, l, w). Al empezar a jugar con los parámetros 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 ejemplo si tomamos el siguiente caso particular:


```

main()
{
  w0=c0; w1=c1; w2=c2; w3=C3; ind=cd;
  árbol (x0,y0,l,w0,ind)
}
árbol(x1, y1, l, w, ind)
{
  si ind>0
  {
    linea(x,y,l,w,x1,y1)
    árbol(x1,y1,l/1.2,w+w1,ind-1)
    árbol(x1,y1,l/1.55,w+w2,ind-1)
    árbol(x1,y1,l/1.8,w+w3,ind-1)
  }
}

```

donde w0, w1, w2, w3 son el ángulo inicial y los incrementos de ángulos para cada llamada recursiva.

Podemos encontrar que si fijamos

$$w0=-72 \quad w2=72 \quad w3=144$$

y modificamos w1 entre 1 y 360 grados podemos generar una familia completamente diferente, ya que si por ejemplo:

- 1) w1=24, tenemos un bosque de helechos.
- 2) w1=4, tenemos una nube en forma de perro.
- 3) w1=139, tenemos una nube.
- 4) w1=169, tenemos un fractal de dragón.

y así sucesivamente hasta regresar a los helechos.

Si el cambio del ángulo se realiza con saltos muy pequeños se puede observar como un elemento se transforma en otro, con lo que para generar un bosque de helechos o una nube solo es necesario cambiar el valor de un solo parámetro de todo un sistema.

El anterior efecto se produce si mantenemos fijos w0, w2, w3 y modificamos w1, ahora bien si modificamos cualquiera de los 4 ángulos la cantidad de elementos que se pueden generar es enorme e incluye nubes, árboles y montañas entre otros.

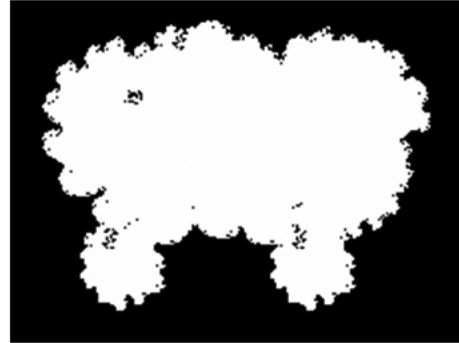
En general las ecuaciones de la forma

$$S \rightarrow e S S \dots S$$

permiten generar múltiples familias de árboles, montañas, nubes, ríos y prácticamente cualquier elemento de la naturaleza.

Por ejemplo si :

- 1) Fijamos $w_0=-27$, $w_1=6$, $w_2=172$, y modificamos w_3 se genera una familia de laderas de montaña.
- 2) Fijamos $w_0=-7$, $w_1=6$, $w_2=72$, modificamos w_3 se genera una familia de árboles en una ladera y árboles con reflejo.
- 3) Fijamos $w_0=-72$ $w_2=172$ $w_3=144$ y modificamos w_1 se genera una familia de alacranes



Como se puede ver, con una rutina de unos cuantos renglones y cambiando solo los ángulos se tiene prácticamente todos los elementos de un paisaje, por lo que, un programa que genera un paisaje se limita a un conjunto de llamados a la misma rutina, a la cual le pasan solo las modificaciones de los parámetros, como se puede ver en el siguiente ejemplo:

```
int w0,w1,w2,w3,color;
main()
{
    /*estrella */ color=3;
    w1=37; estrella(450, 90, 1, 27, 47);

    /*nube */ color=2;
    w1=238; w2=273; w3=144; paisaje(100, 80, 37, -72, 9);
    w1=188; w2=243; w3=144; paisaje(340, 110, 59, -72, 9);
    w1=238; w2=273; w3=125; paisaje(550, 80, 23, -23, 7);
    w1=287; w2=103; w3=144; paisaje(190, 60, 19, -92, 9);

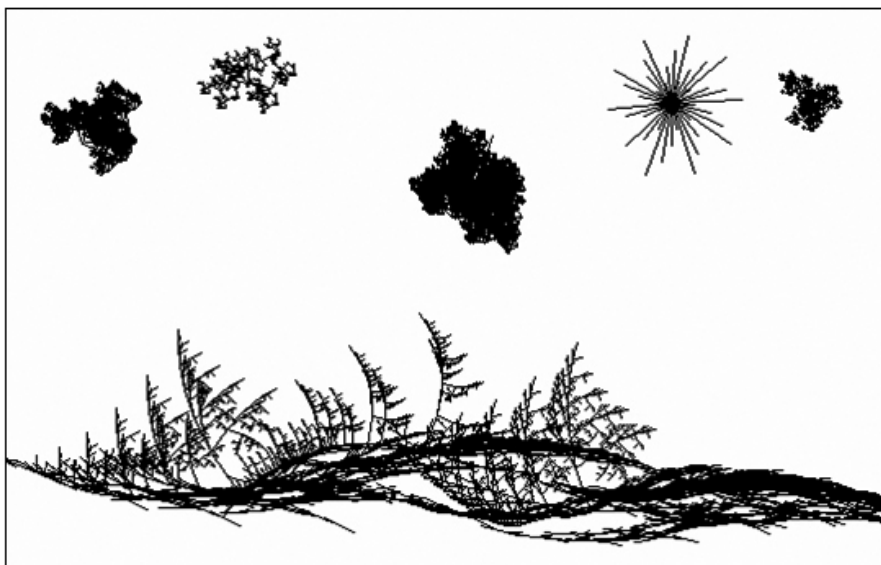
    /*montaña*/
    color=3; w1=6; w2=172; w3=186; paisaje(425, 350, 63, 173, 7);
    color=5; w1=6; w2=172; w3=1; paisaje(210, 320, 63, 169, 7);
    color=9; w1=6; w2=172; w3=186; paisaje(280, 325, 63, 8, 7);
    color=6; w1=6; w2=172; w3=1; w0=172; paisaje(380, 335, 63, w0, 7);
    color=6; w1=6; w2=172; w3=1; w0=173; paisaje(380, 335, 63, w0, 7);

    /*árbol */ color=1;
    w1=8; w2=72; w3=17; paisaje(225, 315, 43, -17, 7);
    w1=-12; w2=352; w3=97; paisaje(340, 315, 43, 15, 7);
    w1=8; w2=72; w3=351; paisaje(443, 340, 43, 0, 7);
}
paisaje(int x0, int y0, int l, int an, int ind)
{
    int x1,y1;
    if(Ind > 0)
    {
        setcolor(ind + color);
        recta(x0, y0, l, an, &x1, &y1);
        paisaje(x1, y1, l / 1.2, an + w1, ind - 1);
        paisaje(x1, y1, l / 1.55, an + w2, ind - 1);
    }
}
```

```

    paisaje(x1 ,y1, l / 1.8, an + w3, ind--);
  }
}
estrella(int x0, int y0, int l, int an, int ind)
{
  int x1, y1;
  if(Ind > 0)
  {
    setcolor(ind + color);
    recta(x0, y0, l, an, &x1, &y1);
    estrella(x0, y0, l + 1, an + w1, ind--);
  }
}

```



Conclusiones

En este documento se presentó una aplicación de la Lingüística Matemática a la generación de paisajes, para lo cual: en primer lugar se planteó que existe una ecuación fundamental de la naturaleza de la forma

$$S \rightarrow e^*S^*$$

que engloba a las ecuaciones que representan la estructura de troncos, caracoles, estrellas, árboles, nubes y montañas, entre otros. Lo anterior es fundamental ya que nos muestra que múltiples fenómenos de la naturaleza tienen la misma estructura.

Más adelante se vio como a partir de la estructura Lingüística se obtiene directamente un programa y en particular se presentó un sistema basado en la ecuación fundamental y se mostró como cambiándole simplemente algunos parámetros genera una gran cantidad de elementos.

Este enfoque generaliza y engloba dentro de la Lingüística Matemática múltiples herramientas como los métodos iniciador-generator y los de los sistemas-L, con lo que, se puede aprovechar todo el poderío de esta herramienta que se utiliza desde hace muchos años para construir compiladores, reconocer formas, medir la complejidad de sistemas y en general para encontrar y representar la estructura de múltiples problemas de la Informática, por lo que este documento es también una invitación al uso de la Lingüística Matemática dentro de la graficación y animación.

Como se puede ver estamos llegando a que la estructura de una cantidad enorme de elementos de la naturaleza es la misma, ya que, por ejemplo únicamente cambiando algunos ángulos dentro de una rutina se pueden generar nubes, laderas de montañas, árboles, alacranes, hojas y paisajes en general.

Agradecimientos

Muchas gracias a Marina Vicario Solorzano y Francisco Aguilar Vallejo de TECCIZ de México, Cuitlahuac Cantú de ORSA, Sergio Rivera, Francisco Molina C. y Cruz Luna Reyes de UPIICSA-IPN por su ayuda y contribución a este trabajo.

Referencias

- [1] Kenton Musgrave, Craig E. Kolb y Robert S. Mace, “The Synthesis and Rendering of Eroded Fractal terrains” In Computer Graphics, Vol 23(3), julio 1989.
- [2] Przemyslaw Prosinkiewicz, Aristid Lindenmayer y James Hanan. Developmental Models of Herbaceous Plants for Computer Imagery Purposes. Computer Graphics, Vol 22(4), Agosto 1988.
- [3] Sofia Bueno Peralta y Antonio Simancas López. Generador de Arboles Fractales. en Memorias del III Congreso Nacional sobre Informática y Computación, Jalapa, Ver. México, Octubre 1990.
- [4] Peter E. Oppenheimer. Real Time Design and Animation of Fractal Plants and Trees. In Computer Graphics, Vol 20(4) SIGGRAPH’86, August 1986.
- [5] Heinz-Otto Peitgen and Dietmar Saupe (Editores). The Science of Fractal Images. Ed. Springer-Verlag, 1988.
- [6] Fernando Galindo Soria. Sistemas Evolutivos: Nuevo Paradigma de la Informática. en Memorias XVII Conferencia Latinoamericana de Informática, Caracas Venezuela, julio de 1991.
- [7] Fernando Galindo Soria. Sistemas Evolutivos. en Boletín de Política Informática. México, Septiembre de 1986.
- [8] Rafael C. Gonzalez y Michael C. Thomason. Sintactic Pattern Recognition. Ed. Addison-Wesley.
- [9] Fernando Galindo Soria. Aplicaciones de la Lingüística Matemática y los Fractales a la Generación de Imágenes. en Memorias Simposium Nacional de Computación. México, Nov. de 1991.
- [10] Emmon Bach. Teoría Sintáctica. Ed. Anagrama.
- [11] Salomaa. Formal Languages. Ed. Academic Press.
- [12] Herbert. A. Simon. Las Ciencias de lo Artificial. Ed. ATE.
- [13] Noam Chomsky. Estructuras Sintácticas. Ed. Siglo XXI.
- [14] Hopcroft y Ullman. Formal Languages and Their Relation to Automata. Ed. Addison-Wesley.

III.7 Sistema Evolutivo Traductor

José Rafael Cruz Reyes⁷

Resumen

En este trabajo se desarrolla un Sistema Evolutivo capaz de aprender la estructura y significado de dos lenguajes diferentes para hacer la traducción entre ellos en forma automatizada. Para esto se utilizan las técnicas de Sistemas Formales, Sistemas Evolutivos, Aprendizaje, y Traducción por Computadora.

I La Realidad y los Sistemas de Información

El hombre siempre se ha visto maravillado por el comportamiento de la naturaleza y el Universo. Uno de los primeros hombres en creer que había descubierto el secreto de este comportamiento fue Pitágoras al observar lo que le sucedía a una cuerda al ser pulsada. Si la recortaba a la mitad de su longitud, esta volvía a emitir la misma nota que antes. De esta y otras observaciones Pitágoras concluyó que la armonía de la Naturaleza se encontraba en las razones entre los números enteros. Llegó a creer que si entendía estas razones entendería también la naturaleza del Universo, desde luego, pronto se convenció de que esto no era así, y finalmente su aportación real fue, el descubrimiento de que el Universo es un lugar peculiar, que no puede ser metido a la fuerza, en un esquema de ideas preconcebidas, no importa, lo elegante ni lo apremiantemente estático que tal esquema parezca ser [1].

Así, a través de la Historia los científicos al buscar las reglas generales que se ocultan detrás de los acontecimientos singulares, han visto que tales reglas no pueden ser elegantes y razonables, sino que conforme se va dando más profundidad al conocimiento de las cosas, este solo indica que aún falta más por conocer o comprender. Este proceso de la ciencia es un proceso evolutivo, es decir, es un proceso que requiere de estar actualizando la visión que se tiene de la realidad para así poder entenderle más cabalmente [2].

Actualmente se cuenta con herramientas muy poderosas que nos permiten proponer y valorar modelos cada vez más cercanos a la realidad, para así mejorar nuestra visión y entendimiento

⁷ José Rafael Cruz Reyes actualmente labora en el Aeropuerto Internacional de Toluca, además es profesor del Instituto Tecnológico De Toluca (ITT) en la División de Posgrado e Investigación. Este trabajo lo realizó como parte de su tesis de Maestría en Ciencias de La Computación en el mismo ITT en 1995.

de la naturaleza. Una de estas herramientas son los sistemas artificiales creados por el hombre y especialmente los sistemas de la Informática llamados Sistemas Evolutivos de que se trata en este trabajo [3][11].

II El Aprendizaje en los Sistemas de Información

A lo largo del tiempo en que se han desarrollado sistemas de Información una de las últimas funciones en integrarse a ellos es la de aprender y automodificarse (enfoque evolutivo). Tradicionalmente al desarrollar un sistema, se estudia el problema a resolver, se propone uno o varios modelos de comportamiento, y finalmente se implementan en un sistema capaz de poner a prueba el modelo desarrollado y con ello dar la posibilidad de resolver problemas específicos. Con esto, tenemos que conforme sea más complejo el problema, y el modelo correspondiente presente mayor dificultad en su desarrollo, el sistema que resulte podrá tener carencias o limitaciones, y en realidad, esta es la situación con respecto a la mayoría de los sistemas en operación.

Una solución a estas limitaciones bien puede ser el dotar al sistema en operación de la posibilidad de aprender del problema las expresiones (o casos) no previstas durante su desarrollo y aplicarlos al modelo que rige al sistema de tal manera que este pueda modificar sus características y así ampliar sus posibilidades [11].

Este enfoque se presenta como uno de los principales objetivos del proyecto de las computadoras de la Quinta Generación propuesto por Japón en los inicios de la década de los 80's. En México, el grupo de investigación encabezado entre otros por Fernando Galindo Soria, desde 1985 ha desarrollado y aplicado en forma muy particular este enfoque a diferentes problemas, partiendo del hecho de que si a un sistema se le da la capacidad de aprender y de automodificarse internamente para responder mejor a los problemas que se le presentan, es posible que el proceso inicie con la construcción de un modelo de la realidad desde una etapa muy básica dando como consecuencia que dicho modelo refleje fielmente la realidad, que es donde se encuentran los problemas a resolver, dando como resultado el evidente ahorro en cuanto a tiempos de desarrollo y de mantenimiento del sistema.

A los sistemas de información que contienen los mecanismos internos que le permitan captar la realidad, representarla internamente y que con esta representación son capaces de resolver los problemas inmersos en esa realidad, se les conoce con el nombre de Sistemas Evolutivos [15]

III Antecedentes de la Traducción por Computadora

La historia de la traducción automática se inicia prácticamente junto al desarrollo de las computadoras, en los años 50's, siendo entonces cuando la gente se preguntaba hasta que punto sería posible usar esas máquinas para traducir [4]. Ahora sabemos que pueden ser una herramienta muy útil aunque casi nadie piensa en utilizarlas para traducir poesía, lenguajes antiguos escritos y jeroglíficos (como los prehispánicos en América), etc.

Existen trabajos y productos muy importantes pero en todos estos casos la traducción se lleva a cabo con la ayuda de mecanismos basados en estructuras rígidas que no permiten su actualización por parte del usuario (a excepción por supuesto de los respectivos diccionarios de palabras) salvo la presente opción que da lugar a este trabajo.

IV El Proceso de Traducción por Computadora

El proceso de traducción de un lenguaje en otro consiste en la identificación de las palabras y su significado en un lenguaje y de la obtención de las equivalentes en el idioma destino, tomando como requisito que en ambos idiomas la idea expresada sea básicamente la misma.

Desde luego, la dificultad no consiste en traducir palabra por palabra recurriendo a un diccionario almacenado, tampoco el problema estriba en dar a la máquina un párrafo exclusivamente en cierto idioma para que ello, genere su correspondiente en otra lengua. Más bien, actualmente nos encontramos en la etapa de entregarle a la máquina una serie de algoritmos y relaciones para que pueda pasar una expresión en un idioma a su traducción en la lengua deseada. Es claro que para alimentar a una computadora con esas reglas es necesario primeramente llegar a un nivel suficiente de formalización de las normas gramaticales y eso para un hombre requiere de esfuerzos y tiempo.

Un enfoque alternativo utilizado es limitar el campo de acción del traductor para hacer uso de algoritmos sencillos que identifiquen e interpreten las palabras de una lengua para producir sus equivalentes en otra. Una vez efectuada la traducción término a término se deben aplicar consideraciones previamente programadas para reacomodar las palabras y que estas, en la lengua destino, tengan el mismo significado que en la lengua origen. Desde luego se mantiene vigente la posibilidad de que la computadora en todo tiempo pueda cuestionar al usuario con la finalidad de aclarar algún significado, en el caso de equivalencias múltiples, o de escoger el que más conviene al caso particular.

Se debe hacer notar que en cualquiera de los casos anteriores es necesario tener como desarrolladores un profundo conocimiento de las reglas y relaciones (gramáticas) de ambos idiomas o, como es el caso más deseable, contar con el respaldo de un grupo interdisciplinario compuesto por un conocedor experto de la materia específica en la que se desean traducir textos, de un lingüista y de los especialistas en informática.

V Sistema Evolutivo Traductor

El Sistema Evolutivo propuesto posee capacidades que se activan en diferentes momentos para afrontar el problema de la traducción entre dos idiomas. Se emplean ejemplos de la traducción entre el idioma inglés y el español para ilustrar las capacidades, aunque estos pueden aplicarse a cualesquiera otros dos idiomas.

El primer caso se tiene en las frases que expresan ideas que no se traducen bajo esquemas gramaticales similares, ni de palabra por palabra, sino que se toman tal como se expresan, como

las frases metafóricas usadas en poesía, refranes y similares (tabla 1).

IDIOMA 1	IDIOMA 2
to rain cats and dogs	Llover a cántaros
to ring up	Llamar por teléfono
upside down	Cabeza abajo
have a good time!	¡que se divierta!

Tabla 1. Tabla de reescritura

Para este tipo de expresiones se utilizan las técnicas de los Sistemas Evolutivos de Reescritura, en los cuales cuando un patrón empata con el lado izquierdo, se sustituye por el texto del lado derecho.

Los párrafos que quedan sin reescribirse se dividen en unidades léxicas y se generan las oraciones canónicas para cada uno, ejemplo:

The brown dog runs in the park

a j s v p a s

donde a=artículo, j=adjetivo, s=sustantivo, v=verbo, p=preposición

y la oración canónica correspondiente al ejemplo anterior es:

a1 j1 s1 v1 p1 a2 s2

Las oraciones canónicas pasan por un proceso de transformación aplicando gramáticas transformacionales, en donde una de las reglas que el mismo sistema puede obtener es:

$a j s \rightarrow a s j$

Al aplicar esta transformación al ejemplo anterior se obtiene:

a1 s1 j1 v1 p1 a2 s2

Al resultado se le aplican reglas de declinación para asegurar la correcta coincidencia en género, número y otros accidentes gramaticales en las palabras que forman una oración o frase, el resultado para el ejemplo anterior es:

El perro café correr en el parque

Una regla de declinación puede ser:

$s1 (3p) v1 \rightarrow s1 (3p) v1 (3p)$

donde 3p=tercera persona

Las transformaciones se aplican sobre las oraciones y al resultado, es decir las palabras resultantes, se procesan con un diccionario.

Como resultado del proceso anterior puede suceder que algunos párrafos o grupos de palabras aún no se tenga su traducción, por lo que en ese momento se da el proceso de aprendizaje asignando entradas en las tablas de reescritura, en la gramática transformacional, dando de alta palabras en el diccionario, generando una regla de declinación, o bien, actualizando cualquiera de los existentes. Este proceso de aprendizaje y reestructuración lo apoya el mismo Sistema Evolutivo que cuenta con capacidades para esto.

Conclusiones

El ser humano siempre ha buscado herramientas que lo ayuden en sus diferentes actividades, al inicio fueron herramientas para cortar, cazar, moler, etc., luego máquinas que lo ayudaron en la construcción de sus ciudades, para desplazarlo rápido de un lado a otro, cortar con más precisión, etc., actualmente, intenta construir herramientas que lo ayuden en su labor intelectual como la traducción automática que le permitirá estar mejor comunicado con los habitantes del único planeta en que conocemos vida: La Tierra. Hemos visto como promisorio el enfoque Evolutivo, como el mejor conjunto de técnicas para afrontar este problema, del que nuestros resultados hasta ahora son satisfactorios.

Bibliografía:

- 1] Stableford, Brian M, “Los Misterios de las Ciencias Contemporaneas”, ed. FCE 1^a.edición, México 1985.
- [2] Lehman, M., “El pensamiento y la Acción Humanos como Componente del Comportamiento de los Sistemas” en Enciclopedia de la Ignorancia compilador Ronald D. Duncan, Ed. FCE 1985.
- [3] Galindo Soria, Fernando, “Sistemas Evolutivos”, IPN-UPHCSA México 1985.
- [4] Tejeda Martínez, A., “Traducción por Computadora” en ICYT, Ed. CONACYT vol. 7 Núm. 109 México 1985.
- [5] Camacho Villanueva, Sandra, Gómez Rendón, Guadalupe Patricia, Olivares Ceja, Jesús Manuel, “SI-VE: Sistema de Visión Experto”, IPN-UPHCSA México 1986.
- [6] Cordero Sánchez, Gabriel, “Aplicación de un Reconocedor de Lenguaje Natural Restringido a la Recuperación de Datos”, IPN-UPHCSA México 1989.
- [7] Berruecos Rodríguez, Elsa, “Sistema Evolutivo Generador de Esquemas Lógicos de Bases de Datos”, IPN-UPHCSA México 1990.
- [8] Ortiz Hernández, Javier, Galindo Soria, Fernando, “Sistema Evolutivo Constructor de Sistemas Expertos”, CENIDET-UPHCSA Cuernavaca México 1990.
- [9] Benton, Peter M., “The Multilingual Edge”, en Byte, March 1991.
- [10] Olivares Ceja, Jesús Manuel, “Sistema Evolutivo para Representación del Conocimiento”, IPN-UPHCSA México, abril 1991.
- [11] Galindo Soria, Fernando, “Sistemas Evolutivos: Nuevo paradigma de la Informática”, Memorias de la XVII Conferencia Latinoamericana de Informatica, Caracas Venezuela. julio de 1991.
- [12] Olicón Nava, Carlos, “Sistema Evolutivo Generador de Paisajes”, IPN-UPHCSA México 1991.
- [13] Morales Rubio, Angel Cesar, “Sistema Evolutivo para Tratamiento de Imágenes”, IPN-UPHCSA México, 1992.
- [14] Aguilar y Aguilar, Araceli, “Sistema Evolutivo Generador de Sistemas”, IPN-UPHCSA, México 1992.
- [15] Galindo Soria, Fernando, “Sistemas Evolutivos Basados en Conocimiento”, IPN-ESCOM México 1994.
- [16] Galicia Haro, Sofía, “Correcciones de estilo en Lenguajes Naturales” en Soluciones Avanzadas Abril 1995.

CAPÍTULO IV

MATRICES EVOLUTIVAS

IV. 1 Una Representación Matricial para Sistemas Evolutivos

Fernando Galindo Soria

Introducción

Cuando se está trabajando en un área, es muy emocionante encontrar que problemas aparentemente diferentes se pueden ver como casos particulares de un problema más general, esto es lo que sentí cuando al estar comentando con algunos alumnos unos trabajos sobre redes neuronales desarrollados junto con Gustavo Nuñez Esquer en 1976, me di cuenta de que las herramientas encontradas, eran idénticas a otras que había descubierto en forma independiente Cuitlahuac Cantú a finales de los 80's para el tratamiento de imágenes y en general de formas.

Esto fue aun más impactante porque en el trabajo de 1976 hallamos una matriz para representar redes neuronales y en el de Cuitlahuac se plantea una herramienta evolutiva, por lo que al conjugar las dos ideas bajo la premisa de una herramienta común, por un lado se refuerzan las redes neuronales para el tratamiento de formas y por otro lado se llega a una herramienta más potente al integrarle en forma natural el enfoque evolutivo, con lo que las matrices evolutivas surgen como un mecanismo de representación generalizado y extremadamente poderoso ya que absorben en forma natural características de las redes neuronales y de los sistemas evolutivos.

Por lo anterior, ya valdría la pena el estudio de las matrices evolutivas, pero en 1992 al estar revisando un trabajo sobre sistemas evolutivos generadores de sistemas expertos, desarrollado junto con Javier Ortiz en el CENIDET de Cuernavaca en 1988, me di cuenta que las reglas de inferencia de un sistema experto se puede representar directamente mediante una Matriz Evolutiva, de donde se pueden plantear sistemas expertos como un caso particular del reconocimiento de formas, que son problemas equivalentes al reconocimiento de imágenes o señales y que se pueden manejar mediante redes neuronales.

A partir de lo anterior, se puede plantear que existe una familia de problemas equivalentes entre si y que se pueden representar como una matriz evolutiva, incluyendo problemas de áreas como el tratamiento de imágenes, voz y otros tipos de señales donde se mantiene el orden entre los elementos de un objeto y problemas en los cuales no es importante el orden en que se presentan los objetos, o sea problemas donde se permite la conmutabilidad entre los elementos.

I Una Representación Matricial para Redes Neuronales

Durante el año de 1976 en el Centro Nacional de Cálculo (CENAC) del IPN se organizó en colaboración con la Escuela Superior de Física y Matemáticas (ESFM) del mismo instituto un Seminario sobre Inteligencia Artificial y uno de los temas tratados dentro de este seminario fue el de Redes Neuronales, en particular, una de las actividades en las que participé junto con Gustavo Nuñez Esquer fue en el desarrollo de una propuesta para representar una Red Neuronal, para lo cual a partir del modelo de neurona desarrollado por Mc Cullot y Pitts generamos nuestra propia propuesta.

Para lo cual partimos de que, una neurona de Mc Cullot y Pitts se representa como:

$$\begin{array}{ll} \text{Si} & s_0 a_0 + s_1 a_1 + s_2 a_2 + \dots + s_n a_n \geq h \\ & \text{entonces } s_{\text{axon } 1} = 1 \\ & \text{sino } s_{\text{axon } 2} = 0 \end{array}$$

Y una Red Neuronal se puede ver como:

$$\begin{array}{ll} \text{Si} & s_{01}^a a_{01} + s_{11}^a a_{11} + s_{21}^a a_{21} + \dots + s_{n1}^a a_{n1} \geq h_1 \\ & \text{entonces } s_{\text{axon } 1} = 1 \\ & \text{sino } s_{\text{axon } 1} = 0 \\ \\ \text{Si} & s_{02}^a a_{02} + s_{12}^a a_{12} + s_{22}^a a_{22} + \dots + s_{n2}^a a_{n2} \geq h_2 \\ & \text{entonces } s_{\text{axon } 2} = 1 \\ & \text{sino } s_{\text{axon } 2} = 0 \\ \\ \text{Si} & s_{03}^a a_{03} + s_{13}^a a_{13} + s_{23}^a a_{23} + \dots + s_{n3}^a a_{n3} \geq h_3 \\ & \text{entonces } s_{\text{axon } 3} = 1 \\ & \text{sino } s_{\text{axon } 3} = 0 \\ \\ & \text{" " " " } \\ & \text{" " " " } \\ & \text{" " " " } \end{array}$$

$$\begin{array}{ll} \text{Si} & s_{0m}^a a_{0m} + s_{1m}^a a_{1m} + s_{2m}^a a_{2m} + \dots + s_{nm}^a a_{nm} \geq h_m \\ & \text{entonces } s_{\text{axon } m} = 1 \\ & \text{sino } s_{\text{axon } m} = 0 \end{array}$$

Y a partir del modelo anterior, desarrollamos un modelo matricial de red neuronal en el 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 dentro de la matriz, 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, como se puede ver en el siguiente ejemplo, donde la red neuronal:

Se representa como:

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	h			
-1	1	1	0	0	0	0	0	1	s^1		x_1
0	0	0	-1	-1	1	0	0	1	s^2	=	x_2
0	0	0	0	0	0	1	-1	1	s^3		x_3
									s^4		
									s^5		
									s^6		
									s^7		
									s^8		
									-1		

donde

sí $x_1 \geq 0$ entonces	$s_7=1$
sí $x_2 \geq 0$ entonces	$s_8=1$
sí $x_3 \geq 0$ entonces	$s_9=1$

O sea que la matriz representa a la red neuronal, el vector a las señales de entrada a la red (s_i), cada renglón de la matriz representa una neurona (x_j) y cada columna una señal de entrada a la neurona (a_k).

La matriz se multiplica por el vector de señales de entrada al sistema y si el valor resultante es mayor o igual a cero entonces se considera que la neurona es exitada por lo que se genera un valor de salida de la neurona igual a uno.

II Matriz Evolutiva y Reconocimiento de Formas

El anterior es 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 su importancia es mayor, ya que durante los 80's Cuitlahuac Cantu director del Grupo ORSA desarrolló en forma también independiente un Sistema Evolutivo para reconocimiento de formas, en el cual representa los objetos a reconocer por medio de una matriz que es prácticamente igual a la encontrada en 1976.

El sistema evolutivo desarrollado por Cuitlahuac consta de varias partes y una de ellas es un mecanismo para representar los objetos que esta reconociendo. Es esta componente la que nos interesa en este trabajo.

Como primer punto ser parte de que, cualquier objeto se puede representa como un vector, de tal forma que por ejemplo, un gato se puede almacenar en un vector como el siguiente:

1 0 1 0 0 1 0 1 0 gato

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

Donde el primer renglón representa al gato, el segundo al perro y el tercero al ratón.

Si en lugar de un gato 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 (es importante recalcar que 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.

Cualquier objeto nuevo que llegue se representa también como un vector, como por ejemplo:

1 0 1 0 0 1 0 1 0

Si se desea reconocer el nuevo objeto, se multiplica el vector que lo representa por la matriz, se ve en que renglón se obtuvo el máximo valor y se le asocia a ese renglón el objeto.

Por ejemplo tomando la matriz anterior y el objeto desconocido, se multiplica su vector por la matriz y vemos que su valor es:

18 para el gato
4 para el perro
2 para el ratón

Si a estos valores les restamos el valor de **h** nos queda:

-2 para el gato
-10 para el perro
-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, el sistema evolutivo esta transformando en forma permanente su imagen de la realidad, de tal forma que conforme pasa el tiempo, se espera que sea cada vez capaz de reconocer mejor los objetos.

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 generalizado para 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). Lo anterior es la base de un proyecto desarrollado por Alejandrina Salazar Torres sobre redes neuronales evolutivas en 1993.

III Matriz evolutiva y Generadores de Sistemas Expertos

El hecho de que en dos problemas independientes se llegó al mismo mecanismo de representación es interesante, sin embargo estas no son las únicas veces en que se ha presentado esta matriz, por lo que se plantea la posibilidad de estar ante una herramienta generalizada.

Durante el año de 1988 junto con Javier Ortiz (en esa época Coordinador de la Maestría en Computación del CENIDET en Cuernavaca, Mor.) desarrollamos un sistema evolutivo generador de sistemas expertos, para lo cual tomamos en cuenta que, una de las características de los sistemas de diagnóstico y toma de decisiones (decision support systems —DSS—, reconocedores de patrones, etc.) en general y de los sistemas expertos en particular se encuentra en que el lenguaje del usuario consta principalmente de oraciones de la forma:

Síntomas: Diagnóstico: Acciones o Tratamiento

Donde este tipo de oraciones nos permiten plantear, por ejemplo, la base de un conjunto de reglas de inferencia o el patrón general de cierto tipo de figuras.

A partir de lo anterior, se detectó que el problema de construir un sistema experto o en general de toma de decisiones, y específicamente la etapa de ingeniería de conocimiento se podía automatizar en buena medida, ya que el problema clásico de encontrar las reglas de inferencia fue sustituido por el problema mas general de encontrar un conjunto de ejemplos significativos de diagnóstico o de toma de decisiones reales, en los cuales se encontraran precisamente inmersos los síntomas, diagnósticos y acciones.

Durante 1992 al estar preparando unos ejemplos de Lingüística Matemática, detecté que una forma alternativa para representar el conjunto de oraciones del sistema era precisamente mediante una matriz evolutiva, por ejemplo, si se tiene el siguiente conjunto de oraciones:

a b c d e f	: D1	: T1 T2 T3
a b x y c m	: D2	: T2 T4
a b k l	: D1	: T1 T2 T6
d m l f g	: D4	: T6
a b k o	: D2	: T7 T8

Donde las letras minúsculas representan los síntomas, las D's los diagnósticos y las T's los tratamientos.

Una forma directa de representarlas seria mediante la matriz:

a	b	c	d	e	f	g	k	l	m	o	x	y	
1	1	1	1	1	1	0	0	0	0	0	0	0	D1: T1 T2 T3
1	1	1	0	0	0	0	0	0	1	0	1	1	D2: T2 T4
1	1	0	0	1	0	0	1	1	0	0	0	0	D1: T1 T2 T6
0	0	0	1	0	1	1	0	1	1	0	0	0	D4: T6
1	1	0	0	0	0	0	1	0	0	1	0	0	D2: T7 T8

El funcionamiento de esta matriz es prácticamente igual al del mecanismo de representación de los objetos del Sistema Evolutivo propuesto por Cuitlahuac.

Al analizar las reglas de inferencia se nota que una de las característica que permite su representación 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 y por otro lado por ejemplo las cadena en Español no se podrían representar en general mediante la matriz, porque no se permite la conmutabilidad.

Conclusiones

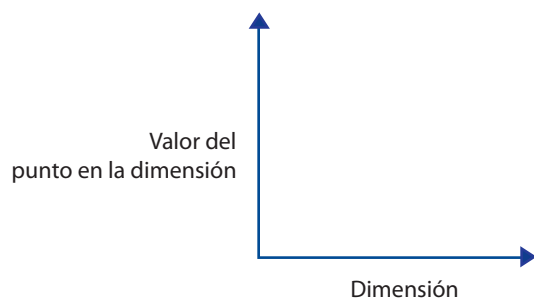
En este documento se integraron trabajos desarrollados en el Instituto Politécnico Nacional (IPN), Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) y por Cuitlahuac Cantú del Grupo ORSA, y **se mostró como tres de las grandes áreas de la I.A.: Reconocimiento de Imágenes y en general de formas, Sistemas Expertos y Redes Neuronales se pueden apoyar en la representación mediante Matrices Evolutivas.**

IV.2 Transformación al Espacio Dimensión-Valor

Fernando Galindo Soria¹

I La Gráfica de un Punto en un Número Continuo de Dimensiones

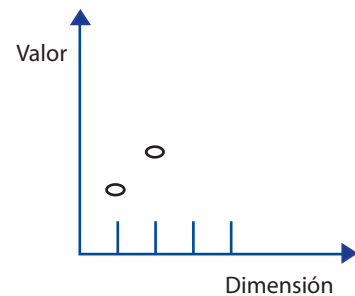
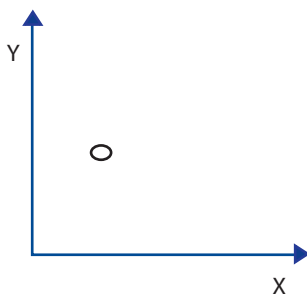
La gráfica de un punto en un número continuo de dimensiones es una línea en el espacio (dimensión, valor)



Ejemplos de explicación:

a) Punto en dos dimensiones

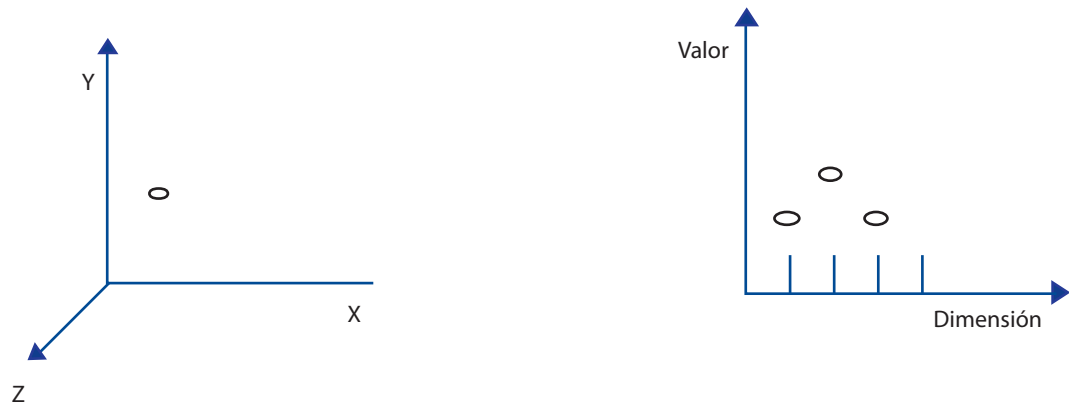
X	Y
3	5



¹ Fernando Galindo Soria ESCOM-IPN 20 de Abril de 1995.

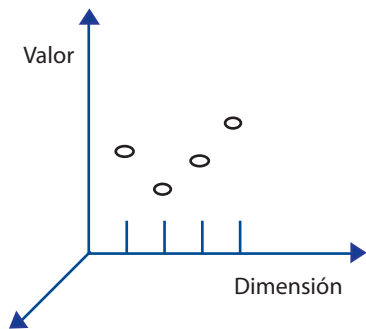
b) Punto en tres dimensiones

X	Y	Z
3	4	3



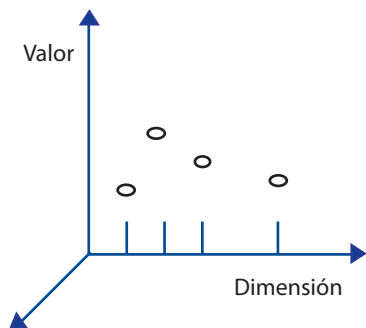
c) Punto en cuatro dimensiones

X	Y	Z	K
5	3	4	6



d) Punto en n dimensiones

d1	d2	d3	...	dn
2	4	3		2



Se esta realizando una transformación del espacio de n dimensiones (o en su caso de un número continuo de dimensiones a el espacio (dimensión, valor).

En la transformación del espacio de n dimensiones y el espacio (dimensión, valor) un punto se ve como n puntos en el espacio (dimensión, valor).

Cambio del espacio de n dimensiones al espacio <dimensión, valor>

$$\langle d_1, d_2, \dots, d_n \rangle \rightarrow \langle d, v \rangle$$

Un punto con un número continuo de dimensiones en un espacio <dimensión, valor> se puede ver como un conjunto transfinito de puntos, segmentos de recta o en el mejor de los casos una recta (pero al final es una curva, no necesariamente continua).

Se podrían reordenar las dimensiones para formar grandes segmentos se ordenan los valores de menor a mayor y se reordenan las dimensiones.

Se obtiene una recta o un conjunto de segmentos de recta (en un continuo de dimensiones) alineados con cierta pendiente.

En general un punto en el espacio <dimensión, valor> se puede ver como una curva formada por múltiples segmentos de curva y en el mejor de los casos como una curva continua.

II Representación de un Punto en N Dimensiones

Viéndolo al revés, una recta formada por n puntos se puede ver como un punto en n dimensiones (lo mismo para línea continua \Rightarrow punto en un continuo dimensional).

- m rectas son n puntos pero la línea de n puntos se puede ver como un punto en un espacio de $n \times m$ dimensiones.

-un continuo de rectas forman un área.

-el área esta formada por m rectas (donde $m \rightarrow X_c$ (aleph c)),

-volumen S área S = Punto

-en espacio $\langle d_1, d_2, d_3, \dots, d_n \rangle$

- La curva que representa una señal (de sonido, luz, presión, temperatura, electrocardiograma, etc), se puede ver como un punto en un continuo de dimensiones.

-O sea que una imagen, sonido, etc. lo puedo representar como un punto en un continuo de dimensiones y cuando se digitaliza lo que se obtiene es una representación como un punto en n dimensiones.

-En el proceso de digitalización conforme aumenta la cantidad de dimensiones tiende a representar cada vez mejor al objeto real.

-Si un objeto lo represento como un punto en una dimensión.

- Si tengo varios objetos y cada objeto se representa 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 quiero representar a los alumnos de un grupo por su calificación y suponemos una distribución uniforme de las calificaciones entre 0 y 10 y puros valores enteros entonces x pertenece a $[0, 10]$ puede tomar 11 posibles valores.

-Si tengo un alumno puede tomar cualquier valor entre 0 y 10.

-Si tengo dos alumnos cada uno puede tomar un valor entre 0 y 10 y la probabilidad de que colisiones es de $2/11$.

A la relación que existe entre el número de valores posibles $[0, 10]$ y el número de objetos (alumnos) en este caso 2, se le conoce como densidad y en general.

Si $[a, b]$ es el número de valores posibles entonces

$|b - a| + 1$ (la distancia entre a y b) nos da el rango de valores y se denota por μ

$$\mu = |b - a| + 1$$

en el ejemplo de las calificaciones

$$\mu = |10 - 0| + 1 = 11$$

la variable probabilística puede tomar 11 posibles valores

Por otro lado al numero de objetos se les denota por λ

En el ejemplo anterior $\lambda = 2$

la densidad se define como $\rho = \lambda/\mu$

En el ejemplo $\rho = 2 / 11$

La densidad entre otras cosas me indica que tan factible es que dos objetos colisiones (o sea que tomen el mismo valor)

Por ejemplo si $\mu = 11$ y tengo

- 1 persona $\rho = 1 / 11$

- 2 personas $\rho = 2 / 11$

Si tengo 10 personas, entonces $\rho = 10 / 11$ y es muy posible que dos personas tomen el mismo valor, si tengo 20 personas, es seguro que varias personas tomen el mismo valor

Por lo que si quiero distinguir entre varios objetos midiendo 1 sola variable (suponiendo distribución uniforme) se necesita que $\rho \ll 1$ o sea que el número de objetos sea menor que el número de valores posibles.

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 suponiendo que la probabilidad de tomar cualquier valor es la misma).

Si el rango de altura es:

[1.50, 1.70] cm.

$$\mu = |169 - 150| + 1 = 20$$

si tengo pocos alumnos funciona pero ya para más de 15 truena.

En general la curva de rho es una curva con un punto de inflexión en .75, antes de .75 la curva crece lento y después crece muy rápido.

Por lo que en general si

$\rho < .75$ los datos colisionan poco.

Por lo que si tengo menos de 15 alumnos la variable altura es fiable para discriminar.

Ahora bien un fenómeno muy hermoso surge si utilizo las dos variables combinadas (dos dimensiones) ya que si las variables no están correlacionadas el espacio probabilístico se multiplica o sea que si categorizo a un conjunto de objetos mediante 2 variables independientes entonces si μ_1 es el número de puntos en x , y μ_2 es el número de puntos en y entonces $\mu_1 \mu_2$ es el número de puntos en el espacio probabilístico.

Por ejemplo en el caso de los alumnos:

$$\mu_{\text{calificación}} = 11$$

$$\mu_{\text{tamaño}} = 20$$

$$\mu_{\text{espacio}} = \mu_{\text{calificación}} * \mu_{\text{tamaño}} = 11 * 20 = 240$$

Con lo que las colisiones se reducen drásticamente ya que aún con un grupo de 50 alumnos.

$$50 < 220 * .75 = 165$$

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

Ahora bien recordando el inicio del trabajo una señal u objeto lo puedo ver como un punto en n dimensiones (al digitalizar cada valor digitalizado corresponde al valor de una dimensión) por lo que una imagen de 600 x 500 pixels corresponde a un punto en un espacio de 300000 (trescientos mil) dimensiones.

La posibilidad de que dos objetos diferentes colisiones puede llegar a ser despreciable y si aumentamos el color como otra variable tendríamos un punto en 600 x 500 x 256 (suponiendo 256 colores).

Podemos discriminar sin problemas alrededor de 50 millones de objetos.

Ahora bien si dejamos de suponer una distribución uniforme y suponemos una distribución normal se tiene que el comportamiento del espacio probabilístico para una variable es de forma normal.

Si tenemos dos variables se tiene un espacio donde se interceptan dos curvas normales.

El espacio con centro en $(x.media, y.media)$ y que tiene como radios la varianza de x y la varianza de y se conoce como elipse de Chebichev.

Conforme aumenta la cantidad de variables se termina teniendo una hiperelipse de Chebichev de n dimensiones con la característica de que todos los puntos de la hiperelipse se pueden ver como casos particulares de un mismo objeto representado por el punto medio $(d_1, d_2, d_3, \dots, d_n)$ de la elipse.

O sea que todos los puntos “cercaños” a $(d_1, d_2, d_3, \dots, d_n)$ se pueden considerar como ejemplos del mismo objeto.

De donde una técnica para reconocer un objeto o fenómeno o patrón de algún tipo consiste en encontrar su representación “promedio” en un espacio de n dimensiones y todos los objetos cercaños (que caen dentro de la elipse de Chebichev) se pueden considerar como el mismo objeto.

Este enfoque tiene una aplicación inmediata en el caso del reconocimiento de imágenes ya que si por ejemplo se quiere distinguir entre varias imágenes (supondremos blanco y negro por facilidad) cada imagen es un punto en un espacio de 300000 puntos.

La implementación computacional se vuelve trivial ya que lo único que se necesita es un mecanismo que obtenga las imágenes promedio pero eso es trivial ya que si cada imagen la almaceno como un vector de n entradas y simplemente voy sumando casos particulares de cada imagen pixel a pixel y llevando un contador al final el promedio es directo.

Entre mas puntos se tengan mas rápidamente se discrimina y la convergencia es mas rápida ya que con una muestra significativa (aproximadamente 90 casos) se tiene una buena representación.

Si esto se integra en un sistema evolutivo la convergencia es en tiempo real.

IV.3 Diseño y Construcción de Sistemas Interactivos aplicando experiencias basadas en el Tratamiento de Imágenes

Fernando Galindo Soria²
Jesús Antonio Jiménez Aviña³

Resumen

La Interactividad estudia los mecanismos para lograr que dos o más dispositivos o sistemas se relacionen recíproca y simultáneamente entre sí, de tal manera que las acciones e información generada por unos repercutan en los otros. Es un área en la que se manejan cientos de miles de millones de dólares y en la que se involucran la robótica, realidad virtual, sistemas evolutivos, telecontrol, simuladores, biótica, multimedia, sistemas de comunicación, control distribuido, música por computadora, etc.

La Interactividad es una de las áreas de mayor desarrollo a nivel mundial, sin embargo en nuestras escuelas prácticamente no existen cursos sobre este tema, lo anterior es en parte por desconocimiento y en parte porque se cree que requiere muchos recursos. Lo último es relativamente cierto, por ejemplo, si se quiere realizar un sistema de realidad virtual o de reconocimiento de voz normalmente no se cuenta con los recursos necesarios para hacerlo.

Sin embargo, al analizar múltiples sistemas interactivos nos damos cuenta que manejan componentes parecidos y que es relativamente fácil aplicar los conocimientos adquiridos al desarrollar unos sistemas en el desarrollo de otros.

Se podría decir que es distinto manejar sonido, presión, temperatura o imágenes, pero lo único distinto es la forma como lo percibimos, ya que si manejamos imágenes tendremos que sacarlas por el monitor y en el caso de sonido por la tarjeta de sonido o por la bocina, pero la computadora los trata a todos como cadenas de 0's y 1's.

Por ejemplo, normalmente lo único que sabemos del sistema de graficación, es que consta de un monitor y una tarjeta de video, pero si empezamos a profundizar nos damos cuenta que es una mina de información sobre interactividad y que prácticamente todo lo que se maneja en un sistema interactivo, tiene una representación concreta en un sistema de graficación.

² Fernando Galindo Soria ESCOM.

³ Jesús Antonio Jiménez Aviña ESCOM.

Es por lo anterior que en este trabajo proponemos que un camino factible para introducir a los estudiantes de Informática y Computación en la interactividad es a través del estudio y manejo de los sistemas de graficación.

Uno de los problemas que se presentan en esta propuesta, es la gran fragmentación del conocimiento ya que mientras en una fuente explican cómo graficar en algún lenguaje, en otras presentan que las imágenes se almacenan en algún tipo especial de formato, en otras hablan de diferentes tipos de tarjetas gráficas, en fin otras presentan la Teoría del Color. Si se profundiza más, se encuentra que la información sobre formatos gráficos, arquitectura de las tarjetas y sobre su manejo esta muy dispersa, y se detecta que se requieren conocimientos que normalmente se dan desvinculados en los cursos de Arquitectura de Computadoras, Sistemas Operativos y Ensambladores.

Es por lo anterior que en este trabajo presentaremos ejemplos en los cuales se ataca el problema del manejo de imágenes y veremos que es similar al del sonido (es prácticamente igual ya que también se manejan interrupciones, puertos, formatos, memoria, etc. y fácilmente una persona que aprendió a manejar imágenes pasa a sonido) y al de otros tipos de mecanismos interactivos.

Entre otros temas se ve como cambiar de un modo gráfico a otro usando interrupciones y un ejercicio sencillo para introducir al manejo de puertos y a la programación de tarjetas, que consistente en cambiar los valores de la paleta de colores programando directamente los puertos que controlan al DAC de la tarjeta de video.

Otro de los problemas clásicos dentro de la Interactividad se presenta en el manejo masivo de información, en particular cuando se manejan imágenes o sonido se esta tratando con cantidades masivas de información, ya que por ejemplo, una sola imagen contiene alrededor de 256 mil caracteres, por lo que se mostraran ejemplos de manejo de sonido e imagen en forma masiva dentro de una computadora, incluyendo el manejo de formatos de video y sonido, con el fin de introducir al tema y mostrar que los diferentes problemas de interactividad siguen siendo similares.

Introducción

La Interactividad o el área de los Sistemas Interactivos estudia los mecanismos y la forma para lograr que dos o más dispositivos o sistemas se relacionen reciproca y simultáneamente entre si, de tal manera que las acciones e información generada por unos repercutan en los otros. Es un área en la que se manejan cientos de miles de millones de dólares y en la que se involucran la robótica, realidad virtual, sistemas evolutivos, telecontrol, simuladores, biotica, multimedia, sistemas de comunicación, control distribuido, música por computadora, etc.

Los sistemas interactivos se encuentran presentes en prácticamente todas las actividades cotidianas. Por ejemplo, la interrelación entre el sensor de temperatura y un calentador para regular la temperatura del agua, la forma como se relacionan una cámara de video con una televisión, una computadora con un sistema de audio, en los sistema para detectar la presencia

de personas en un cuarto, en un simulador (de vuelo, del metro, de una planta termoeléctrica), en sistemas de telecontrol, etc.

Esta es una de las áreas de mayor desarrollo mundial, sin embargo en nuestras escuelas prácticamente no existen cursos sobre este tema. Lo anterior es en parte por desconocimiento y en parte porque se cree que se requiere muchos recursos. Lo último es relativamente cierto, por ejemplo, si se quiere realizar un sistema de realidad virtual o de reconocimiento de voz normalmente no se cuenta con los recursos necesarios para hacerlo.

Sin embargo, al analizar múltiples sistemas interactivos nos damos cuenta que manejan componentes parecidos y que es relativamente fácil aplicar los conocimientos adquiridos al desarrollar unos sistemas en el desarrollo de otros. Prácticamente en todos los casos se construyen o manejan aparatos, interfaces y programas y se requiere usar interrupciones, puertos, archivos con formato, manejadores de entrada/salida, microcontroladores, circuitos programables, etc.

Si se logra crear conciencia entre los estudiantes de las similitudes entre estos sistemas, es relativamente fácil para ellos aplicar los conocimientos adquiridos para desarrollar unos en el desarrollo de los otros.

Y una cosa curiosa y muy interesante es que, no se requiere contar con una gran cantidad de recursos para aprender a construir este tipo de sistemas. Ya que en los sistemas de cómputo actuales existen sistemas interactivos completos que en su momento pueden servir para introducir a la gente en la interactividad sin necesidad de conseguir recursos externos. Tal es el caso de los sistemas de graficación y en general de manejo de imágenes de las computadoras.

Comúnmente lo único que sabemos del sistema de manejo de imágenes de las PC's, es que consta de un monitor y una tarjeta que comunica el monitor con la computadora, pero si empezamos a profundizar nos damos cuenta que estamos entrando en una mina de información sobre interactividad y que prácticamente todo lo que se maneja en un sistema interactivo, tiene una representación concreta en un sistema de graficación por computadora.

Es por lo anterior que en este trabajo proponemos que un camino factible para introducir a los estudiantes de Informática y Computación en la interactividad es a través del estudio y manejo de los sistemas de graficación.

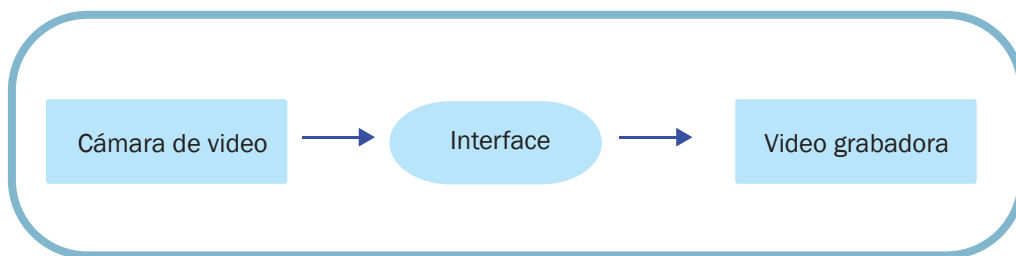
Uno de los problemas que se presentan en esta propuesta, es la gran fragmentación del conocimiento ya que mientras en una fuente explican cómo graficar en algún lenguaje, en otras presentan que las imágenes se almacenan en algún tipo especial de formato (pero sólo lo mencionan y no indican en qué consiste), en otras hablan de diferentes tipos de tarjetas gráficas (pero no explican como se programan), en fin otras presentan la Teoría del Color. Si se profundiza más, se encuentra que la información sobre formatos gráficos, arquitectura de las tarjetas y sobre su manejo esta muy dispersa, y se detecta que se requieren conocimientos que normalmente se dan desvinculados en los cursos de Arquitectura de Computadoras, Sistemas Operativos y Ensambladores.

Y prácticamente en todos los casos se requieren construir o conseguir los aparatos, las interfaces y desarrollar los programas para manejar los sistemas interactivos. Para una gran cantidad de casos se requiere conocer y manejar: interrupciones, puertos, archivos con formato, manejadores de entrada/salida, microcontroladores, circuitos programables, lectura de discos, programación en binario de los registros de las tarjetas, manejo de memoria, etc.

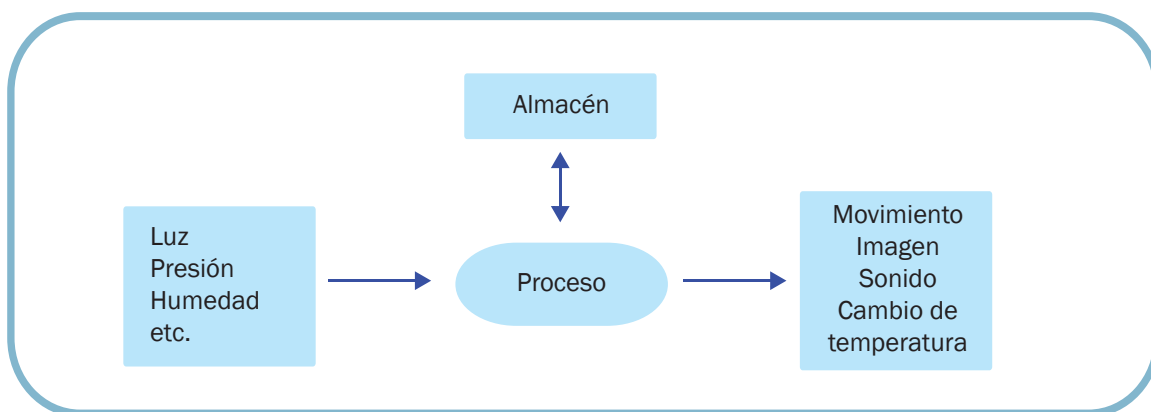
Es por lo anterior que en este trabajo presentaremos ejemplos sobre manejo de imágenes y veremos que el problema es similar al del sonido (es prácticamente igual ya que también se manejan interrupciones, puertos, formatos, memoria, etc. y normalmente una persona que aprendió a manejar imágenes pasa directamente a sonido) y al de otros tipos de mecanismos interactivos.

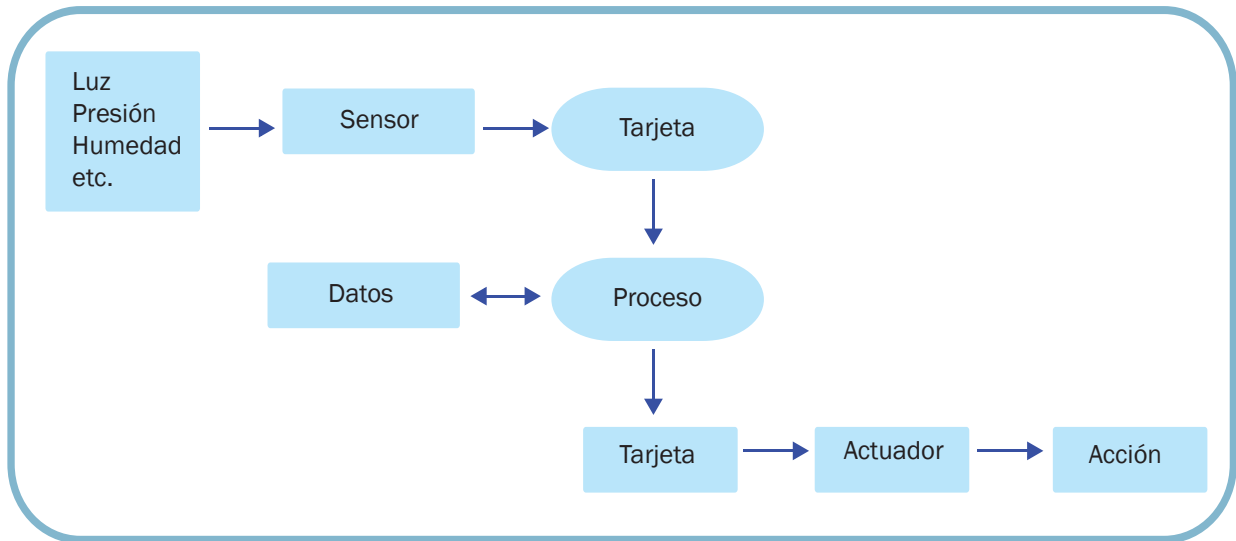
I Arquitectura General de un Sistema Interactivo

Como primer punto comentaremos que, en un sistema interactivo es necesario lograr que al menos dos objetos se interrelacionen, por ejemplo una cámara de video con una videograbadora, para lo cual se requiere normalmente un mecanismo de interfase entre los dos objetos.

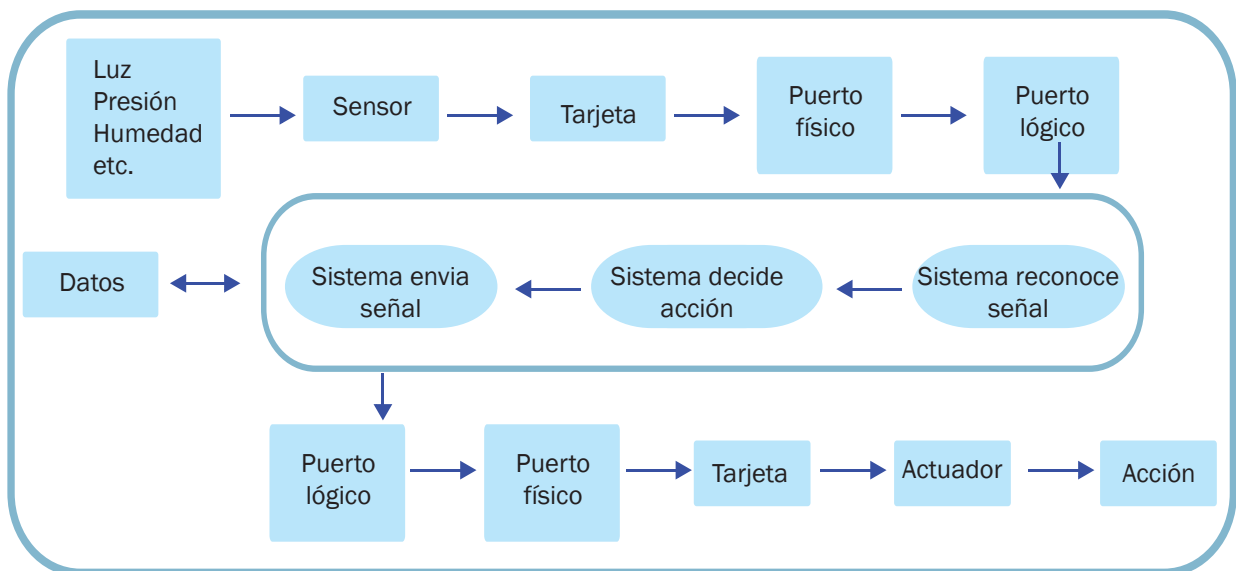


En particular en el caso de las computadoras, la arquitectura de un sistema interactivo es:





Y más específicamente, si detallamos más los componentes del sistema interactivo se llega a una arquitectura como la siguiente:



Como se puede ver, las principales actividades que se realizan en un sistema interactivo son comunes a cualquier tipo de sistema, pero sin embargo en el caso particular de estos sistemas se tienen ciertas características que conviene resaltar: una es la **capacidad de procesamiento en tiempo real**, ya que un sistema interactivo tiene que tener un tiempo de respuesta muy rápido, otra es que comúnmente las **interfaces con el sistema no son las tradicionales**, sino que usan prácticamente cualquier mecanismo de adquisición y salida de información.

Otra característica común a este tipo de sistemas es que normalmente **manejan grandes cantidades de información**, ya que, por ejemplo en el envío de una imagen o de una canción entre dos dispositivos se esta manejando un mayor volumen de datos que los manejados en muchos de los procesos administrativos tradicionales.

Una última característica consiste en que, **si se desarrolla un sistema interactivo específico, la experiencia particular se puede migrar fácilmente a muchos otros**, ya que en todo los casos se requiere desarrollar mecanismo de captura, procesamiento, almacenamiento y salida de los datos.

Por lo que a pesar de que es muy fácil rendirse y no entrar al área de la interactividad por falta de recursos, ya que, no siempre se cuenta con todo el equipo que se quisiera, esta posición no es valida, ya que todas las computadoras cuentan con una tarjeta de video, por lo que podemos introducirnos a la interactividad programando la tarjeta de video y posteriormente migrando esta experiencia a otros tipos de dispositivos.

Se eligió la tarjeta de video porque los problemas involucrados en su manejo son problemas clásicos que se presentan al manejar cualquier otro dispositivo interactivo como una tarjeta de sonido o de adquisición de datos, ya que todas estas tarjetas se pueden programar directamente y prácticamente en todos los casos en forma parecida.

Por lo que, a continuación se muestra como se enfrentan los problema de entrada/salida y almacenamiento de información en un sistema interactivo, usando como casos particulares la programación de la tarjeta de video para que realice algunas funciones específicas y el almacenamiento y despliegue de imágenes en una computadora.

La programación de cualquier dispositivo interactivo puede ser extremadamente compleja y en su caso existen libros completos de cientos de paginas para programar algunos de estos, por lo que no es el objetivo dar un curso sobre el tema, sino simplemente mostrar con ejemplos sencillos como se puede programar un dispositivo interactivo y crear conciencia de que este proceso se puede realizar con muchos otros dispositivos.

II Controladores de vídeo

Se podría decir que es distinto manejar sonido, presión, temperatura o imágenes, pero lo único distinto que tienen es la forma como lo percibimos, ya que la computadora los trata de la misma forma, como cadenas de 0's y 1's. Si manejamos imágenes tendremos que sacar la información por medio del monitor y en el caso del sonido por medio de la tarjeta de sonido o por la bocina interna de la PC.

Para manejar cualquiera de estos tipos de información, además de que es aconsejable conocer como se transmite el sonido o como se generan los colores (ya que, entender esto facilita el manejo de la información), debemos saber también como programar los dispositivos de entrada y salida.

Por ejemplo para manejar gráficas por computadora los compiladores de **C y Pascal** ya cuenta con controladores de video como los ***.bgi**, los cuales nos permiten manejar el modo gráfico para varios tipos de tarjetas, desde **CGA y EGA** hasta **VGA**.

Pero a pesar de que en una tarjeta normal VGA la memoria consta de 256K estos controladores solo permiten una resolución máxima de 640X480 pixeles y 16 colores. Si queremos realizar un trabajo de mejor calidad tenemos que ocupar controladores como el VGA256.BGI o SVGA256.BGI, que aumenta el manejo de colores de 16 a 256. y dependiendo del tipo de tarjeta aumenta la resolución.

Otra opción consiste en construir nuestro propio controlador, para lo cual debemos programar directamente la tarjeta de video. Ahora bien, la información para programarla se encuentra muy dispersa y es difícil de conseguir. Por lo que a continuación se da una introducción al tema.

Si contamos con una tarjeta VGA tenemos ya por default 256 colores y una resolución de 320X200 (modo 13h), lo cual da un total de 64K de memoria convencional, Por lo que, un primer ejercicio de programación de la tarjeta de video consiste en cambiar nuestra tarjeta al modo 13h (dependiendo del lenguaje los **números hexadecimales** se indican con un **\$ en Pascal, 0x en C o una h en ensamblador**)

1 Cambio del modo de vídeo

Para cambiar la computadora de un modo de video a otro usando interrupciones se puede usar la siguiente rutina que llama a la **interrupción 10h** que es la interrupción del video del BIOS:

```
void modogra(int modo) //inicia el modo de video, para tarjetas hasta VGA
{
    union REGS reg;
    reg.h.ah=0;
    reg.h.al=modo;
    int86(0x10,&reg,&reg);
}
```

y si dentro del programa se llama a la instrucción:

modogra(0x13)

la maquina se pone en modo 13h

Antes de terminar el programa es conveniente **regresar a la computadora a un modo normal**, para lo cual se puede usar por ejemplo las instrucciones:

modogra(0x12); //regresa a modo de video 12h 640*480*16 colores

ó

textmode(C80); //pasa a modo texto

2 Memoria de Vídeo

Una de las características mas interesantes de la programación en modo 13h es que asociado a cada pixel de la pantalla se tiene una localidad de memoria, o sea que si se coloca un valor en esa localidad de memoria automáticamente aparece un color en una posición de la pantalla con lo que se pueden sacar imágenes en la pantalla programando directamente la memoria.

El área de la computadora donde se almacenan las imágenes que salen por el video se conoce como **memoria de video y empieza en la localidad A000:0000h**.

Si estamos en **modo 13h** y mandamos escribir un valor en esa localidad podremos ver un punto en la parte superior del monitor. La ventaja de mandar a escribir directamente a memoria, es que agiliza el manejo de información, **(en otros modos no es tan directa la programación de la memoria de video por lo que se recomienda dejar ese problema para el momento en que se tenga mayor manejo del tema)**

Para acceder a la memoria de video en C se debe declarar un apuntador mediante una instrucción como la siguiente:

```
char far *apun= (char far *) 0xA0000000;
```

Para mandar escribir un punto a una localidad se usa una instrucción como:

```
*(apun+j*320+i)=c;
```

donde **apun es la base** o inicio de la memoria de video, **i** nos indica **la columna**, **j el renglón donde se quiere poner el punto** y **c** es el índice del **color** que se quiere poner.

Con lo anterior ya se tiene una base mínima para manejar directamente el video, sin embargo para la gran mayoría de actividades relacionadas con los dispositivos interactivos, se requiere de la programación directa de puertos, por lo que a continuación se muestra un pequeño ejemplo de esta actividad.

3 Paleta de Colores

Un ejercicio sencillo para introducirnos al manejo de puertos y la programación de tarjetas, consiste en cambiar directamente los valores de la paleta de colores, para lograr que tome valores entre 256,000 posibles colores, programando directamente los puertos que controlan al DAC (Digital Analogic Converter) de la tarjeta de video.

La **paleta de colores** no es más que una tabla de **256 registros con 3 componentes para cada registro**. Los componentes son **R, G, B (rojo, verde, azul)**. La **tarjeta de video cuenta con un DAC** el cual permite el acceso a la paleta de colores mediante los **puertos 3C7h, 3C8h, 3C9h**.

Un **puerto** es un dispositivo Físico/Lógico que permite la conexión entre dos aparatos, programando directamente los puertos obtenemos la respuesta en un menor tiempo. Al aumentar la velocidad de respuesta se pueden realizar por ejemplo animaciones en tiempo real, y una interactividad mejor.

El manejo de puertos es muy simple pero puede ser peligroso y en su momento se dificulta su uso porque, para cada aparato que se comunica mediante un puerto se requiere conocer el número de puerto y los valores de las señales que permiten que funcione, y gran parte de esta información es poco accesible

Cuando se programan puertos se tiene que tomar en cuenta que muchas veces se está programando directamente un pequeño procesador y sino se sabe manejar se corre el peligro de dañar un dispositivo, por lo que, es conveniente no realizar experimentos ni jugar con los puertos ya que son delicados, por lo que **en los programas que siguen se recomienda tener cuidado y no ponerse a “hacer experimentos”** y si se quiere continuar profundizando sobre el tema, verificar siempre las instrucciones que se quieren usar.

Las funciones de los puertos del DAC son las siguientes:

3C7h En este puerto se debe almacenar el **índice del color** del que se quiere **leer** sus componentes **R, G, B**

3C8h En este puerto se debe almacenar el **índice del color** que se va a **modificar** en sus **componentes R, G, B**

3C9h Componentes del color (R, G, B)

Cada vez que se hace la llamada al puerto 3C9h, automáticamente apunta al siguiente registro de color. Con un par de funciones podemos modificar los colores o crear nuevas tonalidades que antes no podíamos realizar.

//Rutina para leer un color específico

```
void R_C(int C, int R, int G, int B)
{
    outportb(0x3c7,C); //Escribimos el índice del color
    R= inportb(0x3c9); //Leemos componente Roja
    G= inportb(0x3c9); //Leemos componente Verde
    B= inportb(0x3c9); //Leemos componente Azul
}
```

//Rutina para modificar un color con componente R, G, B.

```
void W_C(int C, int R, int G, int B)
{
    outportb(0x3c8,C); //Escribimos el índice del color
    outportb(0x3c9,R); //Escribimos componente Roja
    outportb(0x3c9,G); //Escribimos componente Verde
    outportb(0x3c9,B); //Escribimos componente Azul
}
```

Con estas funciones podemos tener un total de **256 K** combinaciones de colores en una tarjeta de video VGA.

III Manejo de Información Masiva

Otro de los problemas clásicos dentro de la Interactividad se presenta en el manejo masivo de información, en particular cuando se manejan imágenes o sonido se esta tratando con cantidades masivas de información, ya que por ejemplo, una sola imagen contiene alrededor de 256 mil caracteres y un segundo de animación corresponde a 30 imágenes o sea 7,500,000 caracteres, por lo que se muestran ejemplos de manejo de sonido e imagen en forma masiva dentro de una computadora, incluyendo el manejo de formatos de video y sonido, con el fin de introducir al tema y muestra que los diferentes problemas de interactividad siguen siendo similares.

Estos dispositivos de captura cuentan con formatos propios o estandares para manejar información, entre los más usuales para imágenes se tienen GIF, TIFF, JPG, PCX, BMP etc., y para sonido VOC, MOD, MIDI, WAV, etc. Algunos de estos formatos cuentan con cierto tipo de compactación, por lo cual es difícil manejarlos si no contamos con la información del formato. Pero otros son más simples como es el caso del formato BMP para imagen y WAV para sonido. Por lo que a continuación se mostrara como se maneja información formateada.

1 Formato BMP

En la actualidad existen en el mercado diversos tipos de **formatos para almacenar imágenes**, su diversidad es muy grande. Algunos de los formatos más populares son GIF, TIFF, JPG, BMP, etc. Todos estos formatos son comerciales y no es fácil encontrar información interna sobre ellos.

Uno de los formatos más sencillos es el BMP (Bit MaP, Mapa de Bits). Un archivo con formato BMP consta de dos partes: cabecera y cuerpo. La cabecera esta compuesta de: la identificación del formato del archivo, dirección de inicio del cuerpo, tamaño horizontal de la imagen, tamaño vertical de la imagen, el número de bits por pixel y la paleta de colores.

Con la información de la cabecera obtenemos el tamaño de imagen, el tipo de resolución del archivo (si es de 2, 16 o 256 colores), y la paleta de colores propia de esa imagen.

En el cuerpo de la imagen se encuentra el valor del color de cada pixel, que se acomoda en la posición correspondiente de acuerdo al tamaño de horizontal y vertical, podemos pensar en algún momento que la imagen esta guardada de arriba hacia abajo lo cual es falso. El primer pixel del cuerpo corresponde a la parte inferior izquierda de la imagen, los demás pixeles se irán acomodando de izquierda a derecha y de abajo hacia arriba.

La ubicación de la información dentro del archivo es la siguiente:

Byte	Significado	Nombre de la variable
0	Identificación del archivo.	
10	Lugar donde empieza el cuerpo	CUERPO
18	Tamaño horizontal	TAM_X
22	Tamaño vertical.	TAM_Y
28	Bits por pixel.	BIT_PIXEL
30	Principio de la paleta de colores	PALETA
XX	El principio del cuerpo se calcula por medio de los bits por pixel y la paleta de colores.	

La paleta de colores es una tabla donde se tienen los colores de la imagen, cada color esta formado de 4 componentes R, G, B, A, donde R, G, B, indican la cantidad de rojo, verde y azul de cada color, A es una máscara que nos indica como se combina ese color con los demás, hay que tener cuidado ya que la paleta de colores esta escrita en el orden de B, G, R, A

A continuación se muestra un programa que despliega archivos con formato BMP en la pantalla de la PC.

```

/* Programa que despliega una imagen con formato BMP */
/* Realizado por: J. Antonio Jiménez Aviña */
/* 29-04-96 */
/*modificado por Fernando Galindo Soria
   rutina propia para cambiar paleta
   rutina propia para traer el color del siguiente pixel
   8/vi/96 Cd. de México.*/
#include <conio.h> //Cabeceras
#include <stdio.h>
#include <graphics.h>
FILE *BMP;
//Declaración de variables
char identificación[118];
long int CUERPO, TAM_X, TAM_Y, BIT_PIXEL,
      PIXEL_byte, PALETA, PALETA_TAM;
int mask;
int ShowBMP(char* fname,int,int);
void leecabecera(void);
void leepaleta(void);
void esccolordac16(char color,char r,char g,char b);
void leecuerpo(int,int);
void main(void)
{
    int gdriver=DETECT,gmode;
    initgraph(&gdriver,&gmode,"");
    ShowBMP("ARCHIVO.BMP",0,0); //poner el nombre del archivo donde está la imagen a desplegar
    getch();
    closegraph();
}
int ShowBMP(char* fname,int x0,int y0)

```



```

//Función que abre el archivo y despliega la imagen en el monitor
BMP= fopen(fname,"rb");//abre el archivo de imagen
if (BMP==NULL) return -1;

leecabecera();
leepaleta();
leecuerpo(x0,y0);
fclose(BMP);
return 0;
}

void leecabecera(void)
{
//lee la cabecera
fseek(BMP,0,SEEK_SET);
fread(&identificacion,sizeof(identificación),1,BMP);//lee que sea archivo tipo BMP
fseek(BMP,10,SEEK_SET);
fread(&CUERPO,sizeof(CUERPO),1,BMP); //Lugar donde empiece el cuerpo.
fseek(BMP,18,SEEK_SET);
fread(&TAM_X,sizeof(TAM_X),1,BMP); //Tamaño horizontal.
fseek(BMP,22,SEEK_SET);
fread(&TAM_Y,sizeof(TAM_Y),1,BMP); //Tamaño vertical.
fseek(BMP,28,SEEK_SET);
fread(&BIT_PIXEL,sizeof(BIT_PIXEL),1,BMP);//Bits por pixel.

PIXEL_byte=8/BIT_PIXEL;    //numero de pixels por byte
mask=(1<<BIT_PIXEL)-1;
}

void leepaleta(void)
{
//lee del archivo la paleta de colores y la carga en la tarjeta de video
typedef struct //Estructura para el color
{
    unsigned char b, g, r, a;
} Color;
Color col;
int k;
PALETA_TAM=1<<BIT_PIXEL;
PALETA=CUERPO-4*PALETA_TAM;    //Principio de la paleta de colores.

fseek(BMP,PALETA,SEEK_SET);
for (k=0;k<PALETA_TAM;k++) //Cargando la paleta de colores
{
    fread(&col,sizeof(col),1,BMP);
    escrcolordac16(k,(col.r)>>2,(col.g)>>2,(col.b)>>2);
}
}

void escrcolordac16(char color,char r,char g,char b)
{/*cambia de color paleta llamando directamente a los puertos

```

del convertidor analógico digital de la tarjeta de video

utiliza initgraph y maneja paleta rgb de 16 colores b(lue) g(reen) r(ed)*/

```

char indcol[]={0,1,2,3,4,5,20,7,56,57,58,59,60,61,62,63};
outportb(0x3C8,indcol[color]);/*inicializa el color*/
outportb(0x3C9,r);
outportb(0x3C9,g);
outportb(0x3C9,b);
}

void leecuerpo(int x0,int y0)
{ //Función que regresa el color de los pixels
  // de acuerdo a sus componentes RGB y lo pone en la pantalla

  int xi=0, n=0, k,Xpal,Ypal;
  long int col[4];
  unsigned char b,b1[4];

  fseek(BMP,CUERPO,SEEK_SET);

  for (Ypal=TAM_Y-1; Ypal>=0; Ypal--) //Despliega la imagen en el monitor
  {xi=0; n=0;
  for (Xpal=0; Xpal<TAM_X; Xpal++)
  {
    if (xi%PIXEL_byte==0)    //desempaca los pixels empacados en un byte
    {
      fread(&b,sizeof(b),1,BMP); //trae un byte del archivo
      k=PIXEL_byte-1;
      col[k]= b & mask;  //desempaca cada pixel y lo almacena en un arreglo
      while (k--) col[k]=(b>>=BIT_PIXEL) & mask; //regresa el índice del color del siguiente pixel
      n++;          //cuenta cuantos bytes ha desempacado
    }

    putpixel(x0+Xpal,y0+Ypal,col[xi++%PIXEL_byte]); //despliega el siguiente pixel
  }

  fread(&b1,1,(4-n%4)%4,BMP); //ignora bytes para quedar en múltiplos de 4
  }
}

```

2 Manejo de Sonido

Como se puede ver en un problema tan simple como el desplegar una imagen en pantalla se requieren tener una serie de conocimientos que normalmente pasan transparentes a los usuarios normales de la computación, pero que sin embargo, son cotidianos en el manejo de sistemas interactivos, por otro lado el manejo masivo de información requiere un tratamiento similar en el caso de archivos de voz, ya que también se requiere obtener la información del encabezado y usarla para desempacar los valores de los sonidos, por lo que, el desplegar imágenes es un

buen ejercicio para manejo de información masiva, que se puede aplicar directamente al manejo de otros tipos de datos como es el caso del sonido, por lo que a continuación se comentará sin entrar a detalle y con el fin de que se vea la semejanza con el manejo de imágenes unos ejemplos de manejo masivo de sonido.

Dentro de los **archivos de sonido**, unos de los más importantes son los que tienen formatos tipo **WAV y VOC**. En estos formatos podemos grabar tanto música como voz.

El formato de Microsoft Waveform (WAV), es uno de los más populares actualmente gracias a Windows y sus múltiples aplicaciones en Multimedia, ya que su calidad de sonido es lo bastante buena al ser grabado en 8 o 16 BYTES. Lo malo es que ocupa demasiado espacio para unos cuantos segundos de sonido.

El formato WAV esta organizado de acuerdo a la estructura **RIFF (formato de archivo de intercambio de recurso)**. Esta estructura fue diseñada hace varios años para los **archivos de recursos multimedia**, Un archivo de Formato WAV esta compuesto de una cabecera y un cuerpo.

Los elementos de la **Cabecera** son:

- 1) *Identificación del formato RIFF* (4 BYTES). Regresa la cadena “**RIFF**”.
- 2) *Longitud de datos del formato RIFF* (4 BYTES).
- 3) *Identificación del formato como formato de sonido (WAV)* (4 BYTES). Regresa la cadena “**WAVE**”.
- 4) *Datos del formato RIFF* (puede o no existir) La longitud se da en longitud de datos del formato RIFF.
- 5) *Identifica al bloque como un fragmento de formatos* (4 BYTES). Regresa la cadena “fmt”
- 6) *Longitud de los datos del formato que viene a continuación* (4 BYTES).
- 7) *Categoría al que pertenece el formato WAV* (2 BYTES). No esta documentada pero si regresa un 01 indica que es un formato Pulse Code Modulation (PCM).
- 8) *Canales de salida*. (2 BYTES). Indica si es un archivo Mono o Estéreo (1 para Mono y 2 para Estéreo).
- 9) *Frecuencia de muestreo*, muestras por segundo a la que debe de reproducir cada canal (2 BYTES). Este valor esta dado en Hertz.
- 10) *Número medio de BYTES que deben de transmitirse por segundo* (2 BYTES). También se puede calcular por la siguiente formula: **Número medio de BYTES por segundo = Canales * Número de muestras por segundo * (Número de Bits por muestra / 8)**, donde el Numero de Bits por Muestra es igual al Numero de Muestras por segundo entre ocho.
- 11) *Alineamiento del bloque* (2 BYTES). Ya que la computadora puede procesar solo un bloque a la vez. Este dato se utiliza para alinear los buffers de cada canal.
- 12) *Especifico del formato* (2 BYTES). No se encuentra documentado.
- 13) *Data Identifica al siguiente bloque como fragmento de datos* (4 BYTES). Regresa la cadena “fmt”.
- 14) *Longitud del bloque* (4 BYTES).Indica la longitud del cuerpo del archivo de sonido.

Los elementos del **Cuerpo** son:

- 1) *Bloque de datos*. Es donde se encuentra grabada la información a reproducir.

Como se observa cada uno de los datos tiene una longitud específica la cual esta dada por el fabricante o bien dentro del formato.

Ahora bien, existen muchas otras formas para almacenar y manejar datos masivos, por lo que, finalmente comentaremos otra de las formas de almacenamiento masivo de sonidos conocida como formato VOC. Este tipo de formato es solo para **archivos que hayan sido grabados por medio de un código de modulación de pulsos (PCM) de 8 bits**.

El Formato VOC consta en su cabecera de un bloque de encabezamiento y un bloque de datos, dentro del bloque de datos puede haber a su vez diferentes tipos de bloques, los cuales proporcionan una manera elegante de programar pausas, repeticiones etc. Pero comúnmente solo encontramos un tipo de bloque, el correspondiente a los datos de voz.

Como se puede observar el problema principal que se tiene para manejo de mucha de la información interactiva se reduce al conocimiento o desconocimiento de los formatos con que se almacena o transmite.

Conclusiones

La interactividad es una de los campos mas importantes a nivel mundial e incluye áreas aparentemente tan disimilares como la graficación, robotica, realidad virtual, proceso distribuido y muchas otras que si se analizan independientemente del problema especifico tienen practicante la misma arquitectura y los mismos componente. Por lo que, cada vez es mas importante que nuestros alumnos se introduzcan en esta área.

Sin embargo, prácticamente no se desarrollan actividades académicas sobre este tema, porque se cree que se requiere contar con grandes infraestructuras para desarrollarlas. Es por lo anterior, que el objetivo de este trabajo es mostrar que nos podemos introducir en la interactividad sin necesidad de contar con grandes recursos, ya que prácticamente todos los problemas del desarrollo de un sistema interactivo están presentes en el manejo de una tarjeta de video.

Esta idea ya se ha comenzado a aplicar involucrando a estudiantes desde secundaria hasta posgrado, siendo muy gratificante observar como estos estudiantes cada vez han profundizando mas sobre este tema y lo fácil que ha sido aplicar la experiencia obtenida en la programación de las tarjetas de video a la programación de otros tipos de dispositivos interactivos.

Con lo cual, también se ha logrado romper un bloqueo tradicional ya que mucha gente no se ha dado cuenta que si son capaces de manejar un dispositivo pueden manejar con relativa facilidad muchos otros y que es necesario romper con los feudos de conocimiento, ya que **los problemas que se atacan son similares y corresponden a una gran área conocida como interactividad**.

Bibliografía

- [1] Peter M. Ridge, “Guía oficial de Sound Blaster”, Osborne McGraw-Hill España 1994.
- [2] Boris Bertelson & Mathias Rasch, “PC al limite”, Data Beck Colombia 1995.
- [3] “PC Interno”, Data Beck.

IV.4 Reforned: Reconocimiento de Imágenes por medio de Redes Neuronales, Sistemas Evolutivos y Distribuidos

Alejandrina Salazar Torres⁴

Resumen

En este trabajo se presenta la integración de redes neuronales y sistemas evolutivos, ambos utilizados como herramientas para la elaboración de un sistema de reconocimiento de imágenes.

El sistema consta, en su arquitectura, de una red neuronal basada únicamente en representaciones matriciales, en la cual, las imágenes al ser capturadas y digitalizadas se almacenan dentro de ésta, y de un mecanismo evolutivo, el cual permite retroalimentación, reforzamiento y actualización de imágenes.

Así de este modo REFORNED está constituido de una red neural evolutiva centrándose principalmente, al desarrollo de ésta red, así como de comprobar su alcance de aprendizaje y reconocimiento.

I Planteamiento del Problema

Debido a que los sistemas que están orientados al reconocimiento de imágenes o patrones presentan normalmente, un modelo restringido de la realidad, ocasionando con esto, problemas que día con día se van complicando al igual que sus soluciones; y aunque se han desarrollado diferentes sistemas con diferentes métodos y algoritmos dentro de la Inteligencia Artificial, no permiten la solución de dichos problemas.

Debido a lo anterior, se tomó la idea de utilizar una red neural evolutiva, para reconocimiento de imágenes, centrándose principalmente, al desarrollo de ésta red, y comprobar el nivel de aprendizaje y reconocimiento en los resultados obtenidos.

⁴ Alejandrina Salazar Torres realizó este trabajo cuando cursaba el segundo semestre de la Licenciatura en Ciencias de la Informática en la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA).

La Red Neural Evolutiva posee las características de aprendizaje, reconocimiento, retroalimentación, reforzamiento y actualización.

Cabe decir que el modelo de Red Neural Evolutiva ha sido ya tratada por Fernando Galindo Soria.

II Reformed

El funcionamiento general del sistema se aprecia en el diagrama de la figura 1.

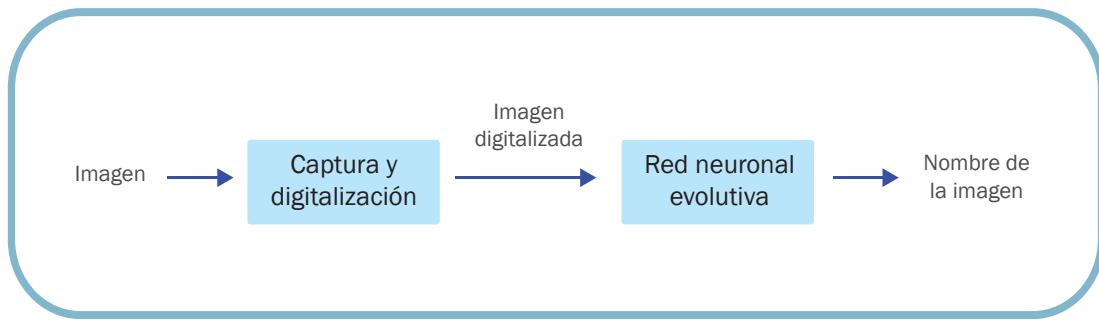


Figura 1. Componentes de REFORNED.

III Red Neural Evolutiva

El manejo de las imágenes al ser digitalizadas, se hace mediante manejo de matrices y vectores, en donde las imágenes digitalizadas se almacenan dentro de la matriz de manera sucesiva; posteriormente, al introducir la imagen a reconocer, se hace en forma de vector, una instancia de la matriz evolutiva se muestra en la figura 2.

Imagen	Umbral	Nombre
I_1	h_1	N_1
I_2	h_2	N_2
...
I_n	h_n	N_n

Figura 2. Una instancia de la Matriz Evolutiva.

El mecanismo Evolutivo permite normalizar, regular el tamaño, reforzar y permitir retroalimentación. Este mecanismo funciona como sigue: ya introducida la imagen, esta recibe un “tratamiento”, que consiste en delimitar el espacio, de tal forma que siempre se use la misma cantidad de puntos; además de realizar todas las combinaciones posibles. Ya hecho este “tratamiento” sobre la imagen, se sobreencima en el mismo vector (trabajos de Cuitlahuac Cantú Rohlik y Angel Cesar Morales Rubio).

Esto permite que exista, al introducirse la imagen a reconocer, una “coladera” en donde la imagen coincide en una línea en donde aumentan considerablemente las variables, y arrojan un resultado mayor en comparación con las demás.

Este proceso logra además, que al sumarse las imágenes digitalizadas con las ya almacenadas en la red, vaya concluyendo el sistema las características principales, secundarias, terciarias, etc. de la imagen.

Conclusiones

Como se sabe, las redes neuronales permiten reconocer imágenes, letras, etc.; pero también se pueden confundir. Hay que tener en cuenta que los humanos tenemos un 4% de error al reconocer objetos o personas; muchas veces hemos confundido a personas a pesar que existen rasgos muy diferentes. Por lo cual no es posible aún lograr un 96% de reconocimiento por medio de la computadora, y mucho menos reconocer al 100% algo.

Las redes Neuronales Evolutivas, no están exentas de este error, ya que puede suceder que al existir una retroalimentación constante, y sí no se normaliza, puede confundir un gato con un perro o viceversa; para aminorar este tipo de error se utiliza una normalización, que consiste en dividir el resultado final con las sumatorias de las entradas activadas.

Este proyecto, al engrosar tres herramientas: Sistemas Evolutivos, Distribuidos y Redes Neurales, es demasiado extenso, ya que falta aún la integración de los Sistemas Distribuidos, la cual se llevará a cabo posteriormente, en la etapa final del proyecto.

IV.5 Aplicación de los Sistemas Evolutivos en el Análisis de Espectros de Rayos Gamma

Luis E. Torres Hernández⁵

Luis C. Longoria G.⁶

Antonio Rojas Salinas⁷

El análisis por activación es una técnica mediante la cual se puede determinar la composición isotópica de un compuesto, esto se realiza mediante la irradiación con neutrones en un reactor nuclear, los elementos se activan y emiten radiación gamma con un espectro característico de cada elemento (fototípico) [1][2]. La recolección de los espectros gamma se realiza en un sistema multicanal y posteriormente son procesados en un software especializado para determinar la posición de los fotopicos que es lo que determina el isótopo y el área bajo el fotopico sirve para determinar la cantidad del isótopo. Normalmente el análisis de los espectros se realiza independientemente conteniendo poca información, y sin ninguna aportación en la experiencia del analista, por consiguiente, hay muchas ocasiones que los resultados no son los correctos.

Para la determinación de los fotopicos de manera automática se esta desarrollando un software en lenguaje C, utilizando el paradigma de los Sistemas Evolutivos, el cual se define como un sistema automatizado capaz de construir y mantener actualizada una imagen de la realidad que lo rodea, esto mediante la adquisición de conocimiento, para utilizarla en la solución de problemas [3].

Una de las herramientas utilizadas en los sistemas evolutivos es la matriz evolutiva, la ventaja de esta matriz se encuentra en que se le puede “enseñar” al sistema a distinguir entre diferentes entradas, si se quiere que a ciertas señales de entrada responda una señal específica de salida, se le dan varias señales para obtener un patrón de referencia, repitiéndose el proceso hasta que aprenda a reconocer la señal esperada. Este método se puede utilizar para enseñar al sistema responder a diferentes señales de alguna forma específica y aún más logrando que aprenda a distinguir entre diferentes tipos de señales [4].

⁵ Luis E. Torres Hernández,

⁶ Luis E. Torres Hernández,

⁷ Antonio Rojas Salinas Instituto Tecnológico de Toluca. Instituto Nacional de Investigaciones Nucleares. Departamento de Experimentación Nuclear.

Las señales de entrada al sistema de espectroscopia gamma que producen los fotopicos corresponden a efectos físicos en el detector, así las señales presentan variaciones dentro de cierto rango, las cuales pueden formar un patrón ó huella característica para cada fotopico. Aplicando la matriz evolutiva al análisis de espectros gamma se han generado algunas “huellas de comportamiento” a partir de varios archivos de datos con formato ASCII, el procedimiento que se sigue es el de representar los archivos de los espectros en forma de vector y varios de estos nos representan una matriz, si los vectores son iguales se suman entre si para determinar un umbral de tolerancia entre los datos, si los vectores son diferentes esto implica que existen uno o varios fotopicos en la sustancia y este vector se agrega a la matriz como un nuevo vector, esto se realiza solo si el experto lo determina como tal o de lo contrario se ignora.

Resultados

Siguiendo los principios de los Sistemas Evolutivos se han obtenido huellas o patrones de espectros gamma para su identificación elemental. En la figura 1 se muestra la huella obtenida con ocho espectros del radioisótopo ^{60}Co .

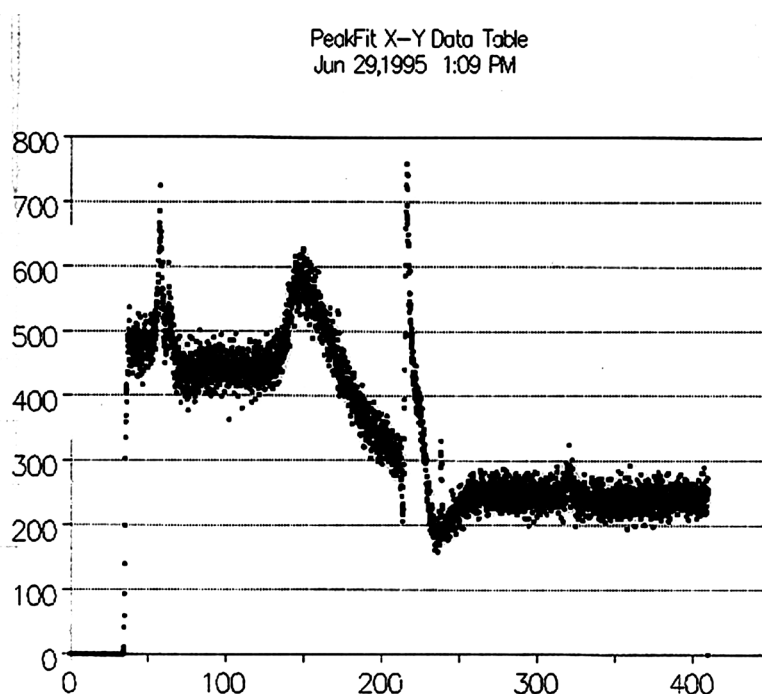


Figura 1. Huella o patrón espectral del ^{60}Co

Bibliografía

- [1] Knoli F, Radiation Measurements and Applications (1979).
- [2] Debertin K. and Helmer R.G, Gamma and X-Ray with Semiconductor Detectors (1982).
- [3] Galindo F, Sistemas Evolutivos Nuevo Paradigma de la Informática. Memorias de la XVII Conferencia Latinoamericana de la Informática, Caracas Venezuela, Julio, 1991.
- [4] Galindo F, Una Representación Matricial para Sistemas Evolutivos.

IV.6 Sistema Evolutivo para el Diagnóstico de Fallas en Máquinas Rotatorias

Eduardo De la Cruz Sánchez ⁸

Luis C. Longoria Gándara ⁹

Rodolfo A. Carrillo Mendoza ¹⁰

En los últimos años el paradigma de los Sistemas Evolutivos ha ganado un lugar importante en el campo computacional, desde su conceptualización como sistemas que pretenden construir una imagen de la realidad, se han desarrollado diferentes técnicas que han sido aplicadas para desarrollar potentes sistemas [1]. En forma más concreta se considera a un sistema evolutivo, como un sistema que debe ser capaz de interactuar con la realidad en la que está inmerso, para crear una imagen de ésta (realidad) y con ella resolver los problemas que se le plantean.

En este trabajo se presenta una aplicación práctica de la técnica “Matriz Evolutiva” a un sistema básico para diagnóstico de fallas en máquinas rotatorias, el cual permite identificar las fallas más comunes en una máquina, como por ejemplo el desbalanceo en un motor eléctrico o un ventilador. Este sistema básico, crea y reconoce diferentes patrones (huellas) generados por los espectros de las señales vibratorias obtenidos del muestreo en una máquina y de acuerdo a las características de cada uno de los espectros que se vayan incorporando, proporciona un diagnóstico. Si alguno de los espectros no es identificado claramente, el sistema “pregunta” al usuario, cual es su interpretación y lo registra, de esta manera podemos decir que el sistema “aprende”.

Palabras Clave: Sistemas Evolutivos, Diagnóstico, Matriz Evolutiva, Patrones de Referencia, Huella.

El sistema se desarrolló en lenguaje C, obtiene los datos de un equipo formado por acelerómetro, amplificador sensitivo a la carga, fuente de alimentación y analizador dinámico de señales; los procesa matemáticamente y los muestra al usuario como espectros en el dominio de la fre-

⁸ De la Cruz Sánchez Eduardo,

⁹ Longoria Gándara Luis C.,

¹⁰ Carrillo Mendoza Rodolfo A. Instituto Nacional de investigaciones Nucleares Depto. de Experimentación Nuclear Carretera México Toluca, Km. 36.5 Salazar Edo. de México, Teléfono : 5182360 ext 113,152 Fax: 5219408 Instituto Tecnológico de Toluca, Av. Instituto Tecnológico s/n. Metepec Edo. de México, Teléfono : 711046 ext 74.

cuencia mediante una interfaz gráfica, al mismo tiempo que presenta el espectro del patrón de referencia identificado en la matriz.

La forma de ir creando los diferentes patrones de referencia y el mismo diagnóstico, se logra precisamente aplicando la técnica de la matriz evolutiva, la cual tiene origen en un modelo matricial de una red neuronal basada en la representación de una neurona hecha por Mc. Culloc y Pitts[2]. La matriz evolutiva representa una red neuronal y el vector a las señales de entrada a la red, cada renglón de la matriz representa una neurona y cada columna una entrada de la neurona [2]. Para hacer la identificación de los espectros muestreados que entran en la matriz se determina un umbral que caracteriza particularmente a cada uno de ellos. Si el espectro es identificado, se proporciona el diagnóstico y se agrega al patrón correspondiente en la matriz, con lo cual se refuerzan estos patrones; el usuario en base a su experiencia puede determinar el número máximo de actualizaciones necesarias para definir el patrón. Si el espectro no es identificado se agrega a la misma matriz como nuevo.

Con este sistema de diagnóstico se han hecho pruebas en algunos equipos (motores eléctricos y ventiladores). Un ejemplo de los patrones que se han obtenido se muestran en las figuras 1 que muestra un espectro de un ventilador en condiciones normales, sin desbalanceo; observar que el nivel de vibración (pico) en la frecuencia de Trabajo (frecuencia fundamental) de 25 Hz es de -90.23 Db. En otra muestra se puede detectar variaciones al espectro de la figura 1, en el mismo ventilador con un desbalanceo inducido con un peso de 0.915 gr, en donde se genera un aumento del pico a -75.49 Db en la frecuencia de trabajo (25 Hz).

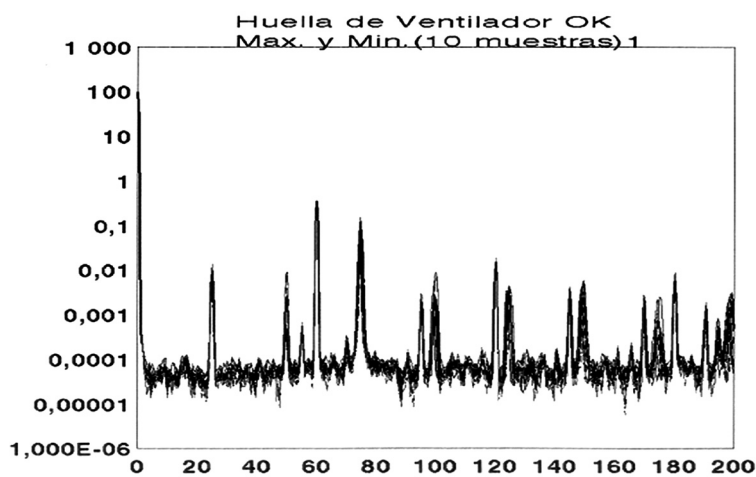


Figura 1. Espectro de ventilador en condiciones normales, sin desbalanceo

Bibliografía

- [1]Galindo Soria Fernando, “Aplicación de los Sistemas Evolutivos al Reconocimiento de Formas”, UPIICSA-IPN.
- [2]Galindo Soria Fernando, “Una Representacion Matricial para Sistemas Evolutivos”, UPIICSA-IPN.

IV.7 Sistema Evolutivo para el Reconocimiento de Símbolos Taquigráficos

Irene Arzola Carvajal ¹¹
José Rafael Cruz Reyes ¹²

Este trabajo muestra una forma de utilizar a una computadora para el reconocimiento de signos taquigráficos, de tal manera, que a medida que la computadora aprenda, vaya reconociendo textos completos de taquigrafía y aprenda a inferir las equivalencias de símbolos aún no aprendidos. Para resolver este problema, se utiliza el paradigma Evolutivo, que establece que un sistema debe ser capaz de interactuar con la realidad en la que está inmerso para crear una imagen de ésta y con ella resolver los problemas que se le plantean. El sistema construido está compuesto de las siguientes etapas:

- 1) *Etapas para conocer la realidad.*
- 2) *Etapas para almacenar el conocimiento.*
- 3) *Etapas para representar conocimiento.*
- 4) *Etapas para generar conocimiento.*
- 5) *Etapas para establecer diálogos.*

Para construir estas etapas, se utiliza la herramienta de los Sistemas Evolutivos conocida como Lingüística Matemática, que contiene operaciones básicas de la teoría matemática, específicamente del Álgebra y que aplicadas a las oraciones de los lenguajes de trabajo (entre los que se va a traducir), es decir, Taquigrafía y Español, permiten obtener la gramática de cada lenguaje y con ellos, haciendo un reconocimiento de patrones por métodos sintácticos, es posible inferir las equivalencias entre las palabras de los lenguajes.

Palabras clave: Sistemas Evolutivos, Reconocimiento de formas, Reconocimiento de patrones, Lingüística Matemática.

¹¹ Irene Arzola Carvajal cuando realizó este trabajo era profesora del Instituto Tecnológico de Toluca además de cursar la maestría en Ciencias de la Computación en el Instituto Tecnológico de Toluca.

¹² José Rafael Cruz Reyes cuando realizó este trabajo era profesor de la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Toluca.

Las operaciones lingüísticas son:

- 1) *Factorización.*
- 2) *Conmutatividad.*
- 3) *Distribución.*
- 4) *Substitución.*
- 5) *Recursividad.*

Después de varios textos de ejemplo, se puede obtener la matriz evolutiva (figura 1).

Signo	Significado
'	el
└	gato
,	es
┐	negro
┌	blanco
ˆ	no
└┐	corre

Figura 1. Una Matriz Evolutiva

Utilizando la información de la matriz evolutiva de la figura 1 se puede traducir la oración siguiente:

'└ ,┌

por el significado correspondientes a cada símbolo, obteniendo:

el gato es blanco

Cada vez que se le da una oración a la máquina, se va generando una matriz evolutiva, la cual cuando se le indica un símbolo existente, almacenado en la tabla de reescritura, devuelve su respectiva interpretación; y cuando no lo conoce, pide su equivalencia, y con esta característica, hacemos del sistema, un Sistema Evolutivo, abierto al aprendizaje del conocimiento que se va conociendo, y además con la posibilidad de inferir nuevo conocimiento, a partir del ya adquirido, mediante técnicas de factorización.

IV.8 Sistema Evolutivo de Reconocimiento de Formas en Dos Dimensiones

Sergio Salcido Bustamante¹³

Diana Karla García García¹⁴

Alfonso Ventura Silva¹⁵

Introducción

La realidad es, ante el ser humano, un enorme mosaico de fenómenos en constante cambio. Algunos de ellos son tan sutiles que sólo la observación dedicada permite apreciarlos, en cambio, otros son determinantes. Sin embargo, todos los cambios del entorno, que son captados por el ser humano mediante alguno de sus mecanismos de percepción, son útiles para que éste pueda recrear en su mente un modelo único del medio que lo rodea.

Como el medio está en constante transformación, el modelo de la realidad de cada individuo comienza a evolucionar desde su nacimiento, y este proceso sólo concluye con la muerte. Las experiencias acumuladas en el transcurso de su vida lo preparan para responder cada vez en forma más adecuada a las transformaciones que le demandan una participación activa. Como resultado de este proceso, con el tiempo, el ser humano es capaz de dominar ciertos campos del entorno y de modificarlo de acuerdo a sus necesidades. De esta forma, la habilidad adquirida le permite interactuar continuamente en su entorno limitado, con un riesgo muy pequeño de encontrarse sin respuesta ante alguna circunstancia.

De la misma forma que el hombre evoluciona en su modelo de realidad, se pretende que los sistemas de información se adapten a entornos en constante cambio mediante la evolución del modelo de su entorno a medida que conoce más sobre él. En otras palabras, se concibe la idea de sistemas de propósito general dinámicos que tengan la disposición de “aprender” del medio en que se desarrollan mediante la integración de nuevas premisas en sus estructuras internas, las cuales comienzan a influir, en tiempo real, en los procesos que el sistema lleva a cabo, habilitándolo así para responder ante circunstancias distintas con modelos actualizados de su entorno.

¹³ Diana Karla García García,

¹⁴ Sergio Salcido Bustamante,

¹⁵ Alfonso Ventura Silva realizaron este trabajo cuando cursaban el 4to semestre de la licenciatura en la Escuela Superior de Cómputo del Instituto Politécnico Nacional.

Un sistema con los mecanismos informáticos que le aporten las cualidades descritas es llamado “Sistema Evolutivo” y es una alternativa para uno de los problemas principales de la informática actual, que consiste en la dificultad para mantener vigentes los sistemas de información, ya que, una vez finalizada la construcción de un sistema convencional (estático e inalterable) es difícil que éste responda a nuevos requerimientos, producto de cambios en el medio en que se desarrolla.

Por otra parte, en el campo de Inteligencia Artificial, uno de los problemas “clásicos” es el reconocimiento de formas, pues está íntimamente relacionado con procedimientos de percepción, abstracción y generalización.

No obstante, las herramientas convencionales para el reconocimiento de formas, son capaces, en el mejor de los casos, de aceptar formas “nuevas” mediante un proceso de reprogramación que permite al sistema adoptar una estructura modificada capaz de integrar la nueva forma en sus procesos de análisis. Sin embargo, la gran mayoría contienen una base de datos estática que reconoce sólo las formas que fueron consideradas y modeladas en su construcción.

Estos sistemas requieren además cantidades grandes de memoria puesto que no pueden integrar en un solo archivo los patrones descriptivos de formas similares, sino que los modelos iniciales son inalterables, y formas identificadas se destinan a nuevos archivos con el consecuente incremento en el uso de memoria.

El problema mayor es que generalmente se diseñan con un propósito específico, (identificación de huellas digitales, reconocimiento de imágenes, etc.) Aunque dos sistemas como estos tengan exactamente el mismo algoritmo de reconocimiento, son incapaces de intercambiar sus medios, imposibilitando su aplicación inmediata en campos diferentes.

En este documento, se describe un sistema evolutivo diseñado para reconocer formas, que no posee archivos predefinidos de imágenes, sino que almacena los datos que le envía el digitalizador en memoria y terminado este proceso compara la forma recibida con todas las formas que ya conoce, las cuales están enlistadas en un archivo.

En caso de que la forma sea completamente nueva para el sistema, ésta se “aprende” bajo la definición que el usuario proporcione y almacena en un archivo nuevo. Por el contrario, cuando la forma presenta gran similitud con alguna otra comprendida dentro de las ya conocidas, en lugar de ambas formas se crea una nueva, en la que se suman las características de cada una.

De este modo, se logra una actualización natural de la forma y con el tiempo, cada vez que se realiza este proceso, algunos rasgos o puntos inciden por encima de los demás, convirtiéndose en esenciales y ayudando a la definición del patrón correspondiente a esta forma.

I Hardware del sistema

El prototipo cuenta con un dispositivo externo que permite digitalizar formas trazadas en láminas de acetato; el digitalizador (scanner) construido es “de mesa” y tiene la cualidad de realizar el recorrido de las imágenes que se le coloquen de forma completamente automática en las 2 dimensiones empleadas (largo y ancho), a diferencia de un digitalizador manual, donde el movimiento en una dimensión debe ser realizado por el usuario. La técnica empleada en el digitalizador de mesa permite obtener imágenes más cercanas a la realidad, pues no existen movimientos donde alguna discontinuidad accidental pudiera provocar alteraciones a la representación de la imagen.

El funcionamiento del digitalizador es controlado por una computadora, que se encarga de enviar las señales necesarias para el movimiento de tres motores. Dos de ellos desplazan la lámina de acetato con la imagen a lo largo del digitalizador y el otro mueve a lo ancho un dispositivo sensor que va realizando la “lectura” de la información.

Los elementos que constituyen a este digitalizador pueden ser descritos a través de tres grandes bloques: dispositivos motores, dispositivos sensores, interfaz con la computadora.

1) *Dispositivos motores*: Se utilizan “motores a pasos”, los cuales poseen varias terminales por donde se introduce una secuencia de pulsos digitales (ceros y unos), que permite que el eje del motor dé un “paso”, es decir, un pequeño desplazamiento angular que siempre es constante. Para que el motor dé varios pasos basta introducir tantas secuencias como pasos se deseen. El eje puede dar una revolución completa en un número finito y constante de pasos. Para realizar la rotación en sentido contrario solo debe introducirse la secuencia de forma invertida.

El movimiento a lo largo del digitalizador se genera por dos motores a pasos y transmitido a través de un sistema de tracción sobre el cual se monta la lámina de acetato. Este sistema de tracción está constituido por dos ejes de acero, donde uno de ellos tiene en cada extremo un enlace con el rotor de un motor. Es conveniente aclarar que estos dos motores giran con la misma secuencia de pasos, de manera tal que sus rotaciones se dan con la misma velocidad y al mismo tiempo. El objetivo de esta particularidad es incrementar la potencia (par) en el giro, proporcionada por estos dispositivos. El segundo eje de acero está montado sobre la estructura de madera que sustenta a todos los elementos, de tal forma que gira sobre si mismo y es paralelo al eje enlazado con los motores. Es necesario que la lámina de acetato esté sostenida por algún elemento que al mismo tiempo le transmita el movimiento deseado, lo cual se logra por medio de un par de bandas plásticas colocadas cerca de los extremos de los ejes, estos últimos cuentan con ruedas dentadas en las regiones de contacto con la banda que suprimen el derrapamiento que pudiera existir y evitan su desplazamiento a lo largo de los ejes.

La distancia que separa a los ejes es el doble del ancho de una hoja tamaño carta, para que una lámina de acetato colocada horizontalmente pueda desplazarse a lo largo del digitalizador y se recorra en su totalidad.

El movimiento realizado a lo Ancho del digitalizador es generado por un tercer motor a pasos que recibe secuencias independientes de los otros dos anteriores. Este desplazamiento horizontal mueve al sensor. Para tal efecto se utiliza el carro de una impresora en el que se ha conservando intacto el mecanismo que desplaza la cabeza de impresión, en lugar de este dispositivo ha sido colocado el elemento que porta al sensor.

El recorrido de una lámina comienza con ésta colocada junto al eje motriz (o eje de tracción), el primer movimiento que se realiza es el desplazamiento del sensor a lo ancho del digitalizador, al llegar al extremo contrario del punto de inicio se produce un pequeño desplazamiento de la lámina en dirección al segundo eje, es decir, un movimiento a lo largo del dispositivo. Inmediatamente la secuencia de pulsos se invierte para el desplazamiento horizontal provocando que el sensor haga un recorrido en sentido contrario hasta llegar nuevamente al punto de inicio y en este momento vuelve a producirse otro pequeño movimiento a lo largo del digitalizador. Este ciclo se repite hasta que la lámina de acetato es recorrida en su totalidad. Ya que se ha terminado el proceso de recorrido, el eje motriz gira en sentido contrario para regresar la lámina a su posición original.

Es conveniente aclarar que la lámina de acetato no se coloca directamente sobre las bandas ya que no se sustenta por sí sola, por lo tanto se requiere utilizar un material que no se flexione con tanta facilidad, tal como la mica. Por su relativa rigidez y transparencia, este es el material ocupado para apoyar la lámina.

2) *Dispositivos sensores:* En el apartado anterior se ha considerado al sensor como un solo elemento, pero requiere de dispositivos auxiliares para un buen funcionamiento. Los elementos que se desplazan a lo ancho del digitalizador son: un sensor de luz infrarrojo y un diodo emisor de luz roja visible.

El sensor recibe constantemente un haz de luz que tiene la capacidad de atravesar la lámina de acetato, excepto en las regiones que tengan algún trazo en negro con el suficiente grosor e intensidad. La configuración del circuito que controla al sensor le permite entregar un uno lógico cuando recibe al haz de luz, pero cuando por alguna obstrucción el haz es interrumpido, el circuito proporciona un cero lógico. La información producida es enviada a la computadora al momento que se genera. El elemento montado sobre el carro de impresora soporta al sensor y al emisor de luz por medio de dos varillas de aluminio, la disposición de estas piezas permite al sensor estar suspendido sobre la lámina de acetato a una distancia de 28 cm. del mecanismo que le proporciona movimiento y al emisor justo frente a él por debajo de la lámina.

En vez de ocupar luz infrarroja para excitar al sensor se optó por la luz roja visible, pues al experimentar con diversos tipos se encontró que la primera traspasa incluso materiales traslúcidos con trazos en su superficie, en cambio, el rendimiento del sensor se reduce con el segundo tipo de luz y por consiguiente es mas fácil evitar su detección con las condiciones de obstrucción ya mencionadas. Los trazos deben ser realizados en láminas de acetato porque son completamente transparentes y esta característica permite que el haz luminoso penetre sin mayor problema en las regiones libres de tinta negra. El emisor de luz roja envía su haz en forma de abanico, lo que disminuye la sensibilidad, pues las líneas muy delgadas no pueden

detectarse. Para lograr una sensibilidad mayor se requeriría la emisión de luz coherente, tal como la proveniente de un láser y un sensor adecuado para ésta.

3) *Interfaz con la computadora*: El software que controla al digitalizador tiene por salida las secuencias de pulsos que activan a los motores y como entrada a la señal proveniente del sensor. Este intercambio de datos se realiza a través del puerto paralelo. De la terminal dos a la cinco se obtienen los pulsos para el motor que mueve al sensor, de la seis a la nueve surgen los correspondientes para los motores que desplazan a la lámina de acetato. En la terminal quince entra la señal que envía el sensor. La terminal 25 es utilizada como referencia (tierra).

Es conveniente notar que como el voltaje y la corriente proporcionadas por la computadora son insuficientes para activar directamente a los motores, debe colocarse una etapa de protección y otra de amplificación de potencia entre el puerto de la computadora y los motores.

La fase de protección está constituida por un optoacoplador conectado a cada línea de salida de la computadora. Este dispositivo transfiere cualquier señal que se presente a su entrada por luz infraroja (IR) dentro del encapsulado, aislando eléctricamente a la computadora de los voltajes y corrientes para motores.

La potencia de las señales que provienen de la etapa de protección es incrementada al voltaje y corrientes necesarios por medio de transistores.

La señal que envía el sensor pasa por una compuerta lógica (inversor) para evitar que la entrada de la computadora reciba señales intermedias entre ceros o unos.

II Sistemas evolutivos

Un sistema evolutivo no conserva modelos fijos de la realidad sino que es capaz de reconfigurarse internamente al interactuar con el medio ambiente. Su reconfiguración interna es posible ya que posee los mecanismos que le permiten:

- 1) *Adquirir conocimiento progresivo*: Es capaz de iniciar su operación sin conocer ninguna forma y crear archivos para las formas que considere nuevas.
- 2) *Establecer relaciones entre los conocimientos adquiridos*: Está habilitado para reconocer nexos entre las formas ya conocidas y las recientes, en términos de similaridad cuantificada por medio de un valor distancia, descrito más adelante.
- 3) *Aprender el lenguaje para comunicarse con las personas*: Utiliza procedimientos de reescritura para indicar al usuario en su propio lenguaje que la forma digitalizada posee un nombre si es reconocida o, en su defecto, para solicitarle uno.
- 4) *Inferir y abstraer nueva información*: Con cada forma que digitaliza, es capaz de abstraer información nueva, la cual se traduce en un renglón más en la lista de formas o, en la misma lista, pero con un archivo que contiene una acumulación más de la forma respectiva, lo que implica un patrón más definido del objeto real que representa.

5) *Comunicarse con el medio ambiente:*. Esta capacitado para establecer comunicación con el usuario, por medio de la interfaz de software y con el objeto de estudio por medio del digitalizador.

III El núcleo del sistema

Las características evolutivas son factibles de implementarse en un programa para computadora si se considera una independencia relativa entre los elementos básicos del sistema: señales de entrada, estructura del sistema, procesos del sistema, datos del sistema, señales de salida, manejador del sistema.

1) *Señales de entrada:* En nuestro caso están representadas por los impulsos recibidos en el puerto paralelo que son interpretados como la presencia o ausencia de un punto sobre la lámina que se digitaliza. Para otros sistemas, esta señal puede ser desde una cadena de caracteres hasta señales de voz, sin que ello afecte la validez del esquema de elementos.

2) *Estructura del sistema:* En nuestro caso es la lista de imágenes conocidas, y en general se denomina “la gramática del sistema”. Básicamente es un conjunto de reglas de reescritura permitidas para el sistema. Aquí es importante destacar que estas reglas son susceptibles de modificarse o incrementarse en número para integrar los nuevos conocimientos que adquiera el sistema. Este sistema reescribe una forma digitalizada con un nombre correspondiente. Sin embargo es igualmente válido reescribir por ejemplo, una señal de voz con la imagen en pantalla del poseedor de la misma, o viceversa.

3) *Procesos del sistema:* Son las rutinas que desempeñan diversas funciones encaminadas a la obtención de resultados, y se describen más adelante por grupos.

4) *Datos del sistema:* Son las representaciones en archivos de las imágenes del sistema. Como se ha comentado, las imágenes similares se acumulan en un solo archivo representativo de todas. Para otros sistemas, estos datos son también representaciones para la computadora del tipo de señales que maneja.

5) *Señales de salida:* Son los mensajes en forma de cadenas de caracteres mediante los cuales el sistema informa al usuario de los resultados del proceso de reconocimiento. En otros sistemas puede contarse con otros dispositivos físicos (como el digitalizador) conectados como periféricos a la computadora para representar mediante voz, luz, imagen, etc. a los resultados.

6) *Manejador del sistema:* Se encarga de controlar todo el sistema mediante la definición de la secuencia y el llamado al resto de los elementos en el momento que se requieren.

La independencia relativa se refiere al hecho de que las modificaciones que se produzcan en alguno de ellos, no afecten al resto de los elementos.

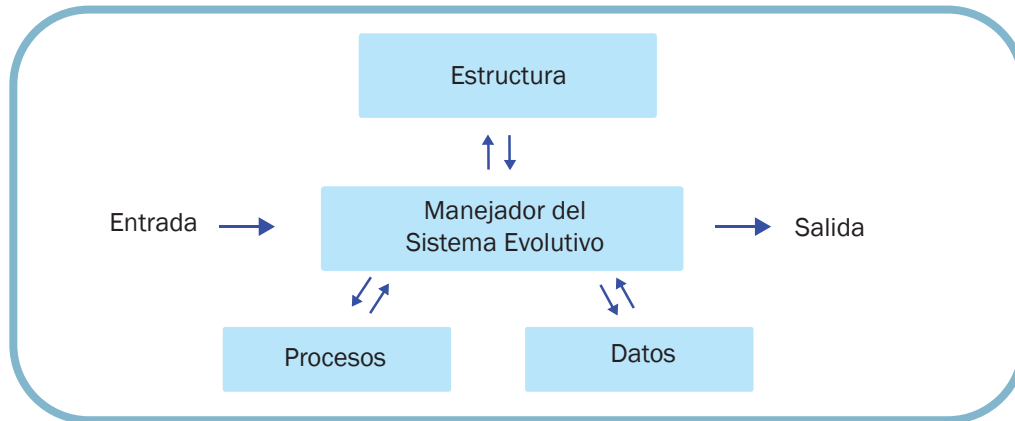


Figura 1. Estructura básica de un Sistema Evolutivo.

IV Matrices evolutivas

Hasta aquí se han observado las características que hacen evolutivo al sistema, sin embargo, el motivo por el cual fue creado es el reconocimiento de formas. Para tal efecto el sistema requiere establecer un formato de captura para las mismas, a partir del cual pueda llevar a cabo las operaciones básicas de:

- 1) *Comparación.*
- 2) *Suma.*
- 3) *Agregación.*

Antes de describir las operaciones es necesario destacar el detalle del formato. El archivo que contenga una imagen se identifica con la extensión .IMG y consta solo de tantos caracteres como puntos detecta el digitalizador. La disposición es por líneas, que representan el barrido con el carro de impresora, de forma que la vista del archivo bien puede interpretarse por el ojo humano. Cuando la forma es nueva, los caracteres solo son '1' y '0'. Pero cuando alguna forma posterior se identifica con la representada por algún archivo, la suma de ellas implica que se representen, en el mismo archivo y conservando su dimensión, los puntos donde ninguna imagen incidió, los que fueron incididos solo por una de ellas y los incididos por ambas. Respectivamente se usan los caracteres "0", "1" y "2". De la misma forma se tratan las incidencias de n formas, empleando los caracteres ASCII subsecuentes al "9", pues al restar al entero ASCII el número 48, se obtiene el número de imágenes que incidieron en el punto de que se trate. Por el momento, n está limitado al alcance ASCII, es decir 200 incidencias, que para propósitos prácticos consideramos suficiente.

Establecido lo anterior, el proceso de comparación requiere extraer de la gramática el número de formas acumuladas para cada archivo que vaya comparando con la forma en memoria proveniente del digitalizador. Con este factor evalúa, punto por punto de las matrices respectivas, la coincidencia obteniendo en cada caso una distancia puntual que se acumula en una variable. La fórmula para este valor es:

$|p.archivo - (factor)(p.digitalizado)|$

donde $p.archivo$ es el punto de la matriz de la forma del archivo en turno, $factor$ son las formas que se han acumulado en ese archivo y $p.digitalizado$ es el punto de la matriz de la forma recién digitalizada.

Hay cuatro casos posibles de incidencia:

- 1) *La imagen digitalizada y la del archivo inciden en el punto.* En este caso, aunque el valor del archivo sea mayor que 1 (por las formas que vayan acumuladas), gracias al factor la diferencia es cero y por lo tanto no se acumula diferencia.
- 2) *Ninguna de las dos imágenes incide en el punto.* La diferencia es entonces cero, manteniendo inalterada la distancia que vaya acumulada hasta ese momento.
- 3) *La imagen del archivo incide y la del digitalizador no.* El peso del punto en el archivo determinará el valor de la diferencia, por lo que el incremento en la distancia acumulativa es proporcional a lo importante del punto.
- 4) *La imagen digitalizada incide y la del archivo no.* Entonces, como se sabe por el factor acumulativo la importancia de que ese punto no sea incidido, la diferencia es negativa y de valor proporcional al factor mencionado. Por el valor absoluto aplicado el efecto en la distancia es de incremento también.

Una vez acumulada la distancia de todos los puntos se evalúa la razón de este valor con respecto al valor matricial de los puntos de la imagen de archivo. Si se excede de cierto valor la imagen digitalizada se juzga diferente y se inicia el proceso de agregación. De lo contrario, la imagen se declara semejante y se realiza el proceso de suma.

El proceso de agregación implica crear un nuevo archivo para la imagen no identificada, así como insertar su nombre en la lista de imágenes para que tome parte, en adelante, de los procesos de comparación. Obviamente el número asociado de imágenes acumuladas es uno.

El proceso de suma modifica los datos o conocimientos del sistema al cambiar los símbolos ASCII del archivo que se relacionó con la imagen proveniente del digitalizador, por el valor siguiente en ASCII, con lo cual se asegura que en análisis posteriores el punto será considerado aún más importante de lo que ya era, dado que una imagen más (la reciente) coincide en él. Además es necesario alterar el factor de imágenes acumuladas que se guarda en el listado de imágenes para los mismos fines.

V Software del Sistema

El programa principal inicia la comunicación con el usuario mediante una pantalla de presentación. Luego aparece una pantalla con las opciones de Digitalizar Imagen o Salir del Programa, la elección Digitalizar presenta la cuadrícula sobre la que se graficará la imagen, e inicia la Digitalización. Una vez terminada se comienza con el proceso de Reconocimiento de la imagen, finalmente una vez que la imagen ha sido reconocida o registrada, los archivos

correspondientes son almacenados y se vuelve a la pantalla de opciones, desde la cuál se puede salir al sistema operativo, no sin antes cerrar el modo gráfico y limpiar la pantalla.

Las funciones básicas del programa son: interfaz con el usuario, manejo de archivos, digitalización, graficación, reconocimiento, apoyo.

1) *Interfaz con el usuario*: Tiene las funciones de mensajes, de fondo y de ventana. La función de mensajes es un despachador de mensajes que se envían al usuario a través de la pantalla. Normalmente el usuario recibe ocho mensajes. La función de fondo dibuja en el modo gráfico de vídeo el fondo de la pantalla, por lo tanto es llamada cada vez que se necesita actualizar la misma. La función de ventana forma el recuadro de colores Magenta-Cyan en el cuál se visualizan todos los mensajes, consecuentemente dicha función es llamada cada vez que se envía un mensaje.

2) *Manejo de archivos*: Incluye las funciones de manejo de archivos. La función carga archivo se encarga de decidir si el archivo que va a procesar es el correspondiente a una imagen, o el listado de formas. Las acciones que realiza son: abrir para lectura un archivo de texto, llamar a la función de mensajes en caso de que el archivo requerido no se pueda abrir por alguna causa, guardar en un arreglo de memoria dinámica el archivo a comparar y cerrar el archivo. En caso de tratarse de archivos de imágenes de manera adicional: determina el número de imágenes acumuladas en el archivo. La función para salvar archivos se encarga de grabar los datos contenidos en los arreglos de memoria dinámica que pueden ser el arreglo de una imagen que acaba de ser digitalizada, o una imagen cuyo archivo ya existía.

3) *Digitalización*: Hace posible que el proceso de digitalización y visualización en pantalla de la forma sea en tiempo real, gracias a que controla el movimiento de los motores, y la señal enviada por el sensor del digitalizador. El contenido de la dirección de entrada del puerto se enmascara para detectar el estado del pin que esta recibiendo la señal. El número de movimientos en cada dimensión queda definido por las variables globales Ancho y Largo. La cantidad de pasos máxima para cada motor queda entonces limitada y por lo tanto la resolución del digitalizador.

4) *Función de Secuencias*: Esta función es llamada por la función digitaliza y envía directamente las señales de salida al puerto, con tiempos de retardo diferente, dependiendo del motor que se debe controlar.

5) *Graficación*: Tiene la función de Cuadrícula que dibuja en la pantalla una cuadrícula con dimensiones Largo por Ancho en unidades de x pixeles, donde x se define en tiempo real de acuerdo a las características del monitor con que se cuenta. La función de punto se encarga de dibujar sobre la cuadrícula el punto proveniente de la imagen que esta siendo digitada, en tiempo real. Recibe las coordenadas y el color de dibujo como parámetros.

6) *Reconocimiento*: Involucra las funciones para los procesos de Comparación, Suma y Agregación descritos anteriormente.

7) *Apoyo*: Son básicamente las funciones de: escape, visualización de cadena en modo gráfico, obtención del nombre de la forma a comparar y conversión de un número flotante a cadena.

Conclusiones

Consideradas las características de los sistemas evolutivos se concluye que son una alternativa muy conveniente para desarrollo de aplicaciones de Inteligencia Artificial como el Reconocimiento de Formas. Para los usuarios son sistemas que integran automáticamente los conocimientos y los modifican sin necesidad de recurrir al programador para que los reconfigure. Sin embargo, a pesar de lo promisorio de los sistemas evolutivos no debe olvidarse que, para todo sistema, por más avanzada que sea la tecnología en que se apoya, no es capaz de conjuntar mas que unas cuantas imitaciones (muy malas) de los mecanismos humanos de percepción, por lo cual las restricciones en cuanto a analogías con el ser humano, son evidentes.

En el campo del reconocimiento de formas, la herramienta desarrollada es de especial utilidad si se considera que se pueden construir bases de datos suficientemente grandes en múltiples áreas del conocimiento que almacenen imágenes. Elementos desconocidos pueden ser identificados y clasificados. Por otra parte, sucesivos ejemplos de una clasificación conocida pueden introducirse al sistema con el fin de obtener un patrón general. Las perspectivas de desarrollo de este prototipo son grandes, sobre todo en cuanto a resolución e información sobre la imagen se refiere (color, precisión, etc.).

Bibliografía

- [1] Chomsky, Noam, Estructuras Sintácticas. Ed. Siglo XXI.
- [2] Bach, Emmon, Teoría Sintáctica. Ed. Anagrama.
- [3] Salomaa, Formal Languages, Ed. Academic Press.
- [4] Galindo Soria, Fernando, Sistemas Evolutivos, IPN-UPIICSA.
- [5].Galindo Soria, Fernando, Generador de Sistemas como Núcleo de un Sistema Evolutivo. Memorias del Festival de Ingeniería TECCONT 90, ITESM-CEM.
- [6] Galindo Soria, Fernando, Una Representación Matricial para Sistemas Evolutivos, IPN-UPIICSA.
- [7] Galindo Soria, Fernando, Sistemas Evolutivos: Nuevo Paradigma de la Informática. Memorias de la XVII Conferencia Latinoamericana de Informática, Caracas, Venezuela. 1991.

IV.9 Sistema Evolutivo Reconocedor de Textos

Jesús Manuel Olivares Ceja¹⁶

Resumen

En muchas aplicaciones informáticas se requiere de interpretar información como señales, voz, sonido, etc., para esto existen diversas técnicas, se presenta aquí una propuesta para el reconocimiento de símbolos basada en matrices evolutivas. Para esto se calcula la distancia entre la entrada y cada patrón de la matriz evolutiva y se selecciona el de mayor similitud; sino se alcanza el mínimo de semejanza aceptable se anexa la entrada como patrón nuevo. Una variante del método presentado puede aplicarse al reconocimiento de otras señales como voz, imágenes de colores, etc.

Introducción

En muchas aplicaciones informáticas se requiere de poder interpretar información como señales, imágenes, voz, sonido, etc. asignándole un significado a cada una. Existen diversas alternativas para realizar lo anterior, en este trabajo se presenta una basada en matrices evolutivas.

En este tipo de problemas lo primero que se debe hacer es percibir la información mediante el sensor apropiado: cámara de vídeo, micrófono, scanner, transductor de presión, etc., luego se digitaliza y posteriormente se aplica alguna técnica de reconocimiento.

Existen técnicas de reconocimiento de patrones basadas en redes neuronales, otras siguen el enfoque del reconocimiento sintáctico de patrones.

La propuesta que se expone en este documento se basa en matrices evolutivas, las cuales surgen de la integración de estudios sobre redes neuronales, conjuntos difusos y sistemas evolutivos.

I Adquisición/digitalización de la información

En varias aplicaciones informáticas se requiere de interpretar información proveniente de diversas fuentes como son señales, imágenes, voz, sonidos, presión, temperatura, etc. La información puede recibirse en forma analógica o digital, mediante un sensor apropiado.

¹⁶ Jesús Manuel Olivares Ceja diciembre 1996.

En caso de ser analógica debe digitalizarse, es decir convertirla a números dentro de un intervalo dado indicado como $[a,b]$. La información digitalizada se puede procesar entonces mediante una computadora digital (figura 1).


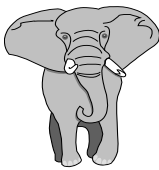

Señal original			
Señal digitalizada	1 0 0 1 0 0 . . .	0 0 0 24 24 0 . . .	0 2080 50 30 . . .
Intervalo	[0,1] 1 bit/pixel	[0,255] 8 bits/pixel	[0,65535] 16 bits/muestra
Significado	Letra: X	Imagen: Elefante	Sonido: Hola

Figura 1. Información y su digitalización.

La información digitalizada se coloca en un vector para usarla como entrada al reconocedor evolutivo y que este indique el significado asociado a la información de entrada o bien que reestructure la matriz evolutiva para que se incluya la entrada nueva (figura 2).

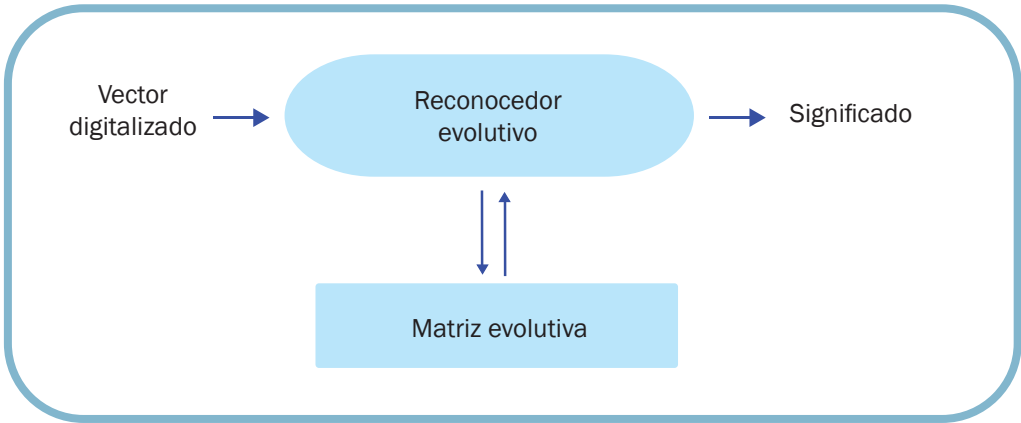


Figura 2. Reconocedor Evolutivo.

Para convertir a un vector una matriz, cada fila (renglón) de información se coloca uno delante del siguiente, como se muestra en el ejemplo del símbolo 2 digitalizado (figura 3).

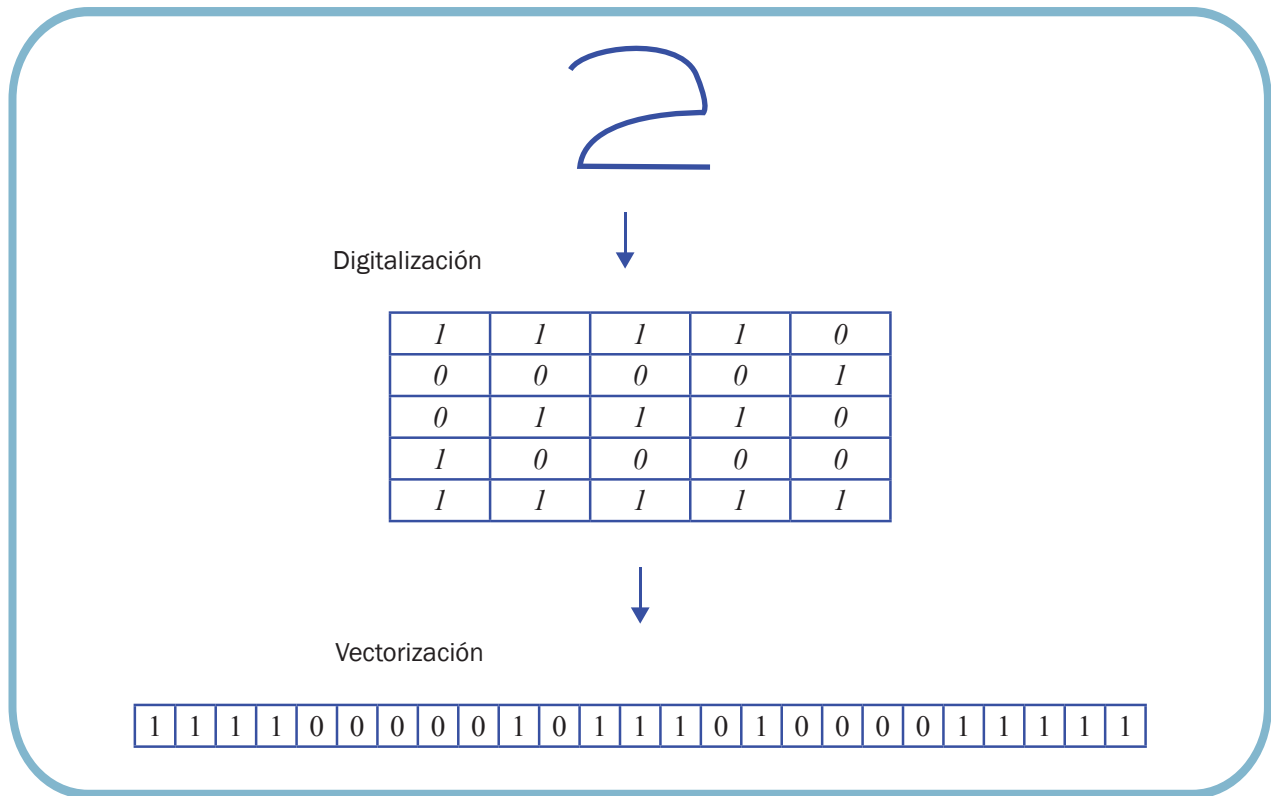


Figura 3. Vectorización del símbolo 2 digitalizado.

II Búsqueda/aprendizaje del patrón

La señal digitalizada se normaliza para obtener valores en el intervalo $[0,1]$ dividiendo cada uno de los elementos que forman al vector (X_i) entre el valor máximo que puede tomar la señal, para el ejemplo de la figura 3, el valor máximo es 1, dado que se trata de una imagen binaria.

El vector normalizado se compara contra cada uno de los patrones almacenados en la matriz evolutiva, la cual consiste de una columna de patrones, un umbral y el significado de cada elemento de la misma (figura 4). El umbral indica cuantos patrones han coincidido con el que se muestra ahí.

x_0	x_1	x_2	x_3	x_4	\dots	x_{n-1}	h	significado
2	3	3	3	2		0	3	tres
2	1	2	2	0		2	2	cuatro
2	4	0	0	0		1	4	uno

Figura 4. Una matriz evolutiva.

Para comparar con cada patrón se usa la distancia siguiente:

$$S = \sum_{i=0}^{n-1} 1 - |X_i \div \max(X) - x_i \div h|$$

donde n es el número de elementos que tiene el vector de entrada (X) y coincide con el número de elementos de cada patrón (x) de la matriz evolutiva; S, es la suma de las semejanzas; max(X), es el valor máximo que puede tomar un elemento del patrón; el término que se suma (dado como 1 - el valor absoluto de la diferencia) es la semejanza entre la entrada y el patrón de la matriz evolutiva.

Para conocer el porcentaje de semejanza entre 0 y 100% se procede a calcular el factor de semejanza (α) como sigue:

$$\alpha = S \div n * 100\%$$

Para cada fila de la matriz evolutiva se obtiene el factor α . De todos los α , se selecciona el mayor y se compara contra el porcentaje mínimo de semejanza aceptable (δ , que debe indicar el usuario para su tipo de información que maneja). Si $\alpha \geq \delta$, entonces se refuerza el patrón existente en la matriz evolutiva sumando el valor de cada X_i y sumándole uno al umbral (h). Si $\alpha < \delta$ se solicita al usuario que indique el significado del símbolo desconocido y se quiere anexarlo en la matriz evolutiva, si lo adiciona, h es uno, significado, lo digita el usuario y cada x_i del patrón nuevo se copia de cada X_i del vector de entrada.

Un ejemplo del cálculo de semejanza es el siguiente: supóngase que se tiene en la entrada el vector del patrón 2 (figura 3), y en la matriz evolutiva se tiene un patrón llamado tres con umbral 3 y digitalizado como se observa en la figura 5

2 3 3 3 2 0 0 0 1 2 0 2 3 3 1 0 0 0 0 3 2 3 3 3 0

Figura 5. Patrón llamado tres en la matriz evolutiva con h=3.

Para obtener la semejanza entre ambos patrones se calcula la semejanza para cada punto con la fórmula indicada anteriormente como sigue:

$$\begin{aligned} S = & (1 - |1 \div 1 - 2 \div 3|) + (1 - |1 \div 1 - 3 \div 3|) \\ & + (1 - |1 \div 1 - 3 \div 3|) + (1 - |1 \div 1 - 3 \div 3|) \\ & + (1 - |0 \div 1 - 2 \div 3|) + (1 - |0 \div 1 - 0 \div 3|) \\ & + (1 - |0 \div 1 - 0 \div 3|) + (1 - |0 \div 1 - 0 \div 3|) \\ & + (1 - |0 \div 1 - 1 \div 3|) + (1 - |1 \div 1 - 2 \div 3|) \\ & + (1 - |0 \div 1 - 0 \div 3|) + (1 - |1 \div 1 - 2 \div 3|) \\ & + (1 - |1 \div 1 - 3 \div 3|) + (1 - |1 \div 1 - 3 \div 3|) \\ & + (1 - |0 \div 1 - 1 \div 3|) + (1 - |1 \div 1 - 0 \div 3|) \end{aligned}$$

$$\begin{aligned}
&+ (1 - |0 \div 1 - 0 \div 3 \div|) + (1 - |0 \div 1 - 0 \div 3|) \\
&+ (1 - |0 \div 1 - 0 \div 3 \div|) + (1 - |0 \div 1 - 3 \div 3|) \\
&+ (1 - |1 \div 1 - 2 \div 3 \div|) + (1 - |1 \div 1 - 3 \div 3|) \\
&+ (1 - |1 \div 1 - 3 \div 3 \div|) + (1 - |1 \div 1 - 3 \div 3|) \\
&+ (1 - |1 \div 1 - 0 \div 3 \div|)
\end{aligned}$$

$$S = 2 \div 3 + 1 + 1 + 1 + 1 + 1 \div 3 + 1 + 1 + 1 + 2 \div 3 + 2 \div 3 + 1 + 2 \div 3 + 1 + 1 + 2 \div 3 + 0 + 1 + 1 + 1 + 0 + 2 \div 3 + 1 + 1 + 1 + 0$$

de donde resulta:

$$S = 19 + 1 \div 3 \cong 19.3333$$

por lo que el factor de semejanza es:

$$\alpha = 19.3333 \div 25 * 100\% = 77.33\%$$

Si $\delta = 95\%$ entonces como $\alpha < \delta$ y suponiendo que con el patrón llamado tres es con quien tuvo la máxima semejanza (el mayor α), entonces pide al usuario que digite el significado de la entrada y se adiciona como un renglón más en la matriz evolutiva. Si en caso dado, hubiera resultado $\alpha \geq \delta$, la respuesta hubiera sido, “es un tres”.

III Eliminación de Ruido

Después de un proceso de reconocimiento/reforzamiento los patrones de la matriz evolutiva pueden tener un umbral (h) grande como 50000, en estos casos lo que puede hacerse es restar un factor (por ejemplo 40000), los elementos que resulten negativos se asignan con cero, de esta forma algunos puntos que dentro del patrón sólo eran ruido se eliminan quedando únicamente los puntos relevantes del patrón.

IV Un programa que Reconoce Vectores Digitalizados

En el listado 1, se muestra un programa que reconoce símbolos de acuerdo a lo presentado en la sección 2 de este documento. La entrada es un vector con 25 elementos, nombrado X_1 cada elemento (**se supone que la información está digitalizada y vectorizada**).

Listado 1. Un programa basado en matrices evolutivas para el reconocimiento de símbolos:

```

#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>

```

```
// ENTRADA:
```

```

# define MAX_X 5
# define MAX_Y 5
# define VALOR_MAXIMO 1
int X[MAX_X * MAX_Y] ;
int IX ;

// MATRIZ EVOLUTIVA
# define MAX_MATRIZ 200
typedef struct {
    int x[MAX_X * MAX_Y] ;
    int h;
    char significado[25] ;
} TipoMatriz ;
TipoMatriz Matriz[MAX_MATRIZ] ;
int UMatriz ,IMatriz ,IMasSemejante ;
double Delta ,Alfa ,AlfaMasSemejante ,S ;
char Linea[100] ;

void main(void)
{
int sigue ;
printf("Matriz Evolutiva de reconocimiento de señales\n\n");
printf("% % Minimo de semejanza aceptable: ");
gets(Linea);
Delta = atoi(Linea);
// INICIALIZA VARIABLES
UMatriz = 0;
sigue = 1;
do
{
    printf("Vector del patron (FIN para salir):\n");
    gets(Linea);
    strupr(Linea);
    if( strcmp(Linea,"FIN") == 0 )
        sigue = 0;
    else
    {
        for( IX = 0; IX < MAX_X * MAX_Y; IX++ )
            X[IX] = Linea[IX] - '0';
        // BUSCA/APRENDE EL PATRON EN LA MATRIZ EVOLUTIVA
        AlfaMasSemejante = 0.0;
        IMasSemejante = -1;
        for( IMatriz = 0; IMatriz < UMatriz; IMatriz++ )
        {
            S = 0.0;
            for( IX = 0; IX < MAX_X * MAX_Y; IX++ )
                S += (1.0 - abs((double)X[IX] / (double)VALOR_MAXIMO - Matriz[IMatriz].x[IX] /
Matriz[IMatriz].h));
            Alfa = S / (double)(MAX_X * MAX_Y) * 100.0;
            if( Alfa > AlfaMasSemejante )
            {

```

```

        IMasSemejante = IMatriz;
        AlfaMasSemejante = Alfa;
    }
}
if( IMasSemejante != -1 && AlfaMasSemejante >= Delta )
{
    printf("El simbolo tiene %3.1f%% de semejanza por lo que es: %s\n",AlfaMasSemejante
,Matriz[IMasSemejante].significado);
    // SE REFUERZA EL PATRON
    for( IX = 0; IX < MAX_X * MAX_Y; IX++ )
        Matriz[IMasSemejante].x[IX] += X[IX];
    Matriz[IMasSemejante].h++;
}
else
{
    printf("Digite el significado del simbolo: ");
    gets(Matriz[UMatriz].significado);
    Matriz[UMatriz].h = 1;
    for( IX = 0; IX < MAX_X * MAX_Y; IX++ )
        Matriz[UMatriz].x[IX] = X[IX];
    UMatriz++;
}
}
} while( sigue == 1 );
printf("Gracias\n");
} // fin del listado

```

Conclusiones

Para resolver el problema de la interpretación de información como señales, imágenes, etc. se ha presentado una propuesta basada en matrices evolutivas, ejemplificándolo con el reconocimiento de símbolos, el mismo algoritmo puede aplicarse al reconocimiento de voz o de otro tipo de señales.

Bibliografía

[1] GALINDO Soria, Fernando, UNA REPRESENTACIÓN MATRICIAL PARA SISTEMAS EVOLUTIVOS, México, 1992.

CAPÍTULO V

SISTEMAS EVOLUTIVOS APLICADOS AL TRATAMIENTO DE IMÁGENES

V.1 Algoritmo Evolutivo para Tratamiento de Imágenes

Angel Cesar Morales Rubio¹

Introducción

El presente trabajo es el resultado de seis años de investigación, que como se muestra, abarcó una amplia gama de especialidades o áreas de diversa índole, debido al interés particular por hallar un nuevo método de Reconocimiento de Patrones, que implementado en un Sistema Evolutivo para Tratamiento de Imágenes, lleve a éste a basar su funcionamiento de análisis en las características intrínsecas que éstas presentan.

Esta nueva alternativa responde a la necesidad de sintetizar una imagen de manera global con cierta naturalidad, teniendo un enfoque de paralelismo en la consecución de este objetivo.

Los frutos de esa investigación se encuentran contenidos en el presente documento estructurados en tres partes que son: Antecedentes, Análisis y Desarrollo.

I Antecedentes

Reconocimiento de Patrones es el proceso por el cual sintetizamos heurísticamente la estructura de un objeto, encontrando así la firma o patrón característico de éste.

Pueden encontrarse algoritmos para reconocimiento de patrones que sin duda alguna cumplen con el objetivo de encontrar dicha firma en los objetos sobre los cuales actúan, éstos se apegan comúnmente a los siguientes lineamientos:

- 1) Se basan en refinamientos de métodos predecesores a ellos pero que siguen teniendo como origen una misma idea.
- 2) No captan de manera global el “significado” o “patrón” de una imagen.
- 3) En general el método de solución no utiliza paralelismo.

La mayoría de estos algoritmos tienen la peculiaridad de representar las imágenes del mundo exterior en un espacio de dos dimensiones, es decir, utilizan una matriz finita como modelo de

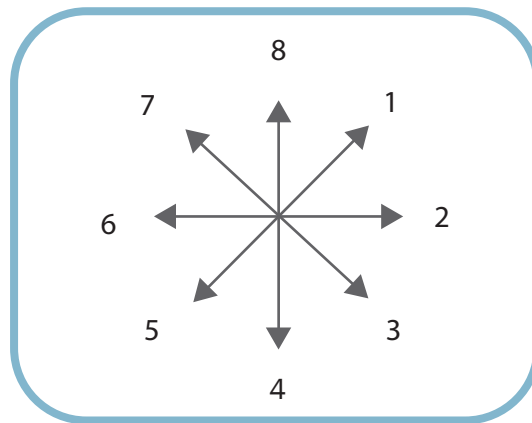
¹ Angel Cesar Morales Rubio elaboró este trabajo como parte de su tesis de licenciatura en ciencias de la informática en la UPIICSA-IPN.

representación. Esto es perfectamente lógico puesto que las figuras que intentamos reconocer “viven” en el mundo real aparentemente en un espacio de dos dimensiones. Por ello, una gran variedad de algoritmos (sin importar las diferencias en cuanto a su implementación) utilizan esta misma mesa de trabajo.

A continuación se describen las características más importantes de algunos algoritmos para Reconocimiento de Patrones, así como un resumen de sus principales ventajas y desventajas.

1 Número de Forma

Este procedimiento asume que dos puntos consecutivos de una imagen sólo pueden estar conectados entre sí en ocho direcciones, en similitud a los puntos cardinales:



Este método inicia tomando algún punto de la figura y marcándolo como inicial (figura 1), y si escogemos un recorrido en el sentido de las manecillas del reloj, podemos decir que existe una conexión de tipo 2 entre el punto de INICIO y su vecino derecho, siguiendo este procedimiento todo el objeto esta representado por la oración 2222222244444666666666688888 o simplificando: (9)2(5)4(9)6(5)8

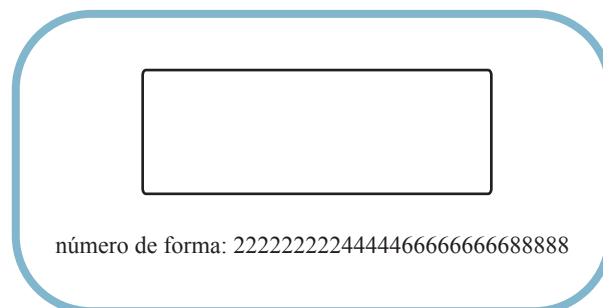


Figura 1. Una imagen y su número de forma.

Así, si aplicamos el proceso en sentido inverso, (a partir de la secuencia de números) estaremos en la posibilidad de generar la imagen original; haciendo un análisis de este procedimiento encontramos dos problemas:

- 1) La síntesis de una *imagen sólida* requiere de la aplicación de algunos procedimientos adicionales.
- 2) Este método es altamente *sensible al factor de escala*.

2 Esqueletos de una Región

Dado un objeto se divide este en lo que corresponde:

- 1) *Al borde*.
- 2) *Al interior*.

entonces, para cada punto en el interior se encuentra su vecino más cercano en el borde, si tiene más de un vecino se toma éste como el punto medio y pasa a formar parte del esqueleto del objeto, en caso contrario este punto se desecha y se toma el siguiente (figura 2). Resulta prácticamente imposible realizar una implementación directa de éste en un equipo de cómputo, ya que como se puede apreciar se tiene que estudiar cada punto individual y compararse con varios bordes en el peor de los casos, incrementándose en forma dramática el cálculo de distancias, además de existir objetos que presentan dificultades, como es el caso de círculos, y figuras abiertas.

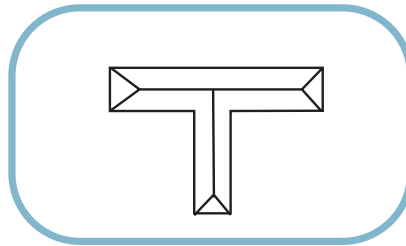


Figura 2. Esqueleto de una región.

3 Cuatrices

El método toma como origen una figura en una matriz comúnmente cuadrada, y la divide en cuatro regiones iguales, como si fuera un plano cartesiano (figura 3), de ahí analiza cada cuadrante y verifica si hay información en él, si la hay divide este cuadrante a su vez en cuatro partes como sucedió originalmente, y aplica el mismo procedimiento en forma recursiva hasta encontrar un cuadrante que ya no se puede dividir, y después de haber analizado todos los cuadrantes de todos los niveles se obtiene una estructura en árbol que representa el objeto original:

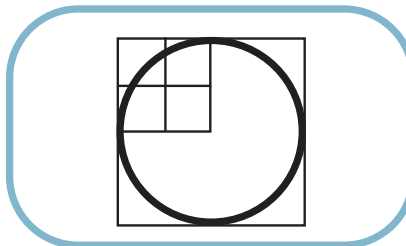


Figura 3. Una imagen con cuatrices.

1) La *implementación* claramente puede realizarse en un ambiente de múltiples procesadores puesto que se puede asignar un procesador a cada uno de los cuadrantes que se vayan encontrando, existiendo la limitante teórica de crear un equipo de cómputo con un número ilimitado de procesadores, lo que claramente marca una deficiencia en la funcionalidad del método, respecto al número de cuadrantes a procesar.

2) *El patrón o firma de la imagen* parece estar compuesta por una estructura de árboles, siendo la raíz el primer nivel o imagen original consiguiendo de esta manera un esqueleto, como en el método anterior, pero con una representación diferente.

II Análisis

Con el fin de encontrar un método de Reconocimiento de Imágenes que sintetize a éstas, se conjugaron tres métodos naturales de modelación de la realidad, estos métodos son: Visión Humana, Redes Neuronales y Fractales.

1 Visión Humana.

Definitivamente el ser Humano parece ser una de las criaturas más perfeccionadas en este planeta en cuanto a su anatomía se refiere. Desde un punto de vista funcional, nuestros ojos dividen el campo visual en cuatro regiones simétricas. Esto es gracias a la distribución de las conexiones entre las fibras ópticas y la corteza visual del cerebro.

De lo anterior podemos concluir el siguiente hecho relevante en cuanto al sistema de captación de imágenes del ser Humano:

El mundo exterior a los ojos se encuentra dividido en cinco zonas, cuatro de las cuales tienen una correspondencia directa con la estructura anatómica de los ojos y la quinta más bien parece tener una función integradora de la realidad.

De esto obtenemos como hipótesis que el concepto de Cuatrices tiene semejanza a la estructura funcional del sentido de la vista humana.

2 Redes Neuronales Fractales

Explicar las ideas que sustentan cada uno de estos conceptos, esta fuera del alcance del presente documento, sin embargo el uso de estos será únicamente desde un punto de vista intuitivo.

III Desarrollo de la Solución

Si dibujamos un objeto en una rejilla finita y definimos equivalencias de nivel, como se muestra en la figura 4 podemos realizar abstracciones continuas de la imagen original, hasta llegar a un solo elemento que contendría de manera sintetizada y global toda la información del objeto original.

Con la ayuda de la figura 4 podemos conceptualizar fácilmente que, el objeto original está disperso en la estructura de la pirámide que se forma al poner los diferentes niveles de abstracción, uno sobre otro. Entonces ya que toda la figura está dispuesta en un espacio de tres dimensiones, concluimos que para analizar imágenes en 2D necesitamos herramientas que actúen en 3D, resulta interesante pensar en el problema que se plantea al reconocer imágenes en 3D originalmente.

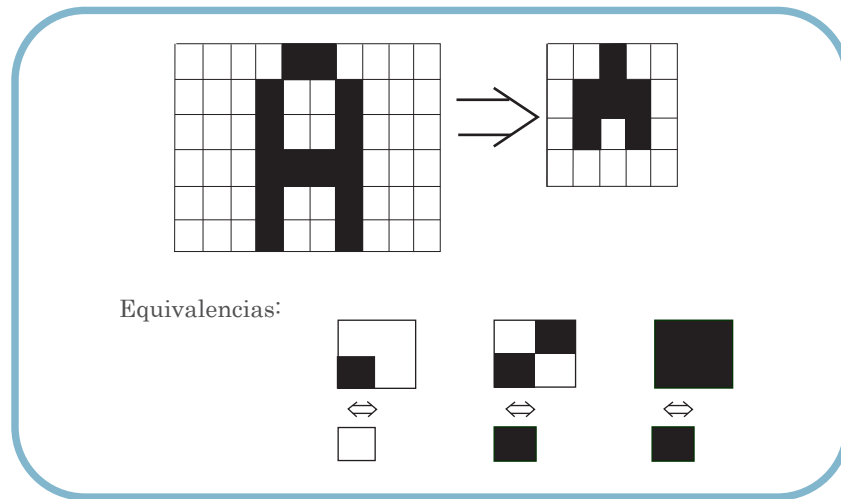


Figura 4. Una rejilla y equivalencias.

Regresando a la visión humana recordemos que ésta divide el espacio en cinco regiones, si intentamos aplicar el concepto de pirámide dentro del contexto de las cinco “regiones humanas”, tenemos como resultado una pirámide por cada región.

De lo anterior se concluye lo siguiente:

- 1) Se puede *asignar un procesador para el control de cada una de las pirámides de la periferia y otro para la pirámide central* actuando como integrador de las otras cuatro, así el tiempo de reconocimiento de un objeto se decrementaría.
- 2) Si se analizan las áreas que tienen en común las diferentes pirámides, podemos inferir que los puntos del objeto en esas regiones se encuentran definidos en dos lugares diferentes, lo que nos lleva a *redundancia en la información* y para poder quitarla sería necesario que los procesadores pudieran comunicarse entre sí, lo que implicaría que el modelo se estuviera moviendo con el propósito de automantenerse, y como el modelo es una imitación de la realidad posiblemente ésta también se estuviera moviendo en su estado original.
- 3) Es factible encontrar una solución en la que se obtengan *múltiples pirámides* controladas por varios procesadores, sin tener la limitante de cinco impuesta por la visión humana. Este enfoque requiere que el método de análisis de la imagen encuentre el número de procesadores dependiendo de la estructura interna del objeto de estudio.

Así, el Algoritmo Evolutivo para Tratamiento de Imágenes contempla las siguientes habilidades:

- 1) Utilizar como medio de análisis el *enfoque piramidal* sustentado en que para analizar imágenes en 2D debemos utilizar herramientas de 3D, de tal forma que encuentre una representación global del objeto original.
- 2) Debe encontrar la *firma característica* de la imagen que se le esta presentando, basándose en la estructura de ésta.
- 3) El proceso de *análisis debe ser evolutivo*, es decir, conforme avance en la elaboración de la síntesis de la imagen, vaya guiando su proceder con base a la experiencia obtenida hasta ese momento.

Dadas las características anteriores, comenzaremos con la descripción del planteamiento propuesto que contempla los requisitos de operación detallados.

Teniendo como referencia la cordillera que se forma con las pirámides por cada región de la vista humana, cada una de las pirámides puede contener a su vez una gran variedad de “pirámides”, como las llamaremos, de menor tamaño o grado. Tendremos pirámides de grado uno, dos, tres, etc., dependiendo de cuanta información contenga el objeto analizado.

También podemos plantear que teniendo la pirámide de tamaño 1 como fractal fundamental, podemos construir a partir de éste la pirámide total del objeto de estudio. Sin embargo describimos a continuación otro enfoque que puede ser más interesante.

Suponiendo que nos encontramos analizando una pantalla en la que se encuentra una letra A, acompañada de otros elementos que pudieran ser considerados como ruido, entonces dependiendo del grado de información a lo largo y ancho de la imagen podemos formar una especie de zona montañosa, que abarca toda la superficie original, donde los picos más altos revelan zonas con una alta densidad de información, logrando obtener por así decirlo, la montaña característica de la letra A, además del patrón del ruido que la rodea. Así de esta forma tendremos que no solo hemos analizado la A original, sino que además se ha captado de una manera global el significado de la escena donde se encuentra ésta.

El enfoque de solución empleado para alcanzar los objetivos propuestos, es tener como vía de solución una derivación de las redes neuronales.

Recordando los planteamientos descritos en antecedentes, las imágenes se trasladan a un espacio bidimensional sobre la cual se aplican los algoritmos que se deseen. El algoritmo de solución abarca el análisis de la imagen con varios colores, iniciando con la lectura de ésta y transportándola a una matriz bidimensional, en la cual cada pixel de la pantalla se encuentra en un nivel de abstracción cero (0) o inicial.

Se establece que el comportamiento de cada píxel en la matriz depende de sus vecinos (red neuronal) tanto en el mismo nivel como en los niveles superiores, llegando a ocupar una posición aleatoria dentro de la “montaña” general de la imagen. Después, si listamos todos aquellos picos que forman parte de las crestas de las montañas que fueron encontradas tendremos como resultado la ***“Firma Característica de toda la Imagen Analizada”***.

V.2 Sistema Evolutivo para Generar Figuras y Realizar Paisajes con Movimiento en 2D

Carlos Olicón Nava²

En este trabajo se tiene como objetivo general realizar un Sistema Evolutivo capaz de generar figuras en 2D y con ellas realizar paisajes y aplicarles movimientos mediante el uso del Lenguaje Natural del Usuario. En particular se busca que el sistema sea:

- 1) Capaz de interpretar la gramática entre los lenguajes Natural del Usuario y el Lenguaje de Gráficas (figuras) para satisfacer los requerimientos de realizar figuras y aplicarles movimiento.
- 2) Capaz de reestructurarse en base a los requerimientos del usuario y asocie nuevas figuras o nuevas palabras.
- 3) Sea amigable, es decir, fácil de manejar por el usuario, y este no tenga que aprender cuestiones de programación ya que el Sistema es el que debe de interpretarlo.

El usuario interactúa con el sistema mediante el uso del Lenguaje natural, con el cual él puede comunicarse fácilmente con la computadora, por ejemplo:

Dibuja un cuadro
Traza un triángulo
Pon el triángulo arriba del cuadrado
Haz que el cuadrado se mueva hacia la Izquierda

El Sistema interpreta las figuras que el usuario define como entrada y las asocia con el Lenguaje Natural del Usuario.

² Carlos Olicón Nava realizó este trabajo como parte de su tesis de Licenciatura en Ciencias de la Informática en IPN-UPIICSA.

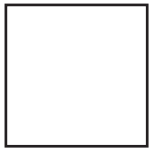
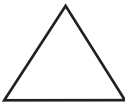
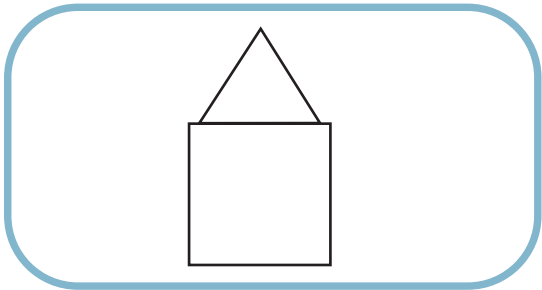
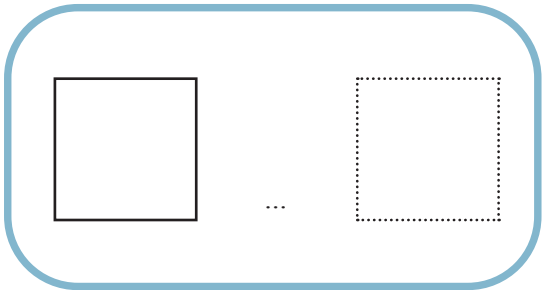
Imagen (figura)	Lenguaje Trayectorias	Lenguaje Natural
	3(a)3(b)3(c)3(d)	Cuadrado
	3(a)3(b)3(c)	Triángulo Equilátero

Figura 4. Una matriz evolutiva.

Pon el Triángulo arriba del Cuadrado



Mueve el cuadrado hacia la derecha



Para ello se aplican algoritmos que interpretan esta gramática para dar una mejor representación del Lenguaje Natural del usuario.

Inicialmente se realizó un estudio sobre el análisis y diseño del Sistema que describimos, que consiste en conocer la información que se maneja (datos de entrada y salida) en el sistema, así como las bases de datos asociadas (Diccionario de Datos, Estructuras Canónicas, Gramática del Lenguaje Natural, Gramática de las Figuras, etc.) definición de programas, pantallas, procesos, etc.

Se procedió a la elaboración del Sistema una vez realizado el Análisis y Diseño, determinándose el paquete de programación a usar para tal efecto se decidió usar un lenguaje de programación que permitiera la recursividad.

La programación se hizo en forma modular y se implantaron los siguientes módulos:

- 1) *Administrador Constructor (Lenguaje Natural y Figuras)*
- 2) *Constructor Léxico (Lenguaje Natural y Figuras)*
- 3) *Constructor Sintáctico (Lenguaje Natural y Figuras)*
- 4) *Constructor semántico (Lenguaje Natural y Figuras)*

Estos módulos funcionan integrados para obtener nuestro objetivo.

La Estructura del Sistema es como se describe a continuación:

El sistema está compuesto por un Administrador que indica al Sistema lo que debe de realizar. Así, la Interfase con el usuario se usa para comunicarse el usuario con el sistema mediante el uso de Lenguaje Natural restringido, para esto se utiliza un Intérprete.

También cuenta con un Constructor que asocia todo conocimiento nuevo e Indica al administrador si puede satisfacer una petición del usuario.

Para ello se le provee al Sistema de conocimiento en base a la Gramática a utilizar para el manejo del Lenguaje Natural, así como la de interpretación, de las Figuras o Gráficas, para que el Sistema tenga la capacidad de asociar figuras y poder reproducirlas cuando el usuario lo requiera, para esto usa la interface entre las Gramáticas.

Además cuenta con una Base de Conocimientos en donde se almacenan los programas que genera la Gramática de Figuras o Gráficas que el Usuario defina, así como de un Diccionario de Datos para determinar el Lenguaje Natural que Maneja el Sistema. Cabe aclarar que ambas bases de Datos tienen la capacidad de crecer, para un dominio mayor del Sistema en cuanto a lo que puede realizar el usuario.

Con esto podemos decir que el sistema puede Generar Figuras partiendo de otras para hacer Paisajes, además de definir en que trayectoria se desplazará (movimiento).

V.3 Sistema Evolutivo Graficador de Moléculas Orgánicas

Jesús Manuel Olivares Ceja³

Resumen

Existen diversas propuestas para modelar moléculas orgánicas, se presenta una alternativa en donde se obtiene la representación gráfica a partir del nombre de la molécula. Los nombres se procesan utilizando una gramática con atributos obtenida de los ejemplos dados al sistema en un proceso evolutivo, esto es, al ir recibiendo más ejemplos, la gramática junto con su semántica se reestructura en caso necesario.

Palabras clave: Sistemas Evolutivos, Gramáticas, Moléculas Orgánicas, Graficación

Introducción

Actualmente existen diferentes propuestas para graficar moléculas orgánicas, en algunos casos se especifican las coordenadas que conforman los átomos, en otros, se bosqueja la estructura mediante una interface gráfica. Dos ejemplos de entre decenas de ellos son: Geom [10], MOGRA [9].

La propuesta presentada es una extensión a [2], en la cual se obtiene la representación gráfica de las moléculas orgánicas mediante una gramática obtenida a partir de las reglas de la IUPAC [1]. En este artículo, el mismo sistema ayuda para obtener la gramática y asociarle su semántica de acuerdo con el enfoque de los Sistemas Evolutivos [3][7].

La modelación de moléculas es una herramienta útil para el análisis y diseño de experimentos, por ejemplo, en la industria farmacéutica cada día crecen las aplicaciones para diseñar o modificar medicamentos [9]. También es útil en cuestiones de simulación y cristalografía.

II El Lenguaje de los Nombres de Moléculas Orgánicas

La nomenclatura utilizada se basa en propuesta por la IUPAC [1]. Algunos nombres deben reexpresarse para explicitar las cadenas y subcadenas que lo componen para aplicar los mismos

³ Jesús Manuel Olivares Ceja realizó este trabajo dentro del curso de Autómatas Celulares en el Centro de Investigación y de Estudios Avanzados del IPN Sección de Computación del Departamento de Ingeniería Eléctrica en mayo 1994.

procesos lingüísticos descritos más adelante, esto es, en el caso de nombres comerciales o nomenclaturas distintas o especiales de la IUPAC.

Para esto se utiliza una tabla de reescritura (figura 1) en donde se tiene el nombre reexpresable y su asociado. Conforme se ingresan datos que corresponden a casos nuevos la tabla crece o se reduce al eliminar algunos obsoletos. La oración a procesar es la resultante de la reescritura.

Oración	Se reescribe como
ISOBUTANO	2-METILPROPANO
NEOPENTANO	2,2-DIMETILPROPANO

Figura 4. Una matriz evolutiva.

Cada nombre de molécula después de reescribirse, se divide en unidades léxicas, las que no se conocen se dan de alta. A cada unidad léxica se le asocia un tipo (categoría léxica). Al conjunto de tipos de una oración se le nombra oración canónica. Los tipos utilizados se muestran en la figura 2.

Tipo	Descripción
prefijo	MET, ET, PROP, etc.
a	ANO
e	ENO
l	IL
n	número
c	cuenta
,	guión
i	ignorar, se utiliza en unidades no significativas

Figura 2. Tabla de unidades léxicas.

Dado que las familias de nombres orgánicos convergen en una terminación común de acuerdo con cada una: alcanos (ano), alquenos (eno), alcoholes (ol), etc., se ve la conveniencia de tomar la oración canónica en sentido inverso (figura 3), de tal forma que las producciones inician por el identificador de cada familia.

Oración	Oración canónica invertida
2,3-DIMETILHEXANO	a hex l met c - n
PINTA EL 2 -METILPROPANO	a prop l met - n
PENTANO	a pent
DIBUJA UN HEXANO	a hex

Figura 3. Oraciones canónicas invertidas.

Se aplica distribución lingüística [5] en las oraciones para explicitar los atributos de subcadenas en una oración, como en el ejemplo del 2,3-dimetilhexano, el cual reexpresamos introduciendo paréntesis y los operadores algebraicos de unión (+) y concatenación (*) como sigue:

$$a \text{ hex } ((l \text{ met}) * (n1 + n2)) = a \text{ hex } (l \text{ met } n1 + l \text{ met } n2)$$

se realizan las operaciones indicadas y se eliminan paréntesis, resultando:

$$a \text{ hex } l \text{ met } n1 \text{ } l \text{ met } n2$$

lo que equivale a:

$$2 \text{ MET IL } 3 \text{ MET IL HEX ANO}$$

Con las oraciones canónicas invertidas y distribuidas se construye una gramática inicial de la que se muestran sus producciones:

$$\begin{aligned} S &\rightarrow a \text{ hex } l \text{ met } n \text{ } l \text{ met } n \\ S &\rightarrow a \text{ prop } l \text{ met } n \\ S &\rightarrow a \text{ pent} \\ S &\rightarrow a \text{ hex} \end{aligned}$$

En la obtención de la gramática, se debe detectar la recursividad [4] en producciones que tienen repeticiones internas para simplificarlas como en el caso de 2,3-dimetilhexano y similares en donde se tiene el factor de repetición “l met n”. En este caso se coloca un punto (.) como la convergencia de la repetición para evitar la cadena vacía.

La producción $S \rightarrow a \text{ hex } l \text{ met } n \text{ } l \text{ met } n$ al sustituirla por una recursiva se expresa como:

$$\begin{aligned} S &\rightarrow a \text{ hex } X \\ X &\rightarrow l \text{ met } n \text{ } X \mid . \end{aligned}$$

En la gramática se aplica la factorización para simplificar las repeticiones entre producciones (externas), ejemplo:

Sean las producciones:

$$\begin{aligned} S &\rightarrow a \text{ hex } X \\ S &\rightarrow a \text{ prop } X \\ S &\rightarrow a \text{ pent} \\ S &\rightarrow a \text{ hex} \\ X &\rightarrow l \text{ met } n \text{ } X \mid . \end{aligned}$$

al factorizar “a” resulta

$$\begin{aligned} S &\rightarrow a Y \\ Y &\rightarrow \text{hex } X \\ Y &\rightarrow \text{prop } X \\ Y &\rightarrow \text{pent} \\ Y &\rightarrow \text{hex} \\ X &\rightarrow l \text{ met } n \text{ } X \mid . \end{aligned}$$

En algunos casos se requiere que se aplique la sustitución de una producción por su símbolo no terminal que lo representa antes de aplicar alguna operación lingüística. Sean por ejemplo las producciones:

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow a X \\ X &\rightarrow \text{met} \\ X &\rightarrow \text{prop} \\ X &\rightarrow \text{dec} \\ B &\rightarrow l Y \\ Y &\rightarrow \text{et} \\ Y &\rightarrow \text{but} \end{aligned}$$

Si llega la oración canónica "a dec l et n l et n l et n" aplicando sustitución queda como $A B n B n B n$, en donde es posible aplicar la operación de recursividad, dando lugar a las producciones:

$$\begin{aligned} S &\rightarrow A Z \\ Z &\rightarrow B n Z \mid . \end{aligned}$$

con las mismas producciones A y B, porque sólo cambio S y se introdujo Z.

III Asociación de Semántica

A cada elemento de la gramática se le asocia un objeto gráfico que puede consistir de un átomo u otra molécula. Esto se indica delante del final de cada producción terminal en la gramática, dando lugar a lo que se nombra como gramática con atributos.

La semántica se expresa siguiendo la notación que consiste de alguna molécula o átomo existente y una lista de enlaces con otras moléculas que la complementan.

```
(MOLECULA_1 (átomo_1 enlace_1
MOLECULA_2 átomo_2 enlace_2)
...
(átomo_1 enlace_1 MOLECULA_n átomo_n enlace_n) )
```

en donde

MOLECULA_i: puede ser un átomo o una molécula;

átomo_i: es el número del átomo con que se enlaza a la MOLECULA_j. Su valor es relativo a la molécula a la que pertenece, si se une a otra molécula su valor puede cambiar enlace_i: es el número del enlace utilizado, siempre es menor ó igual a la valencia del átomo enlazante. El enlace puede ser sencillo, doble o triple. Su valor se conserva en cualquier molécula en la que se coloque.

Ejemplos:

Sea el caso del metano construido a partir de átomos, uno de carbono con hibridación sp³ y tres hidrógenos:

$S \rightarrow a \text{ met} : (\text{Csp}^3 (1 \ 1 \ \text{H} \ 1 \ 1) (1 \ 2 \ \text{H} \ 1 \ 1) (1 \ 3 \ \text{H} \ 1 \ 1))$

Sea el caso del 2-METILHEXANO reconocido por la gramática con producciones:

```
S → A B
A → a hex : (hexano)
B → l met : (metil 1 3)
```

su interpretación es la composición de las cadenas que la forman, como sigue:

$(\text{hexano} (2 \ 1 \ \text{metil} \ 1 \ 3)).$

el radical metil ya incluye el átomo y enlace disponible para unirse a una cadena principal, en este caso en el carbono 2 que es la posición donde se indicó que va el radical usando su primer enlace del carbono.

Los átomos constituyentes de las moléculas se describen mediante vectores en el espacio, descritos por sus coordenadas cartesianas (x, y, z) teniendo el origen (0, 0, 0) como el centro del átomo. Al concatenarse con otros se rota y su centro se traslada según se requiera. Cada enlace puede ser sencillo, doble ó triple; se unen únicamente con su correspondiente: sencillo-sencillo, doble-doble ó triple-triple.

La gramática con atributos obtenida se expresa formalmente como:

$$G = \langle N, T, P, S \rangle$$

donde

S es el símbolo inicial

T = {a, hex, prop, pent, met, n, ...} es el conjunto de terminales, identificados por iniciar con minúscula o ser signo de puntuación.

N = {S, X, Y, ...} es el conjunto de variables, identificados por iniciar con mayúscula.

P es el conjunto de producciones que incluye la interpretación en término de elementos gráficos.

como se ha visto el número de terminales, no terminales y producciones está sujeto a cambios constantes, conforme se dan de alta nombres de moléculas.

IV Sistema Evolutivo Graficador de Moléculas Orgánicas

El sistema se compone de los módulos siguientes:

- 1) Tabla de reescritura, para adecuar los nombres comerciales y aquellos aceptados por la IUPAC para simplificar otros.
- 2) Constructor Léxico, el cual le asocia su tipo a cada unidad léxica del nombre de la molécula. También da de alta aquellos que no se tengan registrados y formen parte del lenguaje. Esto permite que inicie su operación sin conocer las palabras del lenguaje. El resultado es la oración canónica invertida de la oración de entrada.
- 3) Constructor sintáctico-semántico, realiza el reconocimiento sintáctico del nombre de la molécula al tiempo que ubica su interpretación semántica. Si la oración canónica NO es reconocida y es válida de acuerdo con el usuario, se integra a la gramática aplicando alguna o varias de las operaciones lingüísticas: sustitución, distribución, factorización, recursividad. Si le hace falta algún componente para su interpretación se le solicita al usuario que lo proporcione, una vez integrada continua el proceso de interpretación.
- 4) Presentación visual, una vez reconocido e interpretado el nombre de la molécula, a sus componentes gráficos se le aplican operaciones de rotación y traslación para examinar la molécula.

V Obtención del Nombre a Partir de su Representación Gráfica

Dado el vínculo que se tuvo con el Departamento de Química del CINVESTAV-IPN, surgió el requerimiento de un sistema para obtener el nombre de una molécula dada su representación gráfica, en este sentido, Oscar Zarza Villavicencio, cursando el segundo semestre de la licenciatura en Informática en IPN-UPIICSA, realizó un sistema en donde la estructura de la

molécula se representa con un grafo. Sobre el grafo se encuentra el camino más largo del grupo funcional que caracteriza a la molécula, mismo que establece el nombre principal. Aplicando recursivamente la búsqueda de caminos más largos se encuentran las subcadenas. Concatenando los nombres y posiciones encontrados se obtiene el nombre de la molécula.

Conclusiones

Se han mostrado las características del lenguaje de las moléculas orgánicas junto con los módulos de un sistema para su validación gramatical y su interpretación gráfica. Es una alternativa en la modelación que las industrias requieren para analizar y proponer nuevas moléculas. Se mencionó también una alternativa para la generación automática de nombres de moléculas.

Bibliografía

- [1] IUPAC, International Union of Pure and Applied Chemistry, "Nomenclature of Organic Chemistry", Editorial Advisory Board H. W. Thompson, London, 1971.
- [2] J. M. Olivares C., H. V. McIntosh, "Graficación Tridimensional de Moléculas. Orgánicas a Partir de su Análisis Lingüístico" (por editarse), CINVESTAV-IPN, México, D.F., 1994.
- [3] F. Galindo S., "Sistemas Evolutivos: Nuevo Paradigma de la Informática" en Memorias de la XVII Conferencia Latinoamericana de Informática, Caracas, Venezuela, julio 1991.
- [4] E. Berruecos R., "Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos", IPN-UPHCSA, México, D. F., 1990.
- [5] J. M. Olivares C., "Sistema Evolutivo para Representación del Conocimiento", IPN-UPHCSA, México, D. F., 1991.
- [6] C. Olicón N., "Sistema Evolutivo Generador de Paisajes", IPN-UPHCSA, México, D. F., 1992.
- [7] F. Galindo S., "Sistemas Evolutivos" en Boletín de Política Informática, INEGI-SPP, México, D. F., septiembre 1986.
- [8] N. Chomsky, "Estructuras Sintácticas", 9na edición, introducción, notas, apéndice y traducción de C. P. Otero, Editorial Siglo XXI, México, 1987.
- [9] N. V. Ramana, I. Gosh, "Software report: Molecular graphics software MOGRA" en Computer & Graphics Vol. 17 No. 4, Great Britain, 1993.
- [10] H. V. McIntosh, "Geom for drawing spheres and things", Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias de la UAP, Puebla, Puebla, México, noviembre.

CAPÍTULO VI

SISTEMAS EVOLUTIVOS APLICADOS AL DESARROLLO DE SISTEMAS

VI.1 Generador de Sistemas como Núcleo de un Sistema Evolutivo

Fernando Galindo Soria¹

Introducción

En este trabajo se presenta el proceso para construir y usar una herramienta para el desarrollo industrial de sistemas de información.

Para lo cual, se considera que la arquitectura de un sistema de información típico (nómina, compilador, software educativo, sistema experto, etc.) se basa en tres grandes componentes: datos, procesos y estructura del sistema (figura 1).

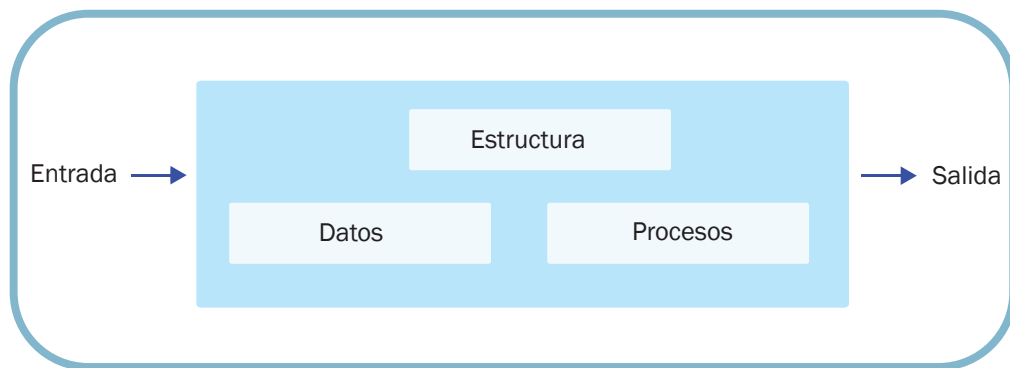


Figura 1. Componentes de un sistema de información.

Con la característica de que por lo común cuando cambia algún elemento de una de las componentes, las otras no necesariamente se alteran (si cambian los datos, no necesariamente tienen por qué cambiar los procesos o la estructura), por lo que, desde hace tiempo se ha tendido a desarrollar herramientas en las cuales sea relativamente fácil modificar una componente sin tocar las otras (por ejemplo, los sistemas manejadores de bases de datos), en particular aquí se presenta una herramienta de este tipo, conocida como Núcleo de un Sistema Evolutivo (figura 2).

Donde el Núcleo del Sistema Evolutivo es un programa de propósito general al cual se le dan como entradas los componentes de un sistema de información específico y es capaz de dar solución a los requerimientos, sobre este sistema particular.

¹ Fernando Galindo Soria IPN-UPHCSA enero de 1990.

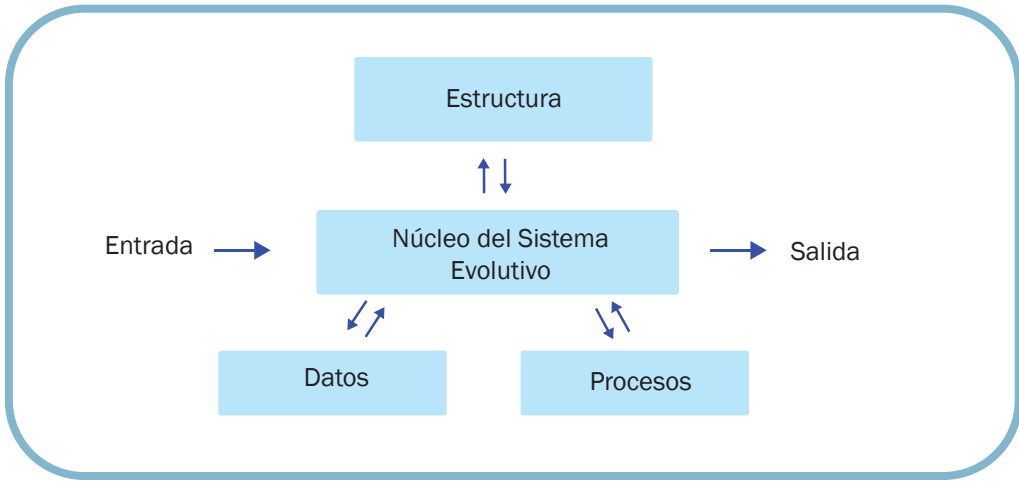


Figura 2. Componentes del Núcleo un Sistema Evolutivo.

Por ejemplo, si se tiene un sistema de información orientado al cálculo de estadísticas (figura 3).

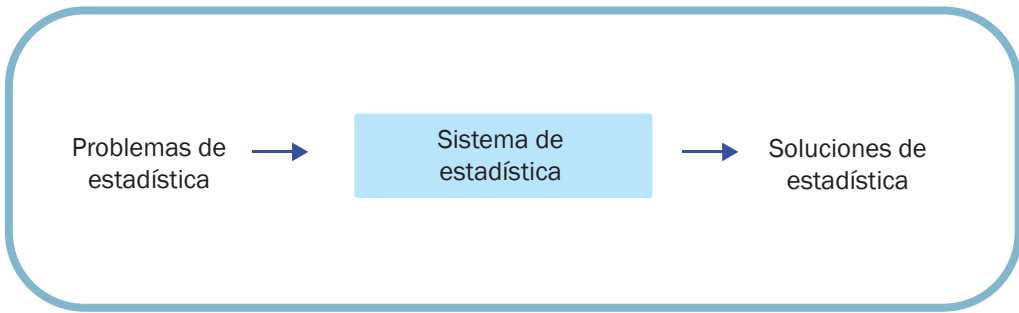


Figura 3. Cálculo de estadísticas.

La arquitectura general del sistema se muestra en la figura 4.

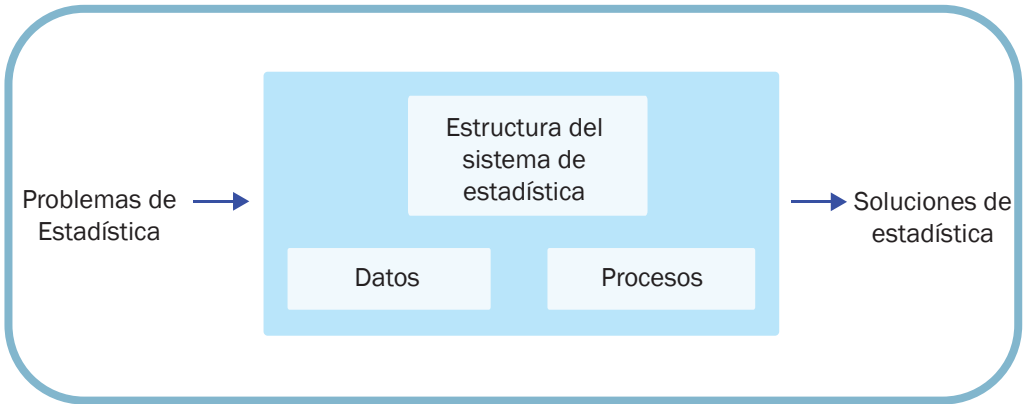


Figura 4. Arquitectura del sistema de estadísticas.

y en particular, al integrarlos como Núcleo del Sistema Evolutivo queda como se aprecia en la figura 5.

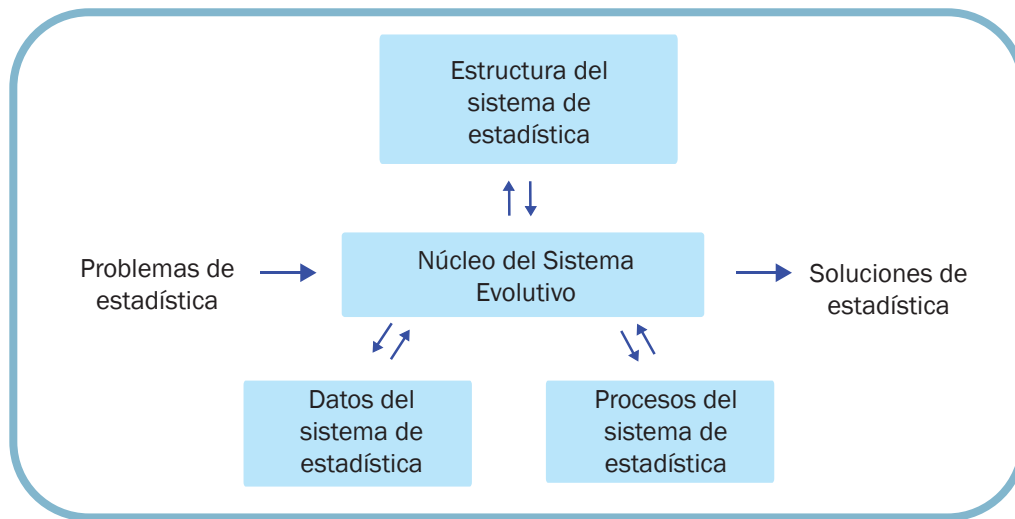


Figura 5. Sistema de estadísticas integrado al Núcleo del Sistema Evolutivo.

Las ventajas de un enfoque de este tipo se presenta por el hecho de que al tener el sistema de información distribuido en sus componentes, es relativamente fácil realizar modificaciones y actualizaciones sobre el sistema, ya que, si por ejemplo, cambia el orden en que se resuelve un problema, lo único que se tiene que cambiar es la estructura del sistema, sin tocar ni los datos ni los procesos.

Además y dado que para un área de aplicación específica (nomina, graficación, estadística) las rutinas involucradas son normalmente las mismas y sólo cambia el orden en que se aplican, es relativamente fácil generar una gran cantidad de sistemas concretos, ya que, al cambiar la estructura (orden en que se aplican los procesos sobre los datos) automáticamente se cambian los sistemas, por ejemplo, en el área de estadística existe un conjunto específico de rutinas con las cuales se cubren muchos de los requerimientos, por lo que, utilizando las mismas rutinas pero en distinto orden se pueden obtener resultados diferentes.

Por otro lado, una herramienta de este tipo es de propósito general ya que si se cambian las componentes de entrada, con el mismo núcleo se pueden obtener los resultados para un nuevo sistema, por lo que, sólo es necesario programar una sola vez el núcleo.

Los principales problemas que se presentan con este enfoque son:

- 1) ¿Cómo se encuentran y/o modifican las componentes de un sistema de información? (diseño lógico).
- 2) ¿Cómo se construye un Núcleo de Sistema Evolutivo? (diseño físico).

Por lo que, en lo que sigue se describen algunas de las soluciones a estos problemas.

I Diseño Lógico: Obtención de las Componentes de un Sistema de Información

Una primera dificultad que se presenta para desarrollar este enfoque se encuentra en el hecho de que en general se siguen desarrollando los sistemas basándose en la idea tradicional de Entrada-Proceso-Salida, figura 6.

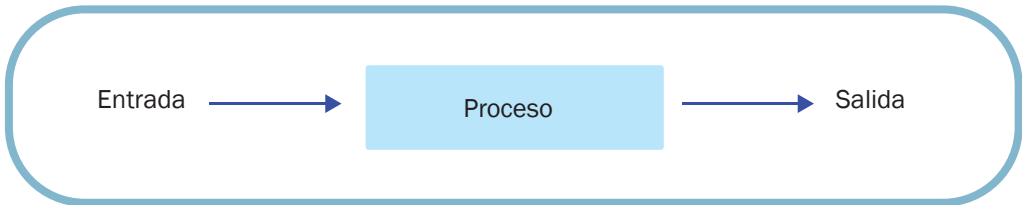


Figura 6. Modelo tradicional de los sistemas de información.

Por lo que cuando se desarrolla un sistema normalmente no nos preocupamos donde están los datos o cual es la estructura del sistema, por lo que, como primer paso se mostrará como detectar las diferentes componentes de un sistema, de tal manera que exista una independencia relativa entre los datos, procesos y estructura.

Por ejemplo, en la oración:

Dame la regresión de $X * 2 + Y$, Z

Es relativamente fácil detectar estas componentes (figura 7), donde la estructura está dada por las líneas que indican el orden en que se aplican los procesos sobre los datos, las acciones y los datos se indican mediante sus unidades léxicas respectivas (a, d).

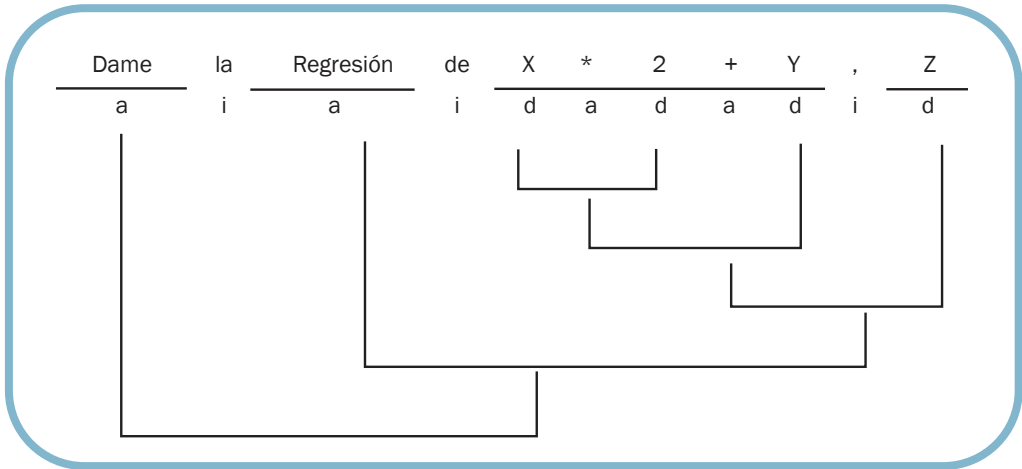


Figura 7. Componentes de un sistema en una oración.

En general ya que se acostumbra uno, es relativamente fácil encontrar las componentes de un sistema de información.

A continuación se muestra como dado cualquier sistema de información se pueden obtener sus componentes (procesos, datos y estructura) y en particular se describe como encontrar las componentes de un sistema descritos mediante Diagramas de Flujo de Datos (DFD) (una de las principales herramientas usadas para describir sistemas).

1 Obtención de los Componentes de un Sistema a Partir del Diagrama de Flujo de Datos

El DFD es una herramienta utilizada comúnmente para describir el flujo de datos dentro de un Sistema. Por ejemplo, en el DFD de la figura 8, el dato d_1 entra al proceso A que genera los datos d_2 y d_3 , los cuales entran respectivamente a los procesos B y C, y estos generan a su vez los datos d_4 y d_5 , quienes finalmente entran al proceso D el cual genera el dato d_6 .

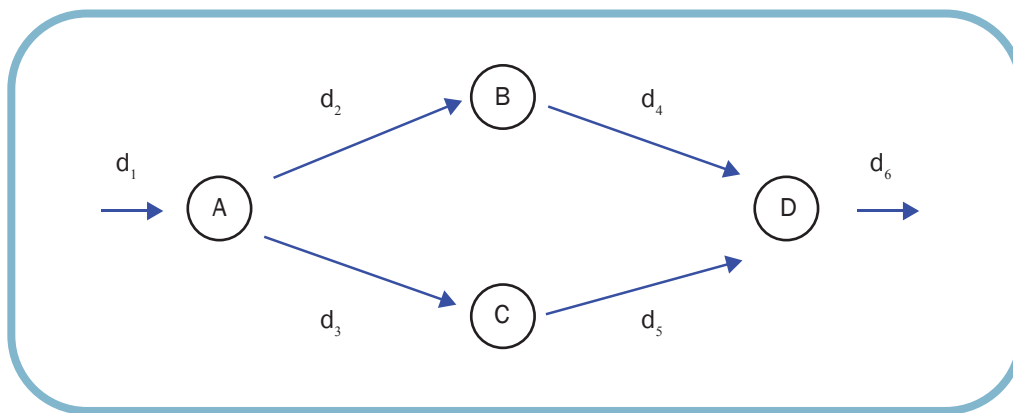
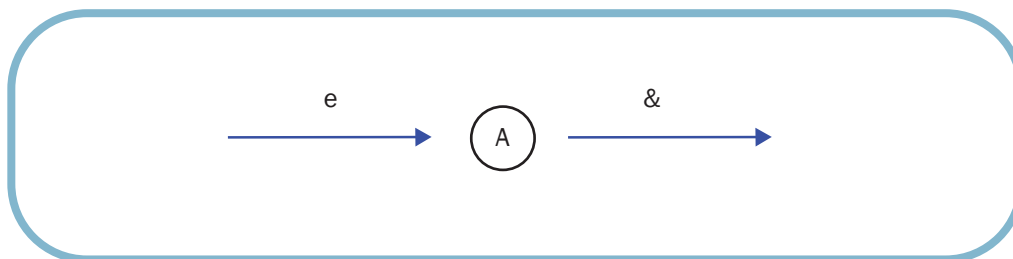


Figura 8. Un diagrama de flujo de datos.

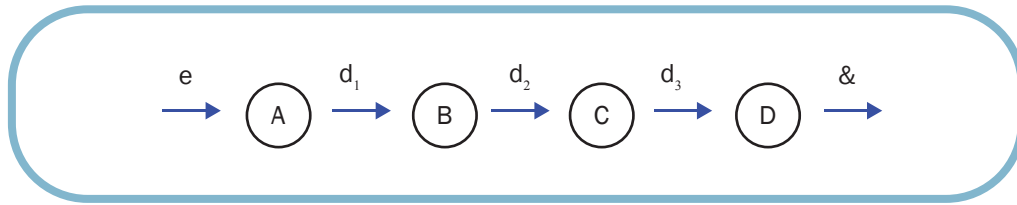
Dado un DFD es relativamente fácil encontrar sus componentes como se ve en los ejemplos que siguen:

1) Dado el DFD



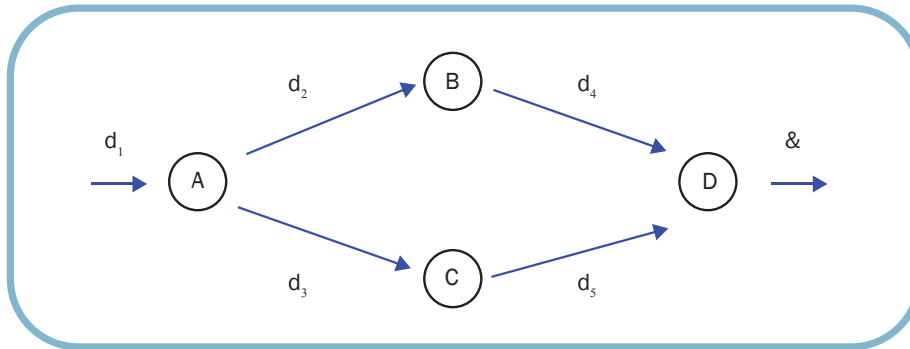
Tiene como datos (e, &), procesos (A), estructura ($S \rightarrow e \ A \ \&$). (El mecanismo para representar la estructura del sistema se puede interpretar como: el sistema S está formado por el dato e seguido del proceso A y terminando con el dato S).

2) Dado el DFD:



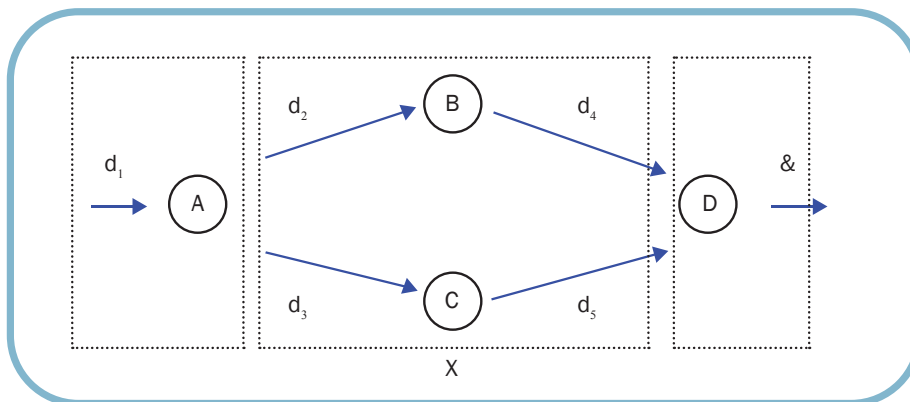
sus componentes son: datos (e, d1, d2, d3, S), procesos (A, B, C, D), estructura ($S \rightarrow e A B C D \&$). (es decir, el sistema S está formado por el dato e seguido de la ejecución de los procesos A, B, C, D y terminando con el dato S)

3) Dado el DFD:



En este caso el problema de la obtención de la estructura se complica ya que la secuencia no es lineal por lo que se aplica un método para transformar el diagrama (red) en un árbol (existen muchos métodos) y en particular se aplicará el método más sencillo, consistente en analizar la red de izquierda a derecha y de arriba abajo buscando formar “módulos” que engloben diferentes subproblemas.

Por ejemplo; en el problema anterior se puede analizar de izquierda a derecha quedando tres grandes submodulos:



de donde:

$$S \rightarrow e A X D \&$$

donde el submodulo X se puede ver como una rutina independiente (que se analiza de arriba-abajo)

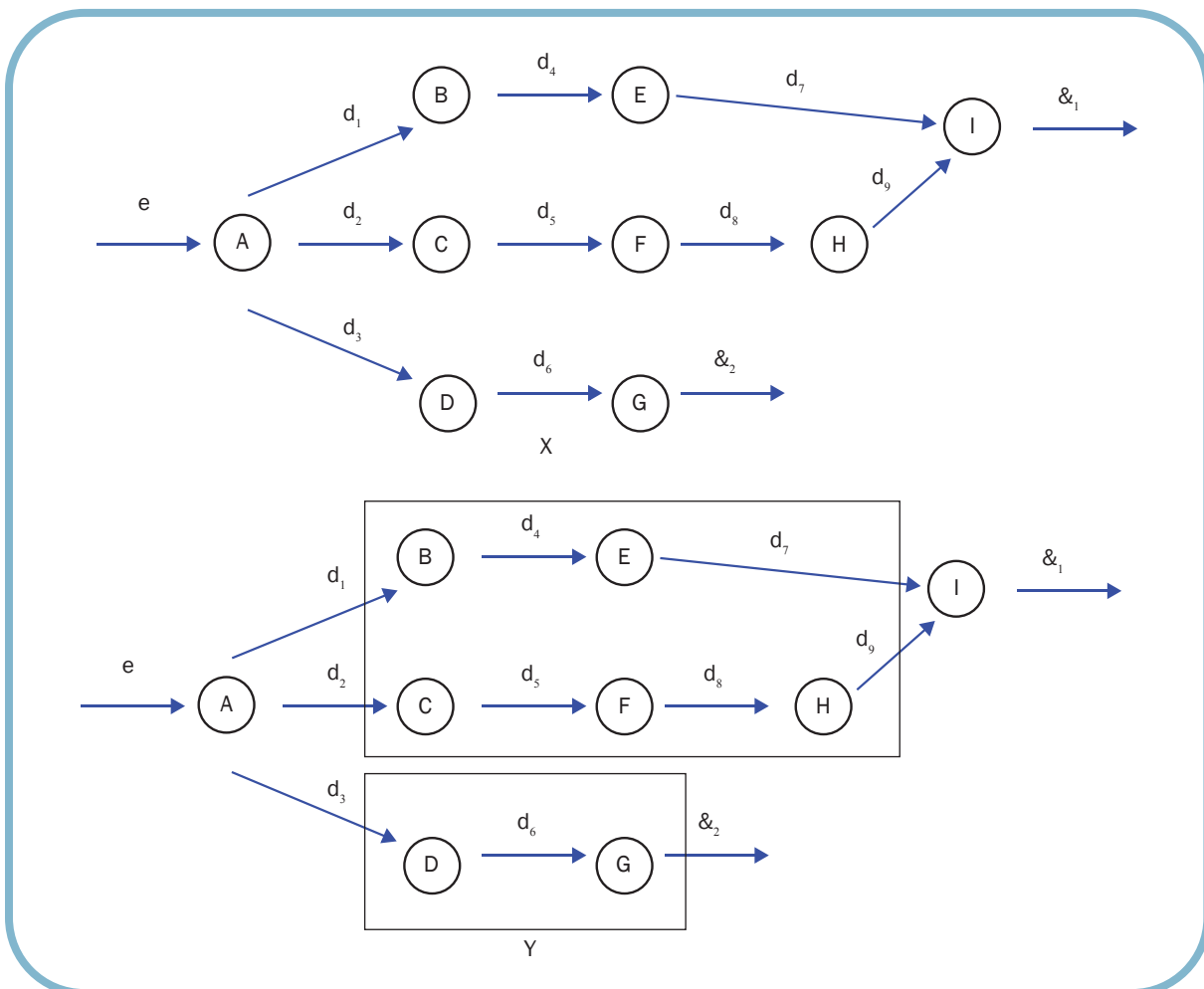
$$X \rightarrow B C$$

de donde la estructura del sistema es

$$S \rightarrow e A X D \&$$

$$X \rightarrow B C$$

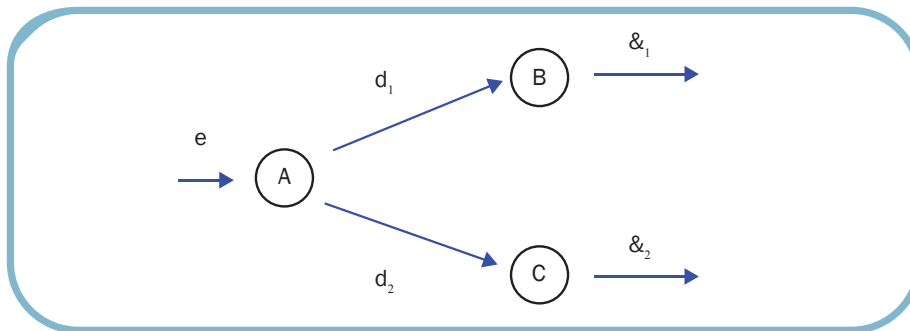
4) Dado el DFD:



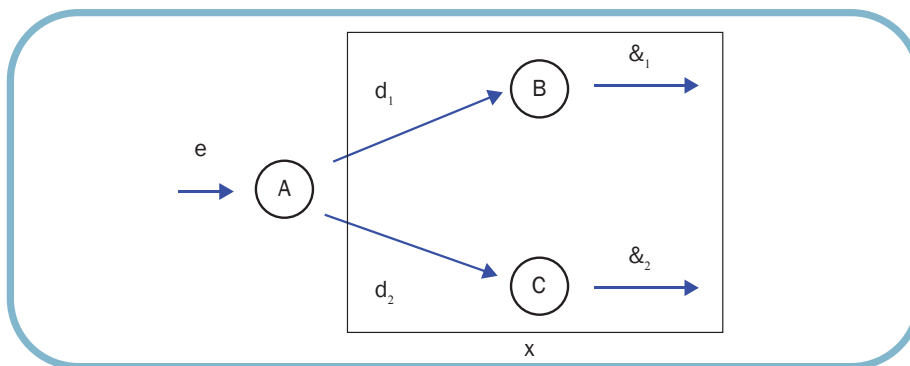
$$\begin{aligned}
 S &\rightarrow e \ A \ X \ \&_1 Y \ \&_2 \\
 &\Downarrow \\
 S &\rightarrow e \ A \ X \ \&_1 Y \ \&_2 \\
 X &\rightarrow X_1 \ X_2 \\
 Y &\rightarrow D \ G \\
 &\Downarrow \\
 S &\rightarrow e \ A \ X \ \&_1 Y \ \&_2 \\
 X &\rightarrow X_1 \ X_2 \\
 X_1 &\rightarrow B \ E \\
 X_2 &\rightarrow C \ F \ H \\
 Y &\rightarrow D \ G
 \end{aligned}$$

En los ejemplos anteriores se supuso que en el DFD se tienen que ejecutar todos los procesos y simplemente se transformó la red del DFD en un árbol, sin embargo, existen algunos casos especiales en los cuales se tiene más de un posible camino y se debe ejecutar uno u otro pero no los dos (como en el IF-THEN-ELSE de la programación estructurada).

Por ejemplo, en el DFD siguiente:



la disyunción puede no indicar procesos en paralelo sino que se ejecute el proceso B o C pero no los dos (esto depende del análisis del sistema). En este caso el método para obtener la estructura es parecido, por lo que el DFD anterior se divide como se muestra:



Que se representa como

$$S \rightarrow e A X$$

Donde en la rutina X se tiene que indicar que existen dos opciones quedando:

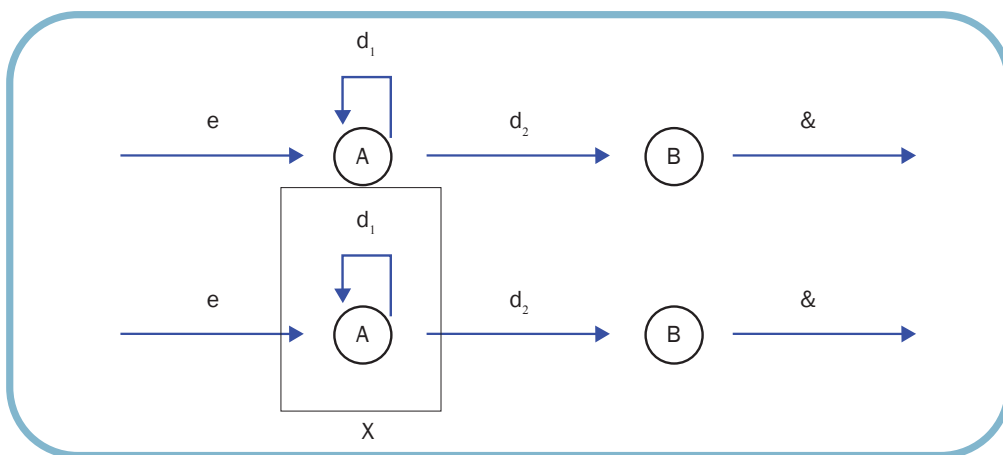
$$B \rightarrow B \&_1 \mid C \&_2$$

Donde la línea \mid indica ó, por lo que se lee, la rutina X ejecuta $B S_1$ ó $C S_2$ de donde la estructura completa es:

$$S \rightarrow e A X$$

$$X \rightarrow B \&_1 \mid C \&_2$$

Otro problema específico que se presenta es cuando se tienen ciclos dentro del DFD, por ejemplo:



En este caso se sustituye por una estructura de tipo recursivo de la forma:

$$S \rightarrow e X B \&$$

$$X \rightarrow A d_1 X \mid d_2$$

que se lee, el sistema S toma un dato e llama a X ejecuta a B y genera S, donde X ejecuta A, si continua el dato d_1 vuelve a llamar a X (proceso recursivo) ó termina el ciclo si llega el dato d_2 .

Como se puede ver, el problema de encontrar los componentes de un sistema de información a partir de un DFD, se reduce al problema de encontrar la estructura del sistema (ya que los datos y procesos se obtienen directamente del DFD) y este se reduce al problema de transformar una red a un árbol y en particular a la representación de las estructuras clásicas de la programación estructurada:

1) Secuencia:

$$S \rightarrow A B C D$$

2) Decisión:

$$S \rightarrow A \mid B$$

3) Repetición:

$$S \rightarrow A \mid S \mid B$$

En general mediante el método de diseño lógico descrito anteriormente, es relativamente fácil obtener las componentes de un sistema de información (figura 9).

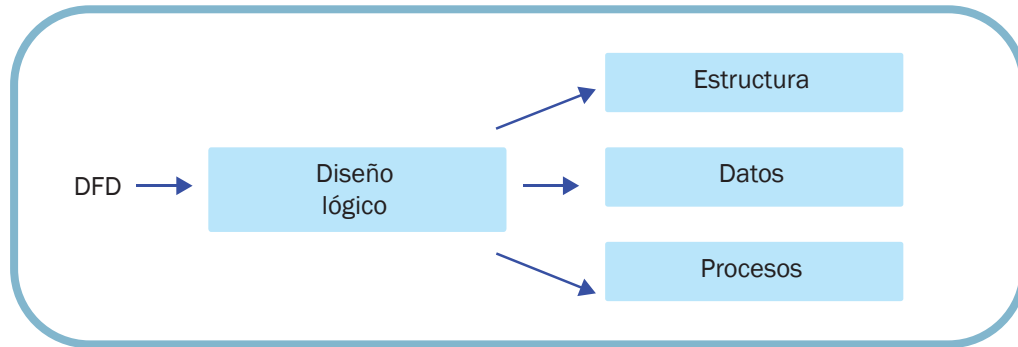
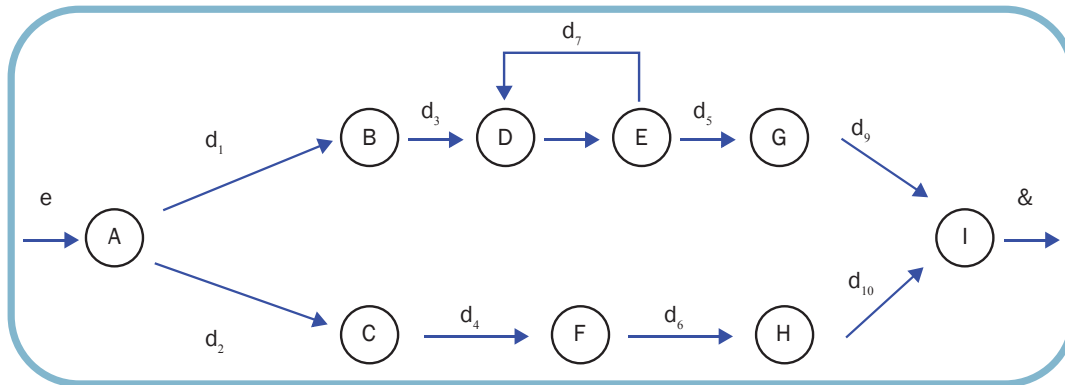


Figura 9. Diseño Lógico para obtener las componentes del sistema de información.

Aplicando lo anterior a otro ejemplo más complejo se tiene que dado el DFD:



se obtienen los componentes siguientes:

1) Datos. $e, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, S$

2) Procesos. A B C D E F G H I

3) Estructura.

$$S \rightarrow e \mid A \mid X \mid I \mid \&$$

$$X \rightarrow d_1 \mid X_1 \mid d_2 \mid X_2$$

$$X_1 \rightarrow B \mid X_3 \mid G$$

$$X_3 \rightarrow d_3 \mid B \mid e \mid X_3$$

$$S \rightarrow C \mid F \mid H$$

2 Diseño Físico: Construcción del Generador de Sistemas

Ahora nuestro siguiente problema es construir un programa X, al cual se le den como entrada los procesos, datos y estructura del sistema de información y sea capaz de responder a los requerimientos de este sistema de información. Programas de este tipo se han construido desde hace bastante tiempo pero con la característica de ser extremadamente complejos y difíciles de desarrollar. En particular a finales de los 70's y principios de los 80's dos trabajos de investigación independientes coordinados por Vicente López Trueba, el primero y por Lino Díaz Bello, el segundo, desarrollaron las bases de la herramienta que se va a presentar a continuación, en el primer grupo se construyeron un conjunto de generadores de programas, cada vez más completos y que en sus últimas versiones son de una simplicidad extrema, por su parte, Lino Díaz, desarrolló un método para construir compiladores basado en el manejo de gramáticas y en cual para obtener un nuevo compilador, lo único que se requiere es cambiar la gramática.

Esquemáticamente, estos dos modelos se pueden ver en las figuras 10 y 11.

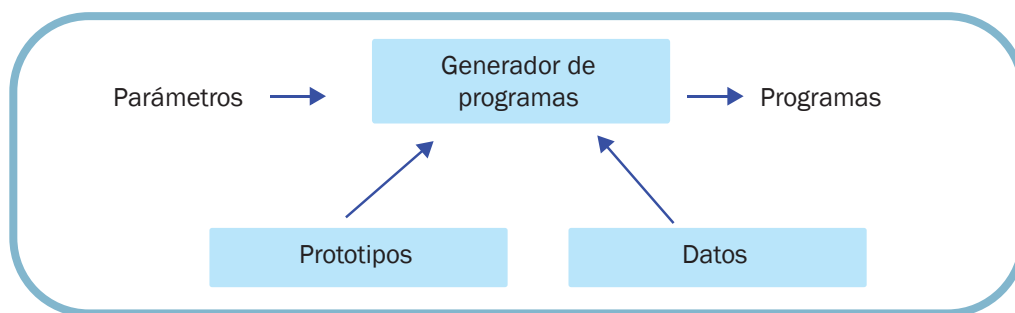


Figura 10. Un modelo de un Generador de programas.

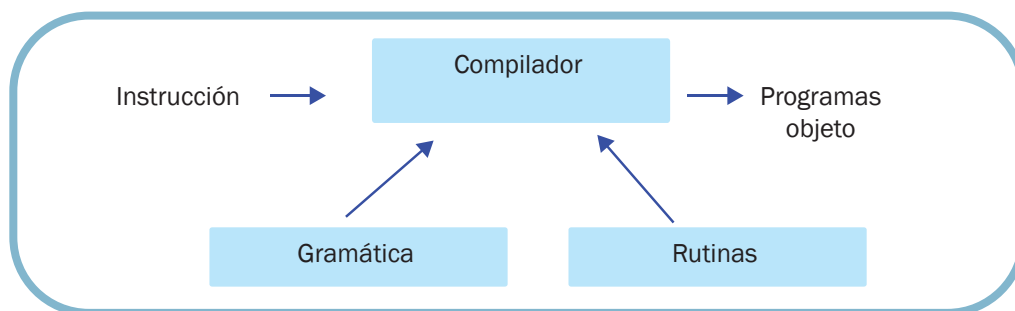


Figura 11. Un modelo de un Compilador Generalizado.

En el caso de la figura 10, los prototipos son programas parametrizados de tal manera que sirven como molde para generar un programa específico, y en el de la figura 11, la gramática es un mecanismo con el cual se representa la estructura de un lenguaje.

Integrando las dos herramientas anteriores y generalizando con el fin de representar la estructura de cualquier sistema, se obtiene la estructura de un generador de Sistemas (figura 12).

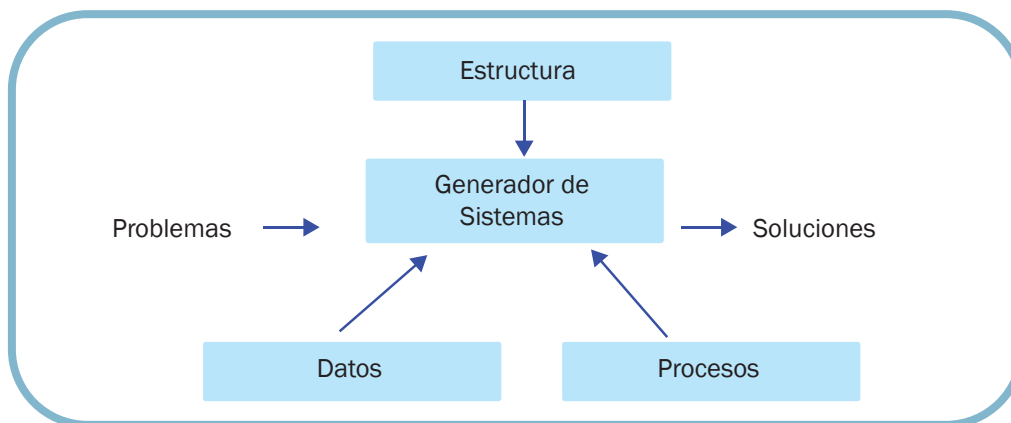


Figura 12. Estructura de un generador de sistemas.

La gramática es precisamente la estructura del sistema, por lo que nuestro problema es cómo lograr que la computadora integre un conjunto de procesos y datos de acuerdo a una estructura, con el fin de resolver un problema dado.

Cuando se analiza la estructura de un sistema o gramática se detecta que en ésta se encuentran involucrados los componentes, por lo que, lo único que se necesita construir es un mecanismo capaz de ir analizando los diferentes elementos de la gramática de acuerdo a los elementos de entrada y en su momento llame a los procesos y datos en el orden requerido.

Por ejemplo, si se quiere ejecutar una suma de variables (este es uno de los problemas más trillados) un sistema de este tipo tiene la arquitectura general mostrada en la figura 13.

Por otro lado, si se quiere ejecutar operaciones aritméticas en general lo único que se cambia es la gramática y el sistema debe de funcionar (figura 14).

Como se puede ver, el gran problema del generador es que debe ser capaz de ir siguiendo la gramática de acuerdo con las entradas y llamando a los procesos y los datos con el fin de obtener los resultados.

Por lo que el problema se centra en el análisis de la gramática.

Si se observa una gramática típica (como las obtenidas durante el diseño lógico) se encuentra que esta compuesta de tres tipos de elementos:

- 1) Datos o señales de entrada (también se les conoce como elementos terminales).
- 2) Llamados a Procesos (también se les conoce como rutinas semánticas)
- 3) Rutinas de control o llamados a otros componentes de la estructura (también se les conoce como elementos no terminales).

Por ejemplo, en la gramática:

$$\begin{aligned} S &\rightarrow v S_1 X \\ X &\rightarrow O S_2 S \mid ; S_3 \\ O &\rightarrow + \mid - \mid * \mid / \end{aligned}$$

se tiene lo siguiente:

- 1) Datos o señales de entrada:
 $v ; + - * /$
- 2) Llamados a procesos o Rutinas Semánticas:
 $S_1 S_2 S_3$
- 3) Rutinas de Control:
 $S X O$

Por lo que, el esquema general del generador de sistemas es el mostrado en la figura 15.

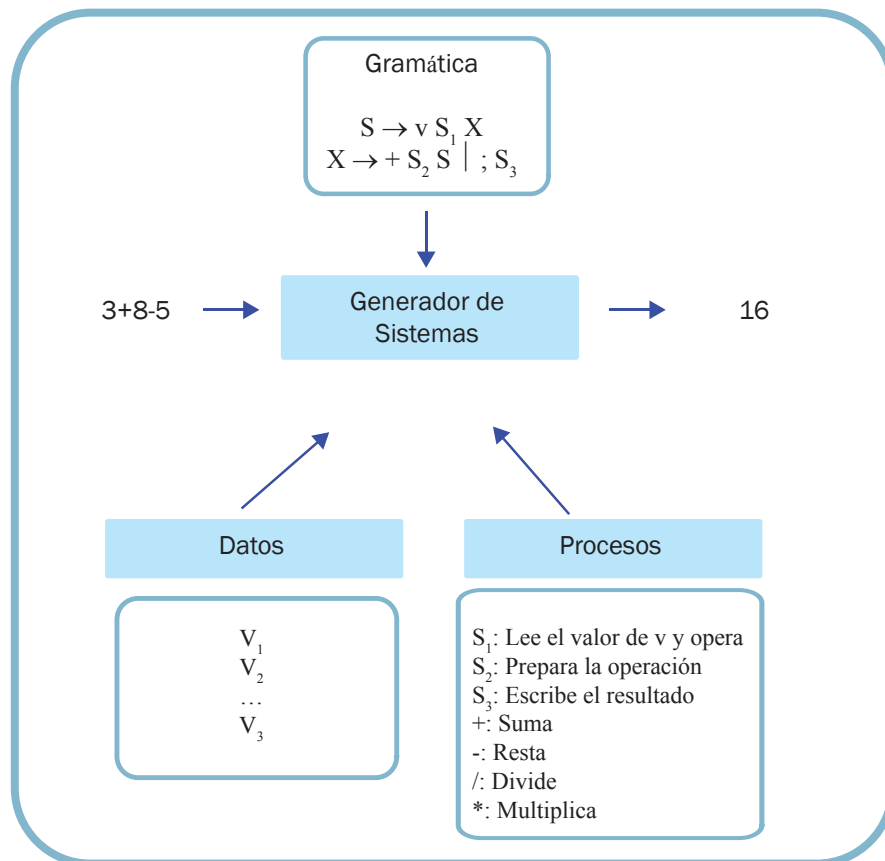


Figura 13. Una instancia del generador de sistemas de suma de variables.

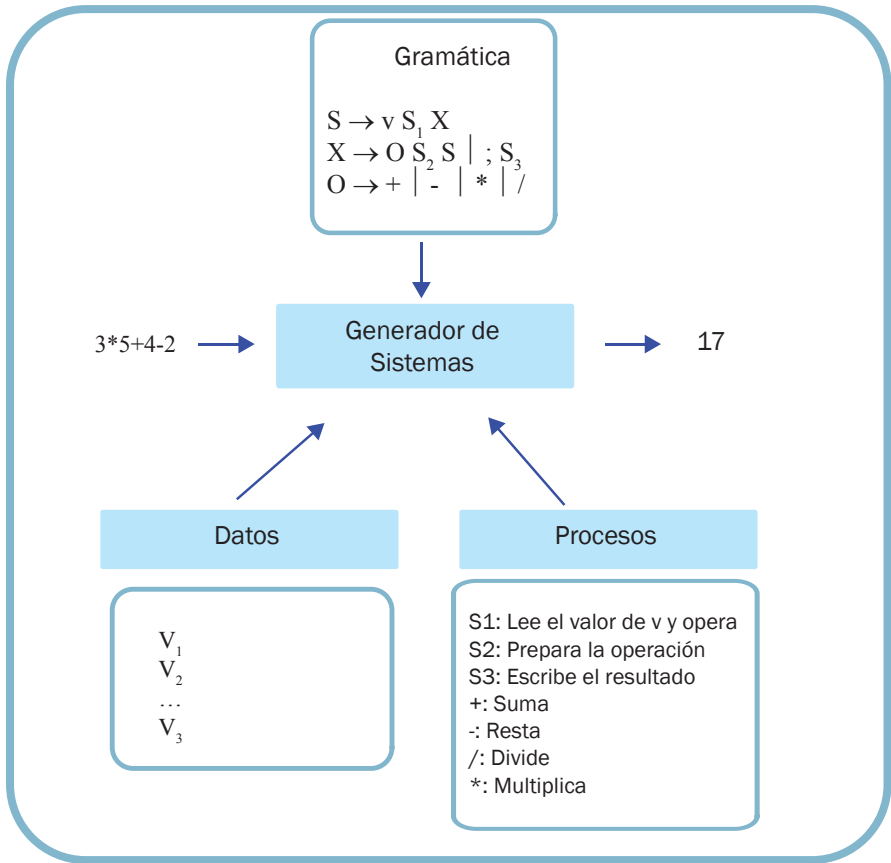


Figura 14. Una instancia del generador de sistemas extendido para varias operaciones.

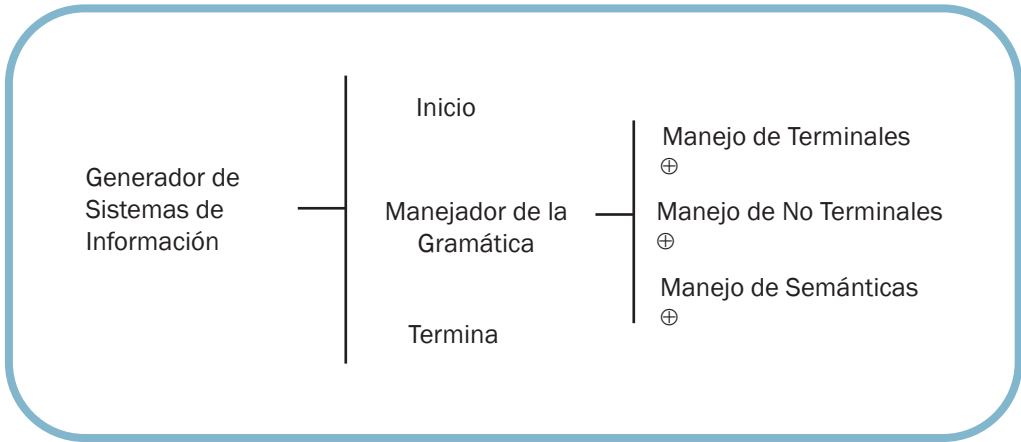


Figura 15. Esquema general del generador de sistemas.

Y el diseño detallado del generador de sistemas es el mostrado en la figura 16.

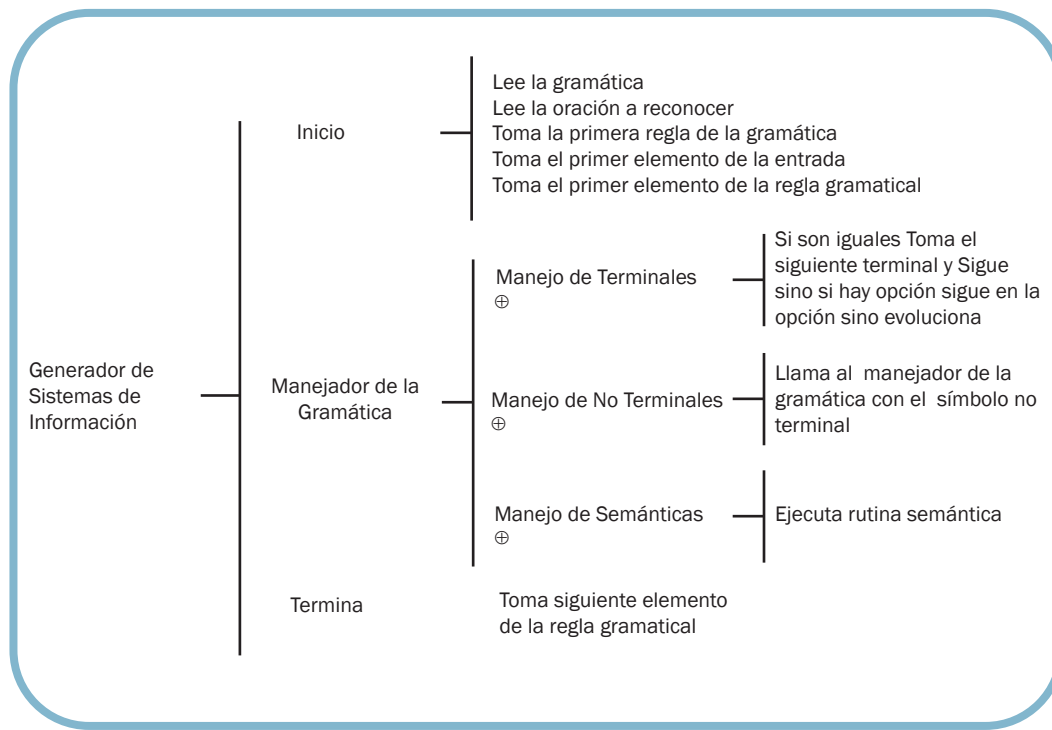


Figura 16. Esquema detallado del generador de sistemas.

Como se puede ver, es un programa pequeño (que normalmente se programa en una o dos hojas) pero sumamente poderoso ya que en la mayoría de los casos basta cambiar la gramática para tener una aplicación completamente diferente, sin necesidad de cambiar la lista de procesos en la mayoría de las áreas (por lo común en la mayoría de las áreas de aplicación ya establecidas se cuenta con una gran cantidad de rutinas y cuando “programamos” lo único que hacemos es “pegar” estas rutinas en distintos órdenes).

Lo interesante de este enfoque, es que si por algún motivo se cambia la estructura no es necesario volver a programar el sistema de información sino, únicamente se cambia el contenido del archivo donde aparece la estructura y el sistema sigue funcionando; lo mismo se puede decir si cambian los datos o los procesos; con lo cual, al tener separados los tres componentes y un mecanismo computacional que los integra es relativamente fácil desarrollar nuevos sistemas de información, ya que directamente del análisis se pueden encontrar estos componentes y almacenarse en tablas que va llamando el Generador de Sistemas.

II Diseño Físico: Núcleo de un Sistema Evolutivo

Finalmente se ve cómo se puede integrar esta herramienta como núcleo de un Sistema Evolutivo con el fin de que también se encuentren los componentes en forma automática a partir de una interacción directa con el ambiente o área problema.

Si se observa el esquema de la figura 17.

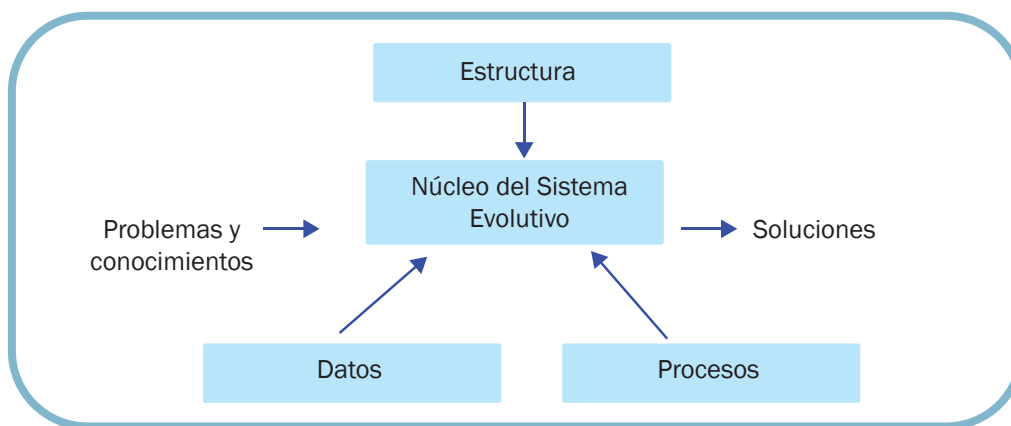


Figura 17. Núcleo del Sistema Evolutivo.

La diferencia principal entre el generador de sistemas y el núcleo de un Sistema Evolutivo es que en este último caso, existe una interacción en los dos sentidos entre el núcleo y las componentes del sistema de información, de tal manera, que si por algún motivo, cambia una rutina (proceso), el Sistema Evolutivo debe ser capaz de realizar el cambio en tiempo real y seguir funcionando, y lo mismo se puede decir si cambia la estructura del sistema de información o los datos.

Para lograr lo anterior, ya se cuenta con una gran cantidad de métodos y herramientas, pero en esencia la idea es que cuando el núcleo encuentra un elemento (dato, proceso) o alguna forma estructural que no reconoce en lugar de mandar un mensaje de error léxico o sintáctico, entre en un proceso de diálogo donde busca información que le permita actualizar las componentes o detectar que realmente se presentó un error.

Por ejemplo; si se está pidiendo ejecutar un proceso que no conoce, trataría de que directamente se le “enseñara” a resolver el nuevo problema (pidiendo por ejemplo el código en PASCAL o C del nuevo proceso), por otro lado, si lo que se encuentra es una estructura gramatical no conocida, verifica que sea una estructura valida y en ese caso modificaría la gramática con el fin de aceptar la nueva estructura (ya existen muchos métodos orientados a este fin, englobados en el área de Inferencia Gramatical).

Conclusiones

El área de los Sistemas Evolutivos es un área relativamente nueva, pero con un campo de acción creciente, sin embargo, aún es poco conocida, por lo que, en este trabajo se describió una de las herramientas que conforman a un sistema evolutivo, pero buscando su generalización sin necesidad de ser un experto en el tema, de tal manera que al contar con una herramienta de este tipo pueda desarrollar sistemas de información en forma industrial, y se motive a continuar investigando sobre este tema y propicie su desarrollo.

VI.2 EVA: Evolución Aplicada

Claudia Marina Vicario Solorzano²
Francisco Javier Aguilar Vallejo³

Resumen

Evolución Aplicada (EVA) constituye un nuevo enfoque en el proceso de análisis y desarrollo de sistemas, propone un acercamiento a la inteligencia de los mismos aprovechando los recursos ya existentes. En este proyecto se sugiere la creación de una máquina que permita a los sistemas integrarse, adaptarse y mejorar, permitiéndoles la continuidad; en otras palabras, que les de la posibilidad de evolucionar.

A esa máquina le hemos denominado máquina de evolución, la cual constituye una estructura que permite implementar Sistemas Evolutivos sumando a cualquier aplicación la capacidad de evolución.

Se presenta a continuación el diseño lógico de lo que se ha denominado (EVA), resultado de la investigación y estudio que sobre los Sistemas Evolutivos se realiza en la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) del Instituto Politécnico Nacional (IPN) México.

I Fundamentos Teóricos

Las áreas del conocimiento que constituyen las bases teóricas en que se cimenta el desarrollo de este proyecto son:

- 1) *Lenguajes y Gramáticas*. Con el empleo de gramáticas como medio de representación del conocimiento, la implementación del sistema resulta más eficiente además de ofrecer características especiales. El uso de gramáticas en nuestro proyecto es semejante al uso de los mismos en compiladores tradicionales. Es decir, nos permiten el reconocimiento de lenguajes en el proceso de comunicación con el usuario.

² Claudia Marina Vicario Solorzano,

³ Francisco Javier Aguilar Vallejo cursaban el séptimo semestre de UPIICSA cuando realizaron este trabajo en diciembre 1989. Eran propietarios desde entonces de la empresa TECCIZ de México.

2) *Teoría de la Información*. Estamos conscientes de que hay información en todo el universo, y que existen leyes que la rigen, lo único que hacemos al igual que en cualquier sistema, es transformarla pero siguiendo ciertos lineamientos que establecen orden y coherencia, permitiendo que los sistemas adquieran la capacidad de evolucionar.

3) *Sistemas*. Desde nuestro punto de vista un sistema está compuesto por tres elementos fundamentales (datos, procesos y estructura) siendo esta última el punto de acción del proyecto que aquí se propone ya que es posible obtener diversos resultados, con sólo modificarla sin que sea necesario alterar los datos y procedimientos (figura 1).

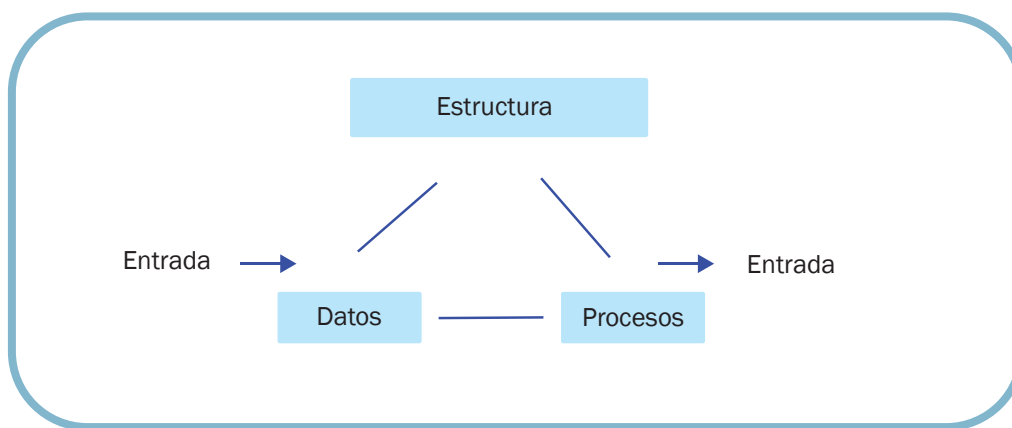


Figura 1. Componentes de un sistema.

4) *Procesos de aprendizaje*. Consideramos que existen dos formas básicas de aprender (memorización y conceptualización). Hasta hace poco tiempo la manera en que los sistemas pretendían adquirir conocimiento era sólo almacenando grandes cantidades de datos (memorización), sin embargo, el proyecto busca lograr el aprendizaje cognoscitivo (conceptual), mediante la implementación de estructuras gramaticales para representar conocimiento.

5) *Evolución*. Es un proceso de transformación controlada que tiene la finalidad de acercar un sistema al logro de sus objetivos, mejorando, adaptando, integrando o substituyendo los medios para alcanzar dichos objetivos.

II EVA: Aplicaciones Evolutivas

EVA es más que un proyecto de sistemas, constituye un enfoque en el proceso de análisis y desarrollo de los mismos, una filosofía en la que se conjuntan los aspectos más importantes para el desarrollo de nuevas tecnologías, ya que busca:

- 1) *Cubrir las necesidades actuales en materia de información.*
- 2) *Crear software comercial competitivo y novedoso.*
- 3) *Constituir un eslabón más en el camino a la creación y uso de aplicaciones inteligentes.*

La principal preocupación es la minimización de los costos de implantación y desarrollo. Propone un acercamiento a la inteligencia de los sistemas, aprovechando recursos ya existentes y avances en materia de Inteligencia Artificial (IA), es decir, en su objetivo no sólo está el crear nuevas aplicaciones con características pseudointeligentes, sino dar la posibilidad a las actuales de adquirirlas; tal y como lo hiciera, un “viejo” que renueva sus conocimientos y habilidades (en la medida de sus posibilidades, para “adaptarse a las condiciones actuales de su realidad”).

EVA sugiere crear una máquina que permite a los sistemas integrarse, adaptarse y mejorar, permitiéndoles la continuidad. En otras palabras, que les de la posibilidad de evolucionar. A esa máquina le hemos denominado máquina de evolución ya que es una máquina lógica que permite la evolución de las aplicaciones; por lo que las convierte en “aplicaciones evolutivas”.

Una aplicación evolutiva es como cualquier sistema evolutivo, un software capaz de aprender de su medio ambiente, es decir, que adquiere conocimiento y lo utiliza describiendo un comportamiento que asemeja al humano. Un sistema evolutivo permite la comunicación con el usuario en su propio lenguaje y la ejecución de instrucciones de manera automática en relación a la orden recibida.

Para que un sistema evolutivo adquiriera conocimiento es necesario que aprenda dos cosas:

- 1) *El vocabulario.*
- 2) *El significado* de cada una de las palabra del vocabulario, ya que esta última parte es la que le permite comprender la orden y en su caso ejecutarla.

Un sistema evolutivo ideal es aquel que aprende de manera automática ambas cosas, sin embargo, hasta el momento ha sido muy difícil partir de “cero conocimientos” y lograr que un sistema adquiriera la información sobre el cómo hacer las cosas (rutinas) sin la intervención de un técnico o en su caso sin que sea sólo para alguna aplicación particular.

Dada esta situación, nuestra propuesta busca obtener aplicaciones evolutivas que aprendan del usuario el vocabulario y del técnico las rutinas correspondientes, aunque por el momento de manera manual, en este último caso; considerando para posteriores versiones la posibilidad de implementar un método general que permita la automatización del mismo (figura 2).

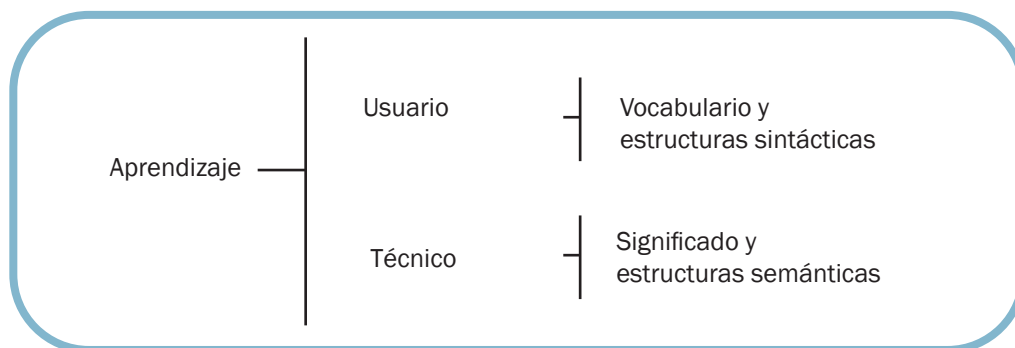


Figura 2. Fuentes de adquisición del conocimiento.

Tradicionalmente se venían creando aplicaciones evolutivas de propósito particular, sin embargo, observando que ellas contenían estructuras internas muy similares que les da capacidad evolutiva, consideramos que cualquier sistema puede convertirse en un sistema evolutivo si se le dan de alguna manera la capacidades para evolucionar y pensamos en la posibilidad de crear una estructura general que “inyecta” a cualquier aplicación dicha capacidad.

Una máquina de evolución (ME) no es un generador de sistemas evolutivos ya que no trabaja como un sistema al que se le proporcionan datos específicos y en base a un molde genera automáticamente sistemas con las características de ese molde; una ME es más bien, una estructura que permite “implantar sistemas evolutivos, sumando a cualquier aplicación la capacidad de Evolución”:

$$\text{APLICACIÓN} + \text{CAPACIDAD DE EVOLUCIÓN} = \\ \text{Aplicación evolutiva}$$

Se espera que al usar una máquina con estas características el usuario logre integrar aplicaciones nuevas y existentes; obteniendo así un sistema diferente que cada vez será más poderoso ya que no sólo tendrá la posibilidad de comunicarse en cualquier lenguaje, sino que integrará las características de numerosas aplicaciones e innovaciones y que aprenderá nuevas actividades como si tuviera la inteligencia que permite a un niño aprender a sumar, después a multiplicar y así cada vez a realizar operaciones más complejas.

La propuesta es pues “generalizar la creación y uso de aplicaciones evolutivas”, partiendo de la premisa de que la tendencia es que el conocimiento se convierta en el recurso más importante de las organizaciones y de la vida futura en general como una solución a la problemática de administración del mismo.

III Elementos Estructurales de una Aplicación Evolutiva

A continuación se describen los elementos que conforman a una máquina de evolución (véase la figura 3) la cual es una estructura, que establece una adecuada interrelación entre los elementos que componen a un sistema tradicional y los módulos de aprendizaje, comunicación y control.

1 Elementos Estáticos

Son los que aportan las características de evolución y aprendizaje, se consideran estáticos por que se mantienen sin cambio en su estructura durante el proceso de ejecución, ellos, realizan las tareas de establecer un medio común de comunicación con el usuario, controlar los procesos de aprendizaje, realizar las operaciones necesarias para ejecutar las acciones solicitadas por el usuario y principalmente coordinar la interacción de todos los procesos.

El conjunto de módulos de comunicación, aprendizaje y control, constituyen los elementos estáticos (figura 4), dichos módulos son de carácter general, es decir, que fueron construidos con las características necesarias para que, en conjunto, sean considerados como un shell (cascarón estructural) el cual ha de ser llenado con la información específica que en principio se tome de alguna aplicación determinada.

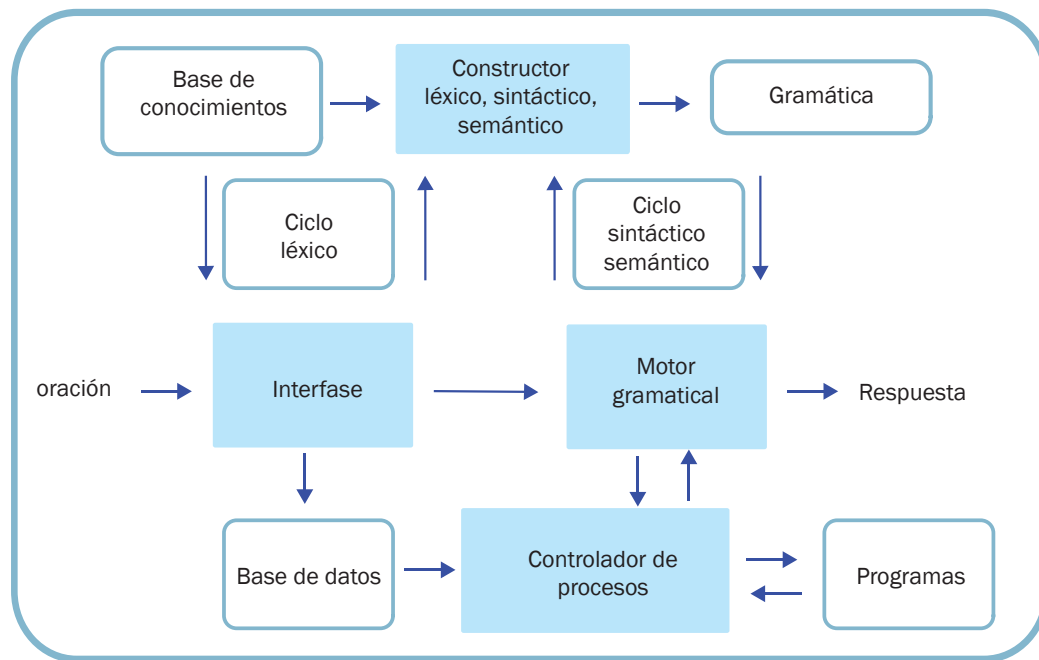


Figura 3. Diagrama estructural de la máquina de evolución.

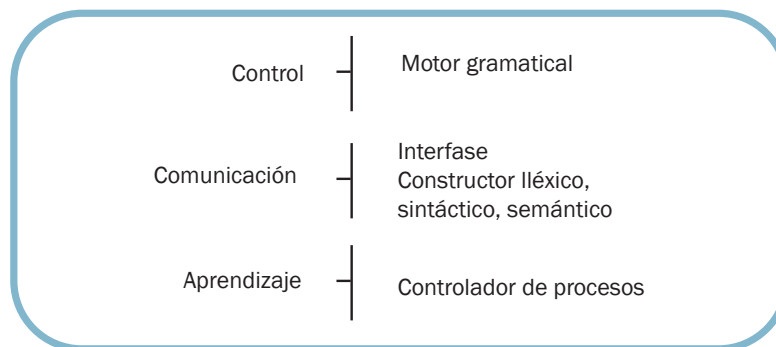


Figura 4. Módulos de los elementos estáticos.

1) *El Motor Gramatical*. Coordina y controla las acciones de las operaciones que se realizan entre los módulos, determina el momento en que ha de ejecutarse un proceso e indica el éxito o fracaso del intento de ejecución.

El motor gramatical, durante la sesión de trabajo, establece un ciclo iterativo de procesos que comienza con la puesta en marcha del módulo interfase, este entrega como resultado de su intervención una cadena canónica que se toma como oración de arranque para el motor, el análisis sintáctico de dicha oración determina el curso de acción a seguir, ya sea la ejecución de una rutina o la llamada a escena del módulo de aprendizaje para registrar un conocimiento nuevo, el motor le indica constantemente el estado actual en que se encuentra el sistema y la ejecución de sus tareas con el fin de ofrecer un monitoreo del éxito o fracaso de alguna operación.

Tiene a su cargo diferentes funciones, las más importantes son: análisis sintáctico, análisis semántico, adquisición de conocimiento y ejecución de acciones.

- i) *El análisis sintáctico*. Verifica la validez sintáctica de la cadena canónica de entrada, durante dicho proceso se efectúa consecuentemente la ejecución de las operaciones indicadas en las rutinas semánticas asociadas en la gramática con atributos. Este módulo indica solo el arranque de los procesos. Si la cadena canónica no se reconoce por la gramática, se inicia un proceso de corrección aprendizaje.
- ii) *El análisis semántico*. Durante este se incorporan las rutinas semánticas a la gramática que son las que determinan el significado de las oraciones canónicas y ejecutan las acciones resultantes de la interpretación semántica.
- iii) *Adquisición de conocimiento*. Esta función se invoca cuando el motor gramatical detecta un conocimiento nuevo (que no se tiene) por lo que el sistema pasa el control al módulo de construcción LSS.
- iv) *Ejecución de acciones*. Durante el análisis sintáctico de la canónica de entrada, se realiza también un proceso de identificación y ejecución de rutinas semánticas (las cuales se encuentran de manera explícita y en forma de llamadas en la gramática) cuando se detecta una llamada a alguna rutina semántica, el motor gramatical indica al módulo controlador de procesos cual es la rutina que ha de ejecutar.
- v) *Coordinación y control de módulos*. Las acciones que realiza la máquina de evolución están de alguna manera controladas por el motor gramatical el cual debe su comportamiento principalmente al seguimiento de las estructuras gramaticales que representan a sus conocimientos.

2) *La Interfase*. Permite establecer un medio de comunicación entre entidades hombre-máquina, sistema-sistema, dispositivo analógico-aplicación.

La comunicación se lleva a cabo considerando el lenguaje natural del usuario (o de la entidad que se trate).

El módulo interfase recibe una oración del usuario que es sometida a un análisis léxico con el objeto de identificar las unidades léxicas que la componen, en ese proceso se verifica la existencia de dichas unidades en la Base de Conocimientos, si por el contrario, alguna no existe, en ese momento se inicia un proceso de aprendizaje controlado por el módulo constructor, con lo cual se completa el ciclo léxico del sistema. Si la oración se reconoce satisfactoriamente, se genera la oración canónica.

El proceso de comunicación que controla el módulo Interfase, puede realizarse no sólo entre hombre y máquina, sino, puede ser entre máquina-máquina o contando con la instalación técnica adecuada también puede reconocer e interpretar señales eléctricas vistas como “oraciones de entrada”.

Para cumplir con el objetivo de este módulo, se realiza una serie de funciones secundarias que pueden englobarse en tres principales.

i) *Identificación de unidades léxicas.* Las oraciones de entrada están compuestas de elementos gramaticales llamados unidades léxicas, las cuales están definidas por patrones, la cadena oración se analiza carácter por carácter hasta encontrar su relación con algún patrón, de esta manera conforme se reconocen cada una de las componentes de la oración se construye la oración canónica.

ii) *Reconocimiento y adquisición de patrones léxicos nuevos.* Los elementos en que se puede descomponer una oración se diferencian entre si por el tipo al que pertenecen o por la forma que tienen, cada tipo está relacionado con un patrón el cual define como debe estar constituida una unidad léxico para que pertenezca al conjunto de cadenas generadas por dicho patrón.

Cuando se tiene una unidad léxica que no se relaciona con ningún patrón conocido, seguramente se trata de un error en la escritura de la oración, pero pudiera ser que fuera un tipo de dato nuevo, en ese caso el módulo interfase debe identificar el “tipo” de dato que se trata para crear un patrón nuevo y entonces agregarlo al conjunto de patrones ya conocidos.

iii) *Construcción de canónicas.* Una vez que se han identificado las unidades léxicas de la oración, se construye una cadena adicional, la cual corresponde a la oración canónica.

3) *Constructor Léxico, Sintáctico, Semántico.* Es el encargado de controlar los procesos de aprendizaje. En el momento en que el sistema se encuentre con información no conocida, es este módulo quien toma el control y efectuará una serie de operaciones con el fin de establecer si se trata de un error en los datos o si efectivamente es un conocimiento nuevo.

El proceso al que se somete la oración depende del origen del problema, ya que puede ser léxico o sintáctico. Cuando es de carácter léxico se efectúan operaciones de actualización en la base de conocimientos, por otro lado, cuando se trata de sintaxis, se obtiene el significado de cada elemento de la oración, para generar la cadena canónica, la cual se somete a procesos de factorización y recursividad (operaciones de inferencia gramatical) para obtener la gramática generativa que sustituye a la anterior. Con esto, en una segunda pasada, la oración que se examina ahora sí se reconoce y se ejecuta el significado resultante.

Las funciones más importantes de éste módulo son:

i) *Validación de información.* Los componentes de la oración de entrada deben relacionarse con la información que existe en la base de conocimientos. El propósito de esta función es verificar la congruencia de la información que entra en relación con la ya existente.

ii) *Obtención de significados.* En el momento que se descartó la posibilidad de error de

escritura en la oración de entrada, el sistema asume que se trata de un conocimiento nuevo, el cual puede ser una palabra nueva, un sinónimo o una estructura no contemplada en la gramática. Enseguida se dispone a investigar cual es el significado de esa oración o palabra. Para lograrlo se inicia una sesión de preguntas al usuario técnico respecto a dicho significado. En las respuestas se pueden emplear conceptos conocidos o definir nuevos con lo que se construye una red semántica que relaciona palabras con significados.

iii) *Reestructuración del conocimiento*. La adquisición de conocimiento nuevo puede afectar directamente a la estructura de significados existente hasta ese momento, de tal manera que se efectúe una reestructuración completa de las relaciones en la base de conocimientos, con lo que se amplía la capacidad de reconocimiento de oraciones y en algunos casos, la asociación de significados se modifica en gran medida.

4) *Controlador de procesos*. Realiza las operaciones necesarias para la ejecución de los procesos que están relacionados con una acción específica solicitada por el motor gramatical.

El controlador trata de proporcionar los recursos de información necesarios para la ejecución de las rutinas semánticas. Al finalizar la ejecución de los procesos se genera un estado de resultado en el que se registra el éxito o fracaso de una operación.

Las funciones principales de éste módulo son:

i) *Identificación de procesos*. Las llamadas que efectúa el motor gramatical durante el análisis sintáctico a las rutinas semánticas son detectadas por el controlador con el fin de identificar cuales son los procesos o rutinas que deben ejecutarse.

ii) *Preparar el ambiente de ejecución*. Los procesos que previamente fueron seleccionados, requieren de cierta información para la realización de sus tareas, esta información está determinada por las condiciones establecidas por el motor gramatical. Por lo tanto, es necesario contar con los recursos de información que cada rutina necesita.

iii) *Ejecución de procesos*. Como fase final, el controlador supervisará la ejecución de los procesos para asegurar su aplicación correcta.

2 Elementos Dinámicos

Los elementos dinámicos se mantienen en constante cambio y transformación durante la ejecución del sistema, ya que son el resultado final de los procesos de aprendizaje, estos constituyen el conocimiento adquirido por el sistema y por lo tanto revisten gran importancia, estos elementos son:

- 1) *Gramática con atributos*.
- 2) *Base de conocimientos*.
- 3) *Base de datos*.
- 4) *Procesos (rutinas semánticas)*.

Conclusiones

Es evidente que el manejo del conocimiento tiende a convertirse en la temática más importante a considerar en el desarrollo de sistemas.

Creemos que la búsqueda de técnicas y mecanismos que faciliten el manejo del mismo debe constituir la tarea a la que deben dedicarse varios investigadores en este momento.

Las aplicaciones evolutivas son, en este sentido, una propuesta de solución a este problema. Sin embargo, el desarrollo de nuevas teorías sobre sistemas inteligentes no es suficiente. La situación actual exige soluciones tangibles y en tiempo real. Por lo que se hace necesaria la participación de investigadores de diferentes áreas trabajando conjuntamente para ver cristalizados los resultados que se esperan.

VI.3 Sistema Evolutivo Generador de Aplicaciones

Jesús Manuel Olivares Ceja⁴

Resumen

Se presenta en este trabajo un Sistema Evolutivo para generar y mantener aplicaciones informáticas. Se parte de la taxonomía típica de una aplicación informática para proponer un modelo evolutivo y posteriormente se describen sus características y módulos que lo conforman para implementarlo.

Introducción

El proceso de generación de una aplicación informática típicamente se siguen las etapas de:

- 1) *Análisis*. En la que se detectan los usuarios y sus requerimientos de información.
- 2) *Diseño*. Se establece el modelo de datos y procesos para resolver los requerimientos detectados en la etapa anterior.
- 3) *Implantación*. En donde se lleva a cabo el diseño físico de las bases de datos, se elaboran los programas, se efectúan las pruebas, se capacita a los usuarios y se libera el sistema. Luego de realizar el sistema se realiza un proceso rutinario de mantenimiento, durante el cual se incluyen aspectos que no se habían considerado en alguna de las etapas del desarrollo, o bien, modificaciones que han surgido como consecuencia de la dinámica del ambiente en que se encuentra el sistema informático.

Gabriel Cordero[1] en un planteamiento dado en 1985, detectó que más que hacer sistemas rígidos con formatos preestablecidos *“Es importante, al menos por ser un hecho que pudo observarse con el uso de”* su Reconocedor de Lenguaje Natural aplicado a la Recuperación de Datos, *“hacer notar como la persona que opera con el reconocedor (y cabe mencionar que esta persona ha operado con otros sistemas de recuperación de datos), como realmente lo único que requiere es conocer la información que maneja en su trabajo, y al percatarse de no tener una vía preestablecida para obtener datos suele decidir por mejores opciones (presentaciones de datos) diferentes a las que de origen fueron planteadas por el usuario mismo. Esto tiene como consecuencia que un usuario vea y sienta al sistema como una verdadera herramienta y no como un algo normativo que lo eduque, pensamiento, bajo el cual aún se rigen muchos desarrolladores de sistemas”*.

⁴ Jesús Manuel Olivares Ceja, Ciudad de México, 23 de mayo de 1997.

Esta y otras ideas revisadas en [2][3][4][5] son antecedentes de lo que ahora se conoce como Sistemas Evolutivos que son un paradigma útil en la conceptualización e implantación de los sistemas de información, los cuales el modelo es evolutivo[6], por ejemplo como el de la figura 1.

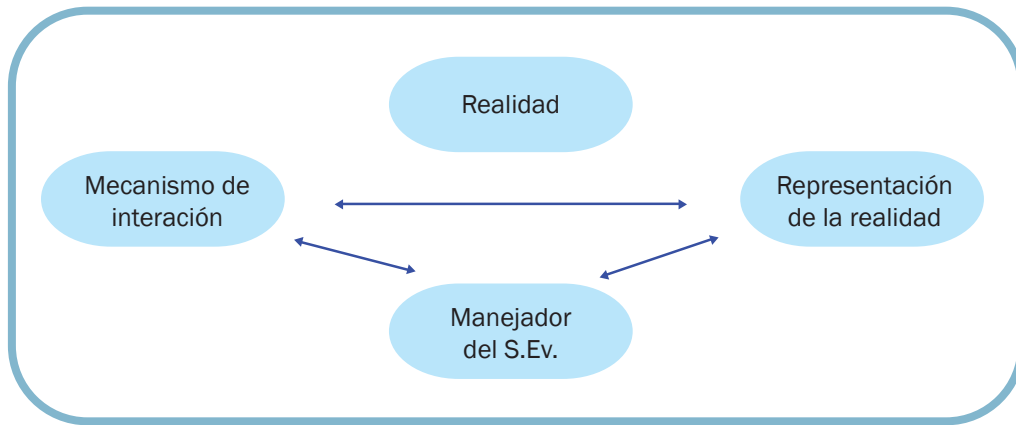


Figura 1. Arquitectura general de un sistema evolutivo.

El modelo de Representación de la Realidad nos permite almacenar una imagen de una realidad dada en términos de sus elementos, las relaciones entre éstos o estructura y su significado (figura 2).

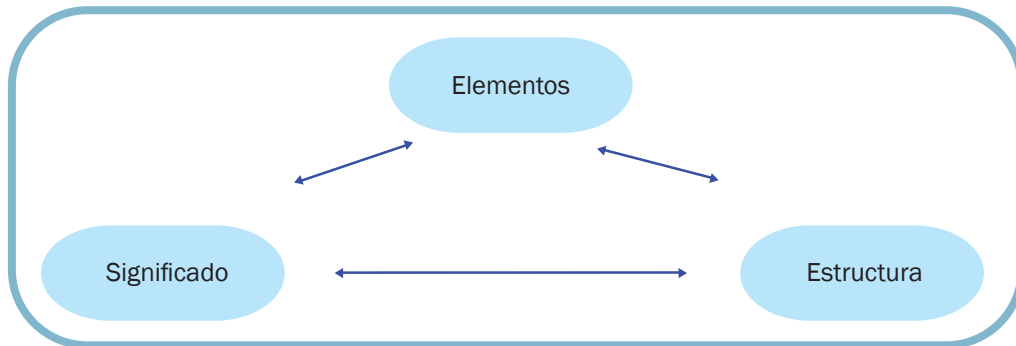


Figura 2. Representación de la realidad de un sistema evolutivo.

I Aplicaciones Informáticas

La atención a requerimientos generalmente se da mediante la presentación de información a través de pantallas, listados o gráficas. Estos se obtienen mediante un conjunto de programas que realiza el grupo de desarrollo del sistema.

Los elementos que aparecen en la pantalla, comúnmente son (figura 3):

- 1) *Campos*. Donde el usuario puede introducir datos al sistema.
- 2) *Letreros*. Son guías que indican al usuario, las acciones que deben realizarse o le explican el contenido de la pantalla.

- 3) *Botones*. Estos disparan alguna acción o conjunto de acciones.
- 4) *Listas de Datos*. Sirven generalmente para que el usuario haga alguna selección.
- 5) *Áreas de Datos Multimedia*. Estas pueden ser imágenes, videos, sonido, etc.

Directorio telefónico

Nombre:

Dirección:

Teléfono:

Figura 3. Elementos que aparecen en la pantalla de una aplicación informática.

Los datos se requieren manejar en la memoria de la computadora y algunos en un almacén permanente como el disco duro, en este caso generalmente se utiliza algún manejador de base de datos. Una base de datos comúnmente se compone de un esquema, archivos, índices y de un conjunto de procesos que los manipulan. Estos se representan en la figura 4.

Esquema de la entidad Persona

Atributo	Tipo (logitud)	Inicio
nombre	A(35)	0
dirección	A(50)	35
teléfono	A(30)	85

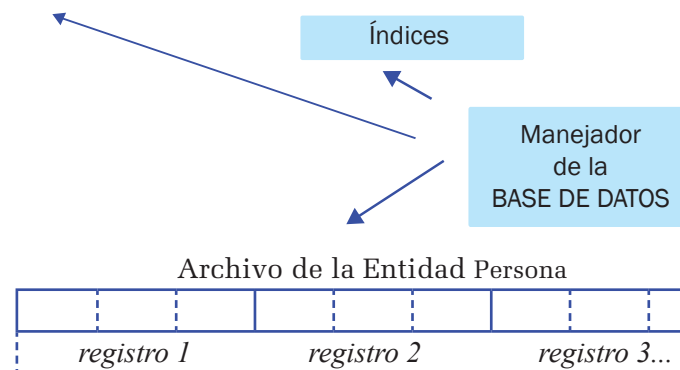
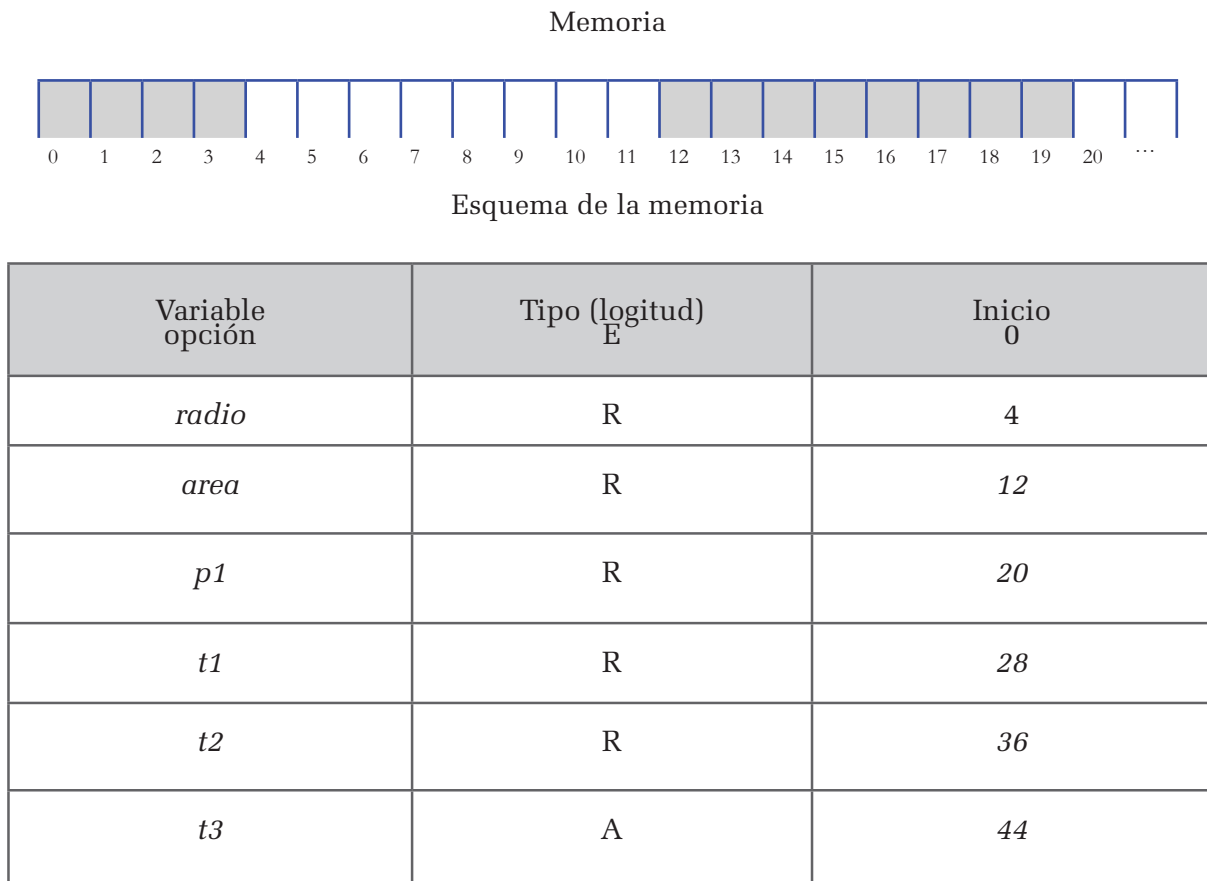


Figura 4. Componentes de un manejador de base de datos.

Los procesos que componen al manejador de la base de datos generalmente son:

- 1) *Manejador de Archivos*. Incluye funciones para crear el esquema de una entidad, y el archivo (o espacio en disco) donde se almacenan los datos. Entre otras hay funciones para recuperar y almacenar un registro dado, recuperar y almacenar los atributos en un registro.
- 2) *Manejador de Índices*. Incluye funciones para creación y manejo de índices (insertar, eliminar, buscar), las técnicas más utilizadas son árboles B y Hashing.
- 3) *Manejador de Transacciones*. Incluye funciones para manejo de transacciones y recuperación en caso de alguna falla, esto mediante una bitácora. En este caso se utiliza un lenguaje de manejo de datos (dml, data management language).

Otro componente de las aplicaciones son el conjunto de operaciones que se efectúan sobre los datos, conocidos como “programas de aplicación”. Un programa es un conjunto de instrucciones, algunas que accedan al manejador de base de datos, otras accedan al manejador de pantallas y otras actúan sobre datos almacenados en la memoria. Por lo que un intérprete de instrucciones tiene la arquitectura mostrada en la figura 5.



donde E=entero, R=real, A=alfabetico.

Instrucciones:

Dir	Instrucción	Lenguaje maquina
0	asigna pi 3.141592	1 20 36
1	desplega “digite el radio: “	2 44
2	lee radio	3 4
3	cuadrado radio t1	4 4 28
4	multiplica pi t1 area	5 20 28 12
5	desplega area	2 12
6	alto	0

Figura 5. Arquitectura de un interprete de instrucciones.

El manejador de pantalla se compone de una tabla de elementos y un conjunto de procesos que los manipulan (figura 6).

Elementos graficos:

Elemento	Coordenadas (x y) → (x y)	Atributos (color, tipo de letra, etc.)
Letrero 1	30 10 50 10	DIRECTORIO TELEFONICO
Letrero 2	5 15 20 15	Nombre:
Campo 1	25 15 70 15	A(35)
Letrero 3	5 25 20 25	Direccion:
Campo 2	25 25 80 25	A(50)
Letrero 4	5 35 20 35	Telefono:
Campo 3	25 35 60 35	A(30)
Boton 1	20 50 30 50	ALTA
Boton 2	40 50 50 50	ALTA
Imagen 1	60 30 90 50	16 colores

Figura 6. Componentes del manejador de pantallas.

La interacción con el usuario se hace a través del mismo lenguaje del usuario, esto se hace mediante un proceso evolutivo para que el sistema adquiriera el lenguaje del usuario. Se han utilizado distintas estrategias para que el sistema adquiriera el lenguaje del usuario varias de ellas se describen en [17]. La más sencilla consiste en usar el espacio para separar cada unidad léxica, y asociarle un tipo, como en el ejemplo siguiente, tomado de [5]:

Calcula la Regresión de $X * Y, Z$

de donde:

<i>Calcula</i>	<i>accion</i> (a)
<i>la</i>	<i>ignora</i> (1)
<i>Regresion</i>	<i>accion</i> (a)
<i>de</i>	<i>ignora</i> (1)
<i>X</i>	<i>datp</i> (d)
<i>*</i>	<i>accion</i> (a)
<i>Y</i>	<i>dato</i> (d)
<i>,</i>	<i>separador</i> (s)
<i>Z</i>	<i>dato</i> (d)

de donde se obtiene la producción gramatical conocida como oración canónica:

$S \rightarrow a i a i d a d s d$

en las producciones de la gramática se adicionan las rutinas semánticas para que cuando se genere reconozca una oración o un conjunto de ellas se generen las instrucciones correspondientes y se ejecuten mediante el interprete descrito anteriormente.

Integrando los componentes descritos hasta ahora, la arquitectura del Sistema Evolutivo Generador de Aplicaciones es como se muestra en la figura 7.

Es conveniente notar que cualquier otro conjunto de manejadores debe adicionarse como otro módulo para el intérprete, o bien puede expresarse como un conjunto de llamados a rutinas de los diferentes manejadores que ya se tienen.

El lenguaje natural del usuario se utiliza para obtener el esquema de la base de datos como se explica en [14] o bien construir nuevos procesos para atender sus requerimientos específicos, mediante la composición de funciones como se describe en [10] para construir moléculas, lo cual se puede realizar mediante un mecanismo de diálogo.

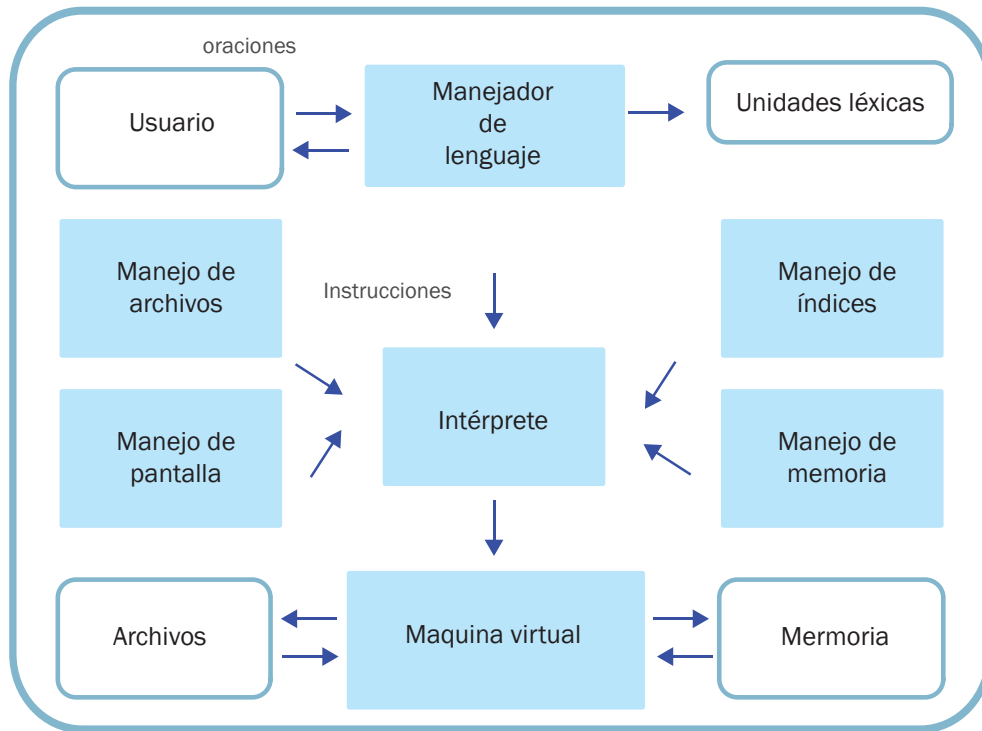


Figura 7. Arquitectura del Sistema Evolutivo Generador de Aplicaciones.

Conclusiones

Se describió la problemática encontrada en las aplicaciones informáticas en cuanto a la falta de flexibilidad y costos excesivos ocasionados por la reprogramación constante que se requiere durante los mantenimientos. Se propuso como alternativa utilizar el enfoque de los Sistemas Evolutivos y se describieron los componentes de un Sistema Evolutivo Generador de Aplicaciones, el cual aprende el lenguaje del usuario y del mismo puede obtener el esquema de la base de datos, los procesos y la descripción de la interfase; asimismo en el mismo lenguaje se pueden expresar los requerimientos que el usuario plantee, dándole oportunidad que el mismo pueda efectuar ajustes en cuanto a los resultados que obtiene del sistema.

Bibliografía

- [1] Cordero Sánchez, Gabriel, “Aplicación de un Reconocedor de Lenguaje Natural Restringido a la Recuperación de Datos”
- [2] Mendoza Padilla, Roberto, “Nuevo Método para el Desarrollo de Sistemas de Información”
- [3] González Vázquez, Juan Martín, “La Comunicación Hombre-Maquina Utilizando Lenguaje Natural Restringido”
- [4] Galindo Soria, Fernando, “Sistemas Evolutivos” en Boletín de Política Informática, INEGI 1986.
- [5] Galindo Soria, Fernando, “Sistemas Evolutivos: Nuevo Paradigma de la Informática”
- [6] Galindo Soria, Fernando, “Arquitectura Lingüístico-Interactiva para Sistemas Evolutivos”
- [7] Galindo Soria, Fernando, “Sistemas Evolutivos de Reescritura”
- [8] Galindo Soria, Fernando, “Una Representación Matricial para Sistemas Evolutivos”
- [9] Olicón Nava, Carlos, “Sistema Evolutivo para Generar Figuras y Realizar Paisajes con Movimiento en 2D”
- [10] Olivares Ceja, Jesús Manuel, “Sistema Evolutivo Graficador de Moléculas Organicas”
- [11] Galindo Soria, Fernando, “Generador de Sistemas Como Núcleo de un Sistema Evolutivo”
- [12] Vicario Solorzano, Claudia Marina, Aguilar Vallejo, Francisco Javier, “EVA:: Evolucion Aplicada”
- [13] Ambriz Carreón, Guillermo, “Sistema Evolutivo para el Control Estadístico del Proceso en un Ambiente Distribuido”
- [14] Berruecos Rodríguez, Elsa, “Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos”
- [15] Galindo Soria, Fernando, “Sistemas Evolutivos Basados en Conocimiento”
- [16] Olivares Ceja, Jesús Manuel, “Sistema Evolutivo para Representacion del Conocimiento”
- [17] Galindo Soria, Fernando, “Concatenacion y Desconcatenacion: Operaciones Fundamentales de la Lingüística “

VI.4 Sistema Evolutivo para el Control Estadístico del Proceso en un Ambiente Distribuido

Guillermo Ambriz Carreón⁵

Introducción

En la actualidad muchos de los equipos de medición (calibradores, básculas, dinamómetros, multímetros, etc.) están diseñados para poder comunicarse con una computadora personal (PC, por sus siglas en inglés), de esa manera se envían las medidas obtenidas hacia la PC para posteriormente poderlas procesar. Sin embargo en la mayoría de los casos esas medidas obtenidas son almacenadas en archivos de paso (se crean y posteriormente se borran) o en el peor de los casos son registradas en un formato establecido por las empresas o por las compañías, para posteriormente ser capturadas en la PC y así después procesar la información en paquetes que realizan las gráficas necesarias para la empresa. Este trabajo hace mención a un paquete capaz de registrar la información emitida por algún equipo de medición y procesarla para la elaboración de las gráficas solicitadas.

De los equipos citados anteriormente cabe mencionar que un gran número de ellos se emplean para la industria manufacturera y/o maquiladora razón por la cual se utilizan para verificar que las piezas o materiales elaborados cumplan con las especificaciones requeridas por parte de los departamentos de control de calidad y/o ingeniería, pero sobre todo monitorear el comportamiento de las máquinas que elaboran dichos productos o materiales. Este monitoreo se realiza comúnmente por herramientas para el Control Estadístico del Proceso (CEP), para determinar así, si la maquinaria es capaz y hábil de producir con los requerimientos del cliente.

Algunos de los equipos de medición cuentan también con un programa para elaborar las gráficas más comunes para el CEP, sin embargo, esos programas están diseñados para el equipo de medición y tiene costos estratosféricos, de aquí que encontramos las siguientes desventajas:

Palabras Claves: Sistema Evolutivo, Control Estadístico del Proceso (CEP), Carta de Control, Ambiente Distribuido.

⁵ Guillermo Ambriz Carreón Instituto Tecnológico de Toluca.

- 1) Por ser paquetes que están diseñados para el equipo de medición, *el hardware muchas ocasiones debe ser especial* (diseñado por el mismo distribuidor).
- 2) El *soporte técnico en la mayoría de los casos es pobre* (personal que desconoce el paquete debido a que no es totalmente comercial).
- 3) En algunos casos se necesita de un *aparato adicional* para poder procesar la información correspondiente.
- 4) Las partes que requiere en ocasiones suelen ser *estándares europeos, por lo que en México no se pueden operar y/o conseguir refacciones*.

I Características Generales del Sistema

Este paquete puede comunicarse con el equipo de medición a través del conector RS-232, también puede trabajar con más de una PC, es decir, se pueden elaborar las gráficas correspondientes que hemos de utilizar y ser manipuladas en un ambiente distribuido totalmente transparente para el usuario, esto representa muchas ventajas con respecto a la mayoría de los paquetes comerciales:

- 1) *No es necesario almacenar la información* (lecturas) en un archivo de paso para poder evaluar dicha información.
- 2) *No todos los paquetes comerciales pueden trabajar en un ambiente distribuido*.
- 3) *El intercambio de datos es totalmente dinámico*, de aquí que la información que se pueda manejar en dos PC diferentes siempre esta actualizada, y esto evita posibles confusiones por utilizar información no actualizada.

También es posible configurar las características del puerto serial con el fin de poder conectar más de un aparato en una PC siempre y cuando la computadora cuente con más de uno de ellos. Las configuraciones que realiza este paquete se muestran en la tabla 1.

Característica	Opciones de configuración
Puerto seleccionado:	COM1, COM2, COM3, COM4
Velocidad:	300, 600, 1200, 2400, 4800, 9600 baudios
Paridad:	Par, Impar, Sin Paridad
Longitud de dato:	7, 8 bits por dato
Bits de parada:	1, 2 bits

Tabla 1 Configuración del Puerto Serial.

La opción de configuración del puerto serie tiene como finalidad ajustarse a los diferentes aparatos de medición ya que no todos trabajan a la misma velocidad, o la longitud del dato no es siempre de 8 bits, etc., por lo que las opciones anteriores nos auxilian para la recepción de los datos sin problema alguno.

Cuando se recibe un dato por la PC, el paquete vuelve a realizar los cálculos necesarios para la elaboración de la gráfica e inmediatamente después actualiza la gráfica, de esta manera la información se tiene actualizada y sobre todo oportuna ya que el operario hace un análisis en ese momento del comportamiento de la máquina o del proceso que se este evaluando.

Si al recibir un dato la PC y al realizar los cálculos encuentra que el dato recibido genera una variación o un punto fuera de control lo indica con un mensaje y además señala con un punto rojo parpadeante el dato en la gráfica para una mejor identificación por parte del operario.

Por lo descrito anteriormente podemos pensar que este paquete solamente puede elaborar gráficas mientras reciba las lecturas de un equipo de medición, esto no es así, ya que si así se desea, la captura de la información se registra manualmente, esta opción es sin lugar a dudas necesaria para algunas empresas, procesos, técnicas, etc., esto es justificable de acuerdo a la naturaleza de las causas.

Otras de las características de este paquete es que se pueden almacenar las lecturas recibidas por la PC o los datos capturados por el usuario, de esa forma aseguramos que los datos registrados pueden ser consultados, modificados, actualizados o incluso se pueden elaborar nuevamente sus gráficas para consultas posteriores. Además de que la información puede ser almacenada existe también la opción de exportar la gráfica a un formato para la edición de algún documento, los formatos que puede exportan son WMF y BMP.

La exportación de las gráficas en alguno de los formatos ya mencionados nos apoyan a la elaboración de documentos con una mayor presentación y por ser gráficas en los dos formatos más usuales, estas se pueden utilizar en paquetes para edición de documentos como Ventura Publisher®, Microsoft Word®, Microsoft Works®, CorelDraw®, WordPerfect®, WordPerfect Presentations®, Microsoft Publisher®, etc., pero sobre todo que ese tipo de paquetes permite escalar las gráficas dentro de la página para ajustarlas a los tamaños y posiciones deseados.

Las gráficas que realiza este paquete son las gráficas X-R, Cpk, Cp y Cmk; estas gráficas son las más utilizadas para el CEP, sin embargo no son las únicas, razón por la cual se está trabajando para poder realizar otro tipo de gráficas para el CEP, entre ellas, las gráficas por atributos (gráfica c, gráfica p, etc.) y complementar las gráficas por variables (X-R, X- S, etc.).

Ahora bien, los paquetes que trabajan bajo ambiente Windows® comparten dos características comunes, la ayuda en línea y la barra de herramientas, este paquete no es la excepción, pues cuenta con su ayuda con las mismas características de las ayudas de Windows®, mostrando la información por índice, historial e hipertexto, y este último se ha convertido en una herramienta muy accesible para el usuario, ya que puede seleccionar un tema con solo presionar el botón izquierdo del ratón (mouse) en una palabra marcada con un color diferente al resto del texto.

La función de la barra de herramientas es permitir un acceso rápido a las órdenes más utilizadas en una aplicación, por ejemplo, en Word® al presionar el botón del ratón en un ícono con la figura de un disco realizamos la grabación de la información en nuestro disco ya sea flexible o en disco duro. El realizar las órdenes más utilizadas por medio de “clicks” del ratón en íconos con figuras relacionadas a esa orden hacen más amigable el uso de algún paquete.

En muchos casos necesitamos analizar los datos de una gráfica e inmediatamente después otros datos de otra gráfica distinta, considerando esos casos, se diseñó este paquete con la propiedad de manejo de múltiples documentos, es decir, que podamos tener en la pantalla más de un concentrado de datos de diferentes archivos o gráficas según el caso. Varias aplicaciones que trabajan bajo el ambiente Windows® así lo hacen, tal es el caso de Word®, que podemos tener más de un documento abierto y que seleccionamos el que deseamos consultar por medio del menú general a través de una lista con los nombres de los documentos abiertos.

Si bien, dentro de las características mencionadas hasta el momento, estamos seguros de que son de gran utilidad para la empresa, la siguiente característica provoca un giro totalmente a todos los conceptos tradicionales de este tipo de paquetes.

Imaginémonos que estamos en alguna empresa en la cual se utilizan las gráficas del CEP y que durante una de las lecturas o captura de la información se genera un punto fuera de control, el procedimiento más común en esos casos es lo siguiente:

- 1) *Reportar al Supervisor de área que existe un punto fuera de control.*
- 2) *El supervisor verifica la existencia de ese punto; y detiene temporalmente el proceso para su análisis.*
- 3) *El supervisor hace el análisis correspondiente para poder determinar la causa que originó la falla, posteriormente toma las medidas correspondientes para asegurar que el proceso tendrá el comportamiento deseado.*
- 4) *Si las medidas tomadas solucionan el problema, el supervisor lo registra para consultas posteriores en casos iguales o similares.*

Como podemos observar, el realizar estos cuatro pasos consumen un tiempo que en múltiples casos detienen la línea de producción demorando tiempos de entrega, producción final, supervisión de producto, proceso, etc., sin embargo, son necesarios para asegurar el producto que se ha de enviar al cliente, y esto último justifica en parte, el retardo de una producción ya comprometida.

Estos pasos citados con anterioridad pueden reemplazarse por actividades de la misma computadora. Esto se logra con el paradigma de los Sistemas Evolutivos, que establece que un sistema debe ser capaz de interactuar con la realidad en la que está inmerso para crear una imagen de ésta y con ella resolver los problemas que se le plantean.

Por ejemplo, pensemos en una máquina cortadora/despuntadora de cable, y que durante el proceso detectemos que la longitud es muy variable, esto se ve reflejado por la carta de control X-R (gráfica de Medias y Rangos), su gráfica se muestra en la figura 1.

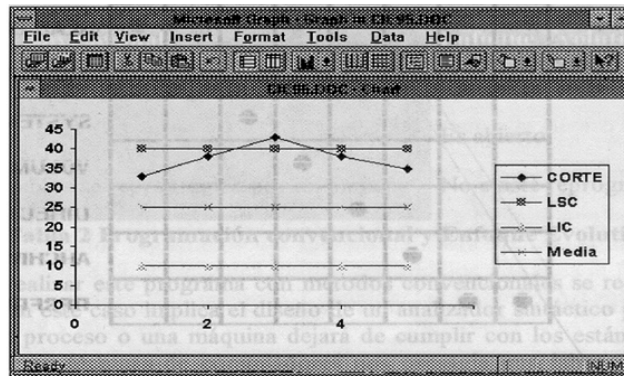


Figura 1 Carta de Control.

El experto en interpretación de Cartas de Control le indica a la computadora que para una gráfica de este tipo la solución está en cambiar los rodillos de la cortadora, entonces esto lo podemos registrar de forma escrita como sigue:

Tipo de Gráfica: Rodillos Desgastados	Sugerencia: Cambiar los rodillos
---------------------------------------	----------------------------------

Por el contrario, si una gráfica nos indica que el proceso o la máquina están dentro de los estándares de control de calidad se indica de la siguiente manera:

Tipo de Gráfica: Correcta	Sugerencia: Ninguna
---------------------------	---------------------

El número de sugerencias irá creciendo conforme lo requiera la empresa o las situaciones en las que cae. Así se eliminan tiempos de captura, posibles tiempos en resolución de problemas, pero sobre todo el contar con un equipo que contenga en su bitácora las sugerencias para cada uno de los posibles casos que se presenten.

También, es posible a través del paradigma de sistemas evolutivos el poder darle la capacidad de predecir cuándo un proceso o una máquina se saldrá de los estándares requeridos, es decir, indicarnos cuándo una máquina requiere mantenimiento preventivo; del ejemplo de la gráfica anterior se indicó en la sugerencia el cambio de rodillos, cuando llegue nuevamente una gráfica que su comportamiento en los primeros puntos (los 3 primeros en este caso) sea similar a la figura anterior nos mostrará un mensaje en el cual nos indica que la máquina va a requerir un cambio de rodillos.

II Enfoque Evolutivo vs Programación Convencional

Con los métodos convencionales de programación se puede realizar un paquete con las mismas características, pero existen tres grandes diferencias contra el enfoque evolutivo (véase la tabla 2), esas diferencias son las que permiten que el programa interactúe con la realidad para resolver los problemas planteados.

Programación convencional	Enfoque Evolutivo
No capta la realidad	Capta la realidad
Está restringido	Es abierto
Los ajustes y cambios se reprograman por humano	No existe reprogramación humana

Tabla 2 Programación convencional y Enfoque Evolutivo.

Para poder realizar este programa con métodos convencionales se requiere de técnicas de compiladores, que en este caso implica el diseño de un analizador sintáctico predictivo para poder predecir cuando un proceso o una máquina dejará de cumplir con los estándares solicitados. De antemano el pensar en técnicas de compiladores nos presagia una labor de “talacha” como la conocemos en el ámbito de la computación. El modelo para el analizador lo podemos ver en la figura 2.

A grandes rasgos, para la elaboración de un analizador sintáctico predictivo necesitamos primero eliminar la recursión izquierda de la gramática, posteriormente una factorización por la izquierda ya que ésta es una transformación gramatical útil para producir una gramática adecuada para el análisis sintáctico predictivo, no omitiendo la tabla de análisis sintáctico M sin descuidar que la gramática a elaborar debe ser del tipo LL(1) [1].

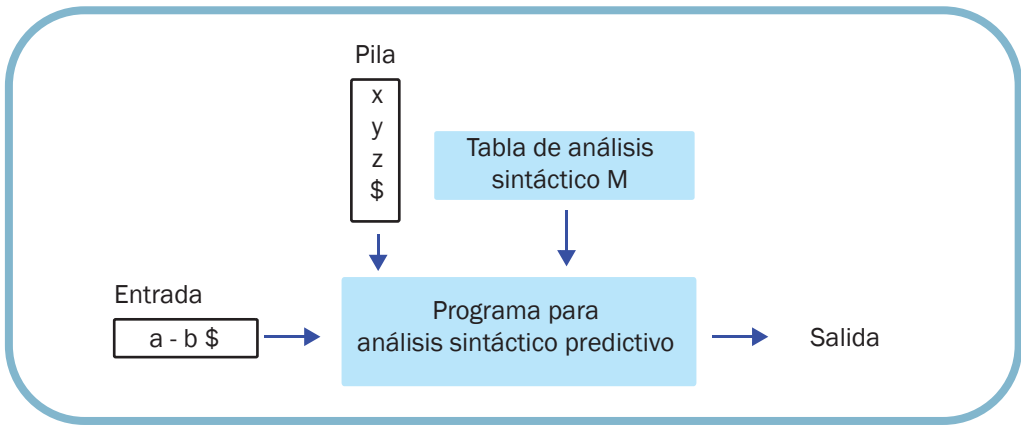


Figura 2 Diagrama de un analizador sintáctico predictivo.

Esta metodología implica un consumo grande de tiempo en la creación de la gramática, y creación de la tabla M, además se deben evitar errores en la elaboración de ambas ya que de lo contrario se reestructurará gran parte de estas si se comete error alguno.

Por su parte, el paradigma de los sistemas evolutivos también consta de herramientas para la realización de los programas que deseamos elaborar. Existen varias herramientas, pero para el diseño de este programa se utilizaron dos herramientas: la matriz evolutiva y la factorización. La primera es la que contiene la información de todos los casos o tipos de gráficas que se han elaborado y se registran en esta matriz. Entiéndase por caso o tipo de gráfica a la característica de la gráfica (correcta, rodillos desgastados, balero desnivelado, etc.). La herramienta de la matriz evolutiva nos apoya para almacenar lo que equivale a la tabla de análisis sintáctico M.

Por otro lado la factorización nos auxilia a la creación de la gramática, la cual gracias a esta herramienta la crea la computadora y no el programador. Esta herramienta (factorización) es equivalente a la factorización algebraica con la diferencia de que en este caso se factorizan componentes de una oración [2].

La matriz evolutiva contiene los elementos terminales (al igual que la tabla M) que en este caso son cada uno los datos registrados del aparato de medición, así, si una gráfica correcta la formaron los datos 8, 10, 8, 9, y 10; y una gráfica con rodillos gastados la forman los datos 7, 8, 10, 6, y 11, se crea un archivo que contiene 7 campos, los primeros 5 almacenan las lecturas, el sexto el tipo de gráfica y en el séptimo se registra la acción a tomar; en este caso la matriz tiene dos renglones (correcta y rodillos desgastados).

En la matriz no se indican los no terminales, ya que estos se obtienen con la factorización y se almacenan en otro archivo, siendo este archivo el que contiene la gramática; para ello se comparan cada uno de los renglones de la matriz evolutiva y en el momento que cambien un elemento con respecto a otro se sustituyen el resto de los valores por un no terminal y se reemplazan los valores en la nueva producción (el no terminal) generando así la gramática.

III Matriz Evolutiva y Factorización

En cada renglón de la matriz evolutiva (tabla 3) se almacena la serie de datos y el diagnóstico o sugerencia para cada tipo de gráfica, la serie de valores se calcula de acuerdo a las necesidades del problema, en este caso es la diferencia entre cada dato leído con respecto a la media aritmética de la serie de datos. Cada nuevo renglón de la matriz se ingresa y se calcula un valor de referencia que se denomina umbral este valor se calcula de acuerdo a las necesidades y naturaleza de la información, en este caso se calculó con la desviación estándar (de aquí que el umbral del primer elemento es 0.7483, del segundo es 1.3266 y así sucesivamente).

Así cuando llegue una nueva gráfica se calcula el umbral y se compara con cada uno de los renglones existentes, se obtiene una diferencia entre umbrales y si es mínima se le indica al

usuario que es una gráfica del tipo “x”, sin embargo el usuario puede indicar el tipo de la gráfica si así lo desea. Por otro lado, si la diferencia entre los existentes es muy alta, solicita el tipo de la gráfica y lo ingresa a la matriz junto con su sugerencia.

Datos					Diagnóstico
7	6	5	7	6	Correcta
6	6	4	7	8	Rodillos desgastados
5	5	7	5	3	Cuchillas sin filo
3	3	6	6	5	Peso desnivelado

Tabla 3. Matriz Evolutiva.

Si a una gráfica se le indica un tipo ya existente entonces no se anexa otro renglón sino que se le suma vectorialmente el nuevo renglón al renglón existente, para poder así obtener lo que podemos llamar la firma de la gráfica.

Con lo que respecta a la herramienta de factorización debemos recordar que es igual a la del álgebra, por ejemplo, si se ingresan los siguientes datos y se indica que es una gráfica correcta, es importante aclarar que la primera ocasión nuestros archivos se encuentran vacíos (gramática y matriz evolutiva):

7	6	5	7	6	Correcta
---	---	---	---	---	----------

Este renglón ingresa a nuestra matriz evolutiva. Como no existen datos en nuestro archivo de gramática nos queda que:

S: 7 6 5 7 6

Donde “S” es una llave de referencia (puede ser un apuntador, un carácter, una dirección en disco, etc.). Más adelante nos llega otra serie de datos junto con su diagnóstico, se compara con el renglón existente, el umbral difiere considerablemente por lo que se anexa a nuestra matriz:

7	6	4	6	8	Falta energía
---	---	---	---	---	---------------

Se compara con nuestra gramática existente, observe que ambos renglones inician con el 7 y 6, de aquí se dice que:

S: 7 6 X

X: 5 7 6

X: 4 6 8

Si observamos bien esta es una pequeña gramática, la cual se construyó de una manera sencilla, y con esta herramienta y de esta manera es la computadora la que obtiene la gramática y no el programador. Recuerde que en la matriz evolutiva se encuentran cada uno de los tipos de gráficas y es de ahí de donde tenemos que realizar la factorización. El programa debe realizar esa factorización ya que de lo contrario nosotros la tendríamos que realizar y eso sería programación convencional.

Un pseudocódigo básico para realizar la factorización es el que se muestra en la tabla 4.

Tanto el vector_uno como el vector_dos son arreglos que contienen el primer y segundo renglón de la matriz evolutiva a comparar para poder factorizar. Así la siguiente vez que ingrese una nueva gráfica a la computadora buscará si existe en la matriz evolutiva, posteriormente verificará su comportamiento en el archivo de la gramática para predecir posibles fallas; por ejemplo, supóngase que se tiene una gráfica con los valores 6, 6, 4, 8, 8; al analizar su gramática se da cuenta que los tres primeros valores corresponden al tipo de rodillos desgastados y emitirá un mensaje de aviso informando que la máquina va a requerir dentro de poco tiempo comprar rodillos, rectificarlos o lo que sea necesario.

```

pos = 1
abrir archivo de gramática
mientras pos <= ultimo_elemento
si vector_uno[pos] = vector_dos[pos] entonces
    escribir en el archivo vector_uno[pos]
sino
    colocar en el archivo llave de referencia // es el no terminal "x"
    vaciar el contenido de vector_uno[pos] en adelante
    colocar en el archivo llave de referencia
    vaciar el contenido de vector_dos[pos] en adelante
    pos = ultimo_elemento
fin si
incrementar a pos en uno
fin mientras
cerrar archivo de gramática

```

Tabla 4. Pseudocódigo para factorización.

Bibliografía

- [1] Alfred V. Aho, “Compiladores. Principios, técnicas y herramientas”, Addison Wesley, Estados Unidos de Norteamérica, 1990.
- [2] Fernando Galindo Soria, “Algunas Propiedades Matemáticas de los Sistemas Lingüísticos”, Memoria del curso tutorial de Sistemas Evolutivos, Ier Congreso Internacional de Investigación en Ciencias Computacionales, Metepec México, 1994.

CAPÍTULO VII

SISTEMAS EVOLUTIVOS BASADOS EN CONOCIMIENTO

VII.1 Representación del Conocimiento

Fernando Galindo Soria¹

Introducción

La necesidad de contar con mecanismos que nos permitan obtener, preservar y transmitir el conocimiento es uno de los problemas que se presentan al tratar de comprender la realidad que nos rodea y en particular con la explosión de la información que se presenta en nuestro tiempo, esta necesidad de obtención-transmisión de conocimiento ha crecido en forma acelerada, por lo que cada vez se necesitan mecanismos más “compactos” que nos permitan conservar el conocimiento que se está manejando. Lo anterior ha propiciado el surgimiento de diferentes niveles de abstracción en la representación del conocimiento, comenzando con los mecanismos en los cuales simplemente se listan los hechos, continuando con aquellos en los que se busca ordenar y jerarquizar esos hechos y así hasta llegar a obtener reglas generales que explican y sintetizan el conocimiento aportado por múltiples hechos (por ejemplo, las leyes de la Física).

Con el surgimiento de la Computación, de la Informática y en particular de la Inteligencia Artificial, el problema de representar el conocimiento adquiere una relevancia trascendental ya que ahora se trata de que los sistemas automatizados capten, almacenen, transmitan, evalúen y generen conocimiento, de donde uno de los puntos cruciales de la Inteligencia Artificial se centra en los mecanismos que permitan representar el conocimiento en forma automatizada.

Por lo anterior, se presenta un panorama de las múltiples formas de representación del conocimiento y sus posibilidades de automatización, para lo cual se parte de la asociación de datos a hechos y objetos, se ven los diferentes niveles para relacionar datos (expresiones, funciones, relaciones, algoritmos, heurísticas) continuando con herramientas como las Bases de Conocimiento y el Álgebra Cualitativa y terminando con mecanismos en los cuales el conocimiento no se almacena en un nodo específico, sino que se encuentra representado por toda la complejidad de un sistema, como es el caso, por ejemplo, de los circuitos de computadora (almacenan operaciones en el sistema) y las Redes Neuronales (almacenan conocimiento en el sistema).

¹ Fernando Galindo Soria, realizó este trabajo en junio 1985 en la Unidad de Investigación y Desarrollo de la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas del Instituto Politécnico Nacional.

1 Conceptos Generales

El ser humano siempre ha tratado de representar y comprender la realidad que le rodea y en ese intento ha generado múltiples teorías y desarrollado su creatividad artística, tecnológica y científica.

En particular, se considera que la informática es una herramienta con la cual se puede facilitar la construcción de modelos de la realidad: ya que, dado un universo conceptual en el cual existen relaciones de materia, energía, normas o datos entre los entes u objetos que forman el universo, se puede encontrar un modelo informático en el cual tanto el conocimiento que se tiene de los entes como de las interrelaciones entre ellos se pueden representar como datos y transformaciones entre estos.

De lo anterior se observa que uno de los problemas de la informática se centra en los mecanismos para representar el conocimiento que se tiene acerca de un universo dado.

Antes de estudiar los mecanismos específicos se verán algunas consideraciones sobre el conocimiento y sus diferentes niveles.

Desde el punto de vista de la pedagogía se tienen escuelas que estructuran los niveles de conocimiento de acuerdo a una secuencia o taxonomía, de estas la más conocida es la Taxonomía de Bloom, en la cual se considera que el conocimiento que se tiene de algo puede caer en alguna de las siguientes categorías:

- 1) *Conocer*, se sabe que existe algo.
- 2) *Comprender*, se entiende algo y se puede usar de alguna manera ese conocimiento.
- 3) *Aplicar*, dado un problema es capaz de solucionarlo utilizando el conocimiento adquirido.
- 4) *Analizar*, se es capaz de encontrar las partes constitutivas de algo y las relaciones entre estas.
- 5) *Sintetizar*, se es capaz de integrar diferentes partes para formar un todo.
- 6) *Evaluar*, se puede formar un juicio sobre algo.

Por otro lado, en el área de la Lingüística Matemática se manejan los siguientes niveles de conocimiento:

- 1) *Fonético*, se conocen los elementos “atómicos” de un universo.
- 2) *Léxico*, se conocen las relaciones entre los elementos atómicos para formar elementos básicos o “moléculas”.
- 3) *Sintáctico*, se conocen las reglas que indican cuales son las estructuras o relaciones entre elementos permitidas.
- 4) *Semántico*, se conoce el significado general de los elementos y de sus relaciones.
- 5) *Pragmático*, se conoce el significado particular o referencia a un contexto de los elementos y sus relaciones.

Si se observa el proceso de concretización de una disciplina o área del Conocimiento se puede observar que en general esta pasa por los siguientes niveles:

- 1) *Temas aislados*, se tiene una serie de temas reunidos en forma circunstancial (diccionario).
- 2) *Temas agrupados*, los temas se agrupan en temas generales según un criterio de agrupamiento (enciclopedia).
- 3) *Temas jerarquizados o disciplina taxonómica*, los temas se integran según un orden y se tiene una primera relación entre ellos (biblioteca especializada).
- 4) *Sistema de conocimiento o disciplina empírica*, en este nivel a partir de los temas se abstraen reglas, técnicas, métodos y principios no aceptados en general y la disciplina se describe en término de estas abstracciones y no de los temas específicos.

2 Formas Básicas de Representación del Conocimiento

El conocimiento que se tiene de una realidad puede caer en cualquiera de los enfoques y niveles presentados anteriormente, por lo que es difícil pensar en un único mecanismo para representar el conocimiento (esto no indica que no exista tal mecanismo, sino únicamente que, el nivel de conocimiento sobre este tema es aún elemental).

Por lo que a continuación se verán algunas de las formas de representación que se manejan actualmente y en particular aquellas que son independientes de una computadora.

La forma más sencilla de representar el conocimiento consiste simplemente en asociar un dato o etiqueta a lo que se quiere representar, como por ejemplo, asociar un nombre a una persona y almacenar ese nombre o asociar un número a cierta cantidad de manzanas, un mecanismo más complejo de representación surge cuando se observa que existe una relación entre el número de manzanas que se obtiene al juntar varios grupos de manzana y el número de manzanas de cada grupo y que este tipo de relaciones se mantienen independientemente de si lo que unen son manzanas o cualquier otra cosa.

Después de observar múltiples veces este fenómeno se puede pasar a un nuevo nivel de representación de conocimiento en el cual se plantea una regla del tipo Si unimos un grupo con “n” elementos y otro grupo con “m” elementos en total se tienen “l” elementos siguiendo este proceso, mas adelante se llega a obtener una regla general, por ejemplo, para la suma. Es así que en la regla de la suma se encuentra representado el conocimiento de múltiples generaciones. Este proceso se ha repetido para las otras reglas básicas de la aritmética, donde cada regla es una forma de representar conocimiento.

Pasando a otro nivel se tiene que, por ejemplo, la regla de la suma se puede abstraer y generalizar para representar la suma entre cualquier grupo de datos con lo cual se pasa a una nueva forma de representar el conocimiento en donde se representan las relaciones entre los datos de un universo dado como ocurre por ejemplo en el Álgebra.

Las operaciones aritméticas son un caso particular de lo que se conoce como una relación y nuevamente este concepto nos marca un nuevo nivel de representación de conocimiento, ya que al establecer una relación se están generalizando múltiples casos particulares.

Nuevamente se puede observar que al pasar de un nivel de representación a otro existe una compactación del mecanismo con lo que, por ejemplo miles de reglas particulares se integran en una regla general que las abarca y las sustituye.

Hasta este punto se han mencionado mecanismos de representación que funcionan como reglas generales, sin embargo conforme los problemas atacados crecen en complejidad, la misma regla general es difícil que se quede en una sola acción u operación, es por eso que se requieren mecanismos más complejos para representar conocimiento.

Estos mecanismos se pueden dividir en dos formas:

- 1) *Grupos de reglas.*
- 2) *Tablas de relación.*

El primer caso se encuentra que dado un problema, se puede plantear un conjunto de reglas que representan el conocimiento necesario para resolver el problema. Nuevamente este conjunto de reglas no surge por lo común en forma espontánea, sino que, es el resultado del trabajo de múltiples gentes, cuando estas reglas se orientan a obtener una solución independientemente de que sea la mejor o no, se conocen como heurísticas y prácticamente todas las actividades comunes están repletas de heurísticas o reglas de trabajo.

Por su parte cuando las reglas pretenden llegar a obtener la mejor solución a un problema se tiene un algoritmo.

Otra forma de representar un conocimiento complejo se tiene mediante las tablas de relaciones.

Nuevamente se tienen dos tipos de tablas, una en las cuales a cada relación entre elementos se les asocia un valor y este es inmutable y cuantitativo como por ejemplo en el caso de la suma, multiplicación, tangente, etc., etc.

Como no todos los problemas se pueden reducir a un esquema cuantitativo, en el otro tipo de tablas se representa un “conocimiento de todos los días” que no necesariamente es medible.

Combinación	Blanco	Gris	Negro
blanco	blanco	gris	gris
gris	gris	gris	negro
negro	gris	negro	negro

#	Blanco	Gris Claro	Gris	Gris Oscuro	Negro
Blanco	Blanco	Blanco	Gris Claro	Gris	Gris Oscuro
Gris Claro	Blanco	Gris Claro	Gris Claro	Gris	Gris Oscuro
Gris	Gris Claro	Gris Claro	Gris	Gris	Negro
Gris Oscuro	Gris	Gris	Gris	Gris Oscuro	Negro
Negro	Gris Oscuro	Gris Oscuro	Negro	Negro	Negro

En base a las relaciones cualitativas se puede construir toda una Álgebra Cualitativa en la cual se opere de acuerdo a las tablas cualitativas y se represente precisamente el “conocimiento de todos los días”, por ejemplo:

Gris # Gris Claro = Gris Claro

Gris Claro # Negro = Gris Oscuro

Negro # Blanco # Blanco = Gris Oscuro # Blanco = Gris

En un momento dado se pueden tener múltiples tablas y combinar las operaciones entre los datos para obtener nuevos datos.

3 Representación del Conocimiento en la Computadora

Cuando la cantidad de objetos que se quieren representar y las posibles relaciones entre estos crece en forma considerable surge la necesidad de herramientas automatizadas para representarlas, es por lo anterior que en el área de computación se cuenta con varios medios de representación del conocimiento, de los cuales los más sencillos constan solamente de datos almacenados en un campo (bit, carácter, palabra, etc.) concatenando varios campos se puede almacenar información más compleja ya que si se quiere almacenar la edad de 10 personas se colocan 10 campos para edad.

Por otro lado para representar las características que nos interesan sobre una persona se pueden tener un grupo de campos en cada uno de los cuales se almacena un dato diferente, por ejemplo, se puede tener campo de nombre, edad, sexo, grado de estudios, etc. y los valores que tomemos dan el conocimiento acerca de la persona por ejemplo: nombre/Juan Pérez, edad/30, sexo/m, grados de estudios/2do. de secundaria.

Entonces a cada objeto o ente que se quiera conceptualizar se le puede asociar un grupo de campos a los que se les asocia los atributos más relevantes.

Si en lugar de un ente se tienen varios entes del mismo tipo por ejemplo varias personas), entonces se puede agrupar la representación de todos los entes para formar una entidad.

Cuando existen múltiples entes de diferentes tipos se pueden formar agrupamientos de entes del mismo tipo en entidades y en un momento dado tener varias entidades que agrupan a los entes y cada ente está caracterizado por un conjunto de atributos.

A pesar de que esta forma de almacenar el conocimiento se puede considerar fuerte, resulta que tiene múltiples limitaciones ya que en un momento dado los entes representados se encuentran relacionados entre sí, y esa relación no se conserva, por otro lado se está manejando un nivel de representación muy elemental, ya que a cada ente se le asocia solo un conjunto de campos y no tiene otros mecanismos de abstracción.

Para atacar el primer problema se ha desarrollado en la Informática lo que se conoce como Base de Datos, para el segundo las Bases de Conocimiento.

En una Base de Datos además de almacenar los datos de cada ente, también se almacenan las relaciones sintácticas que existen entre diferentes entes.

Por su lado en las Bases de Conocimiento, a cada ente se le asocian relaciones con otros entes, buscando que en sí la relación representa un nuevo conocimiento y enfatizando la relación entre los entes más que las características de un ente en particular, con lo cual ya se cuenta con un mecanismo capaz de representar el conocimiento en una forma que se puede utilizar para inferir nuevo conocimiento.

Cuando se almacena el conocimiento en una Base de Conocimiento por un lado se tienen hechos o sea relaciones entre diferentes entes y por el otro lo que se conoce como reglas de inferencia con las cuales se tiene un nivel más avanzado de representación que indican características generales sobre los hechos y marcan que si se cumple un conjunto de circunstancias se puede inferir algo nuevo. Estas reglas son las que le dan su fuerza a las bases de conocimiento y le permiten manejar un nivel más abstracto y general aunque estas reglas de inferencia se aplican a cualesquier hechos que cumplan con sus premisas independientemente de los entes particulares que represente. Asociados con estos mecanismos se tiene lo que se conoce como rutinas semánticas, las cuales son, algoritmos o heurísticas que representan el conocimiento de cómo hacer algo y son llamadas o ejecutadas cuando se cumplen ciertas reglas y por otro lado tenemos el álgebra cualitativa con la cual se representa el “conocimiento de todos los días”.

La combinación de las Bases de Datos, Bases de Conocimiento, Rutinas Semánticas, Álgebra Cualitativa, etc, nos proporcionan un mecanismo muy poderoso de representación de conocimiento, ya que, en la base de conocimiento se almacenan los entes y las relaciones entre estos y las reglas que permiten inferir nuevo conocimiento a partir del ya existente, por otro lado los entes pueden apuntar a las bases de datos donde se encuentra la información de detalle y por su parte ciertas reglas de inferencia pueden inferir alguna rutina semántica o alguna operación cualitativa entre diferentes componentes.

4 Representación en Base a Propiedades Emergentes de las Estructuras

Al analizar un sistema como el propuesto anteriormente se puede observar que mucho del conocimiento no está almacenado en términos de datos concretos, sino que, surge como una manifestación de las relaciones entre los hechos y de las reglas que permiten combinar estos hechos. Es por lo anterior que un mecanismo de representación de conocimiento no tiene porque estar formado únicamente por nodos concretos en los cuales se almacene este conocimiento y se considera que se puede tener un mecanismo en el cual el conocimiento se encuentre difundido por todo el sistema y sea el resultado emergente de la concatenación de múltiples nodos aplicados a un caso particular.

Como ejemplos de mecanismos de este tipo se tienen los Circuitos Aritméticos y las Redes Neuronales.

En el caso de los Circuitos Aritméticos, por ejemplo, los sumadores binarios, se tiene un sistema en el cual se almacena el conocimiento de como sumar pero **no en una tabla de la suma ni en un algoritmo de suma**, sino en un mecanismo que obtiene la suma en base a relaciones que surgen en el sistema, lo cual indica que no existen datos representando el conocimiento sino que este se encuentra inmerso en todo el sistema.

Este ejemplo nos indica que un camino alternativo para representar el conocimiento, es logrando que se difunda en todo el sistema y que se aplique para cada caso particular, con lo cual se tiene un nivel de abstracción alto, ya que no son los datos concretos o reglas específicas, ni datos o reglas generales lo que se almacena sino una abstracción de todo esto aplicable a cualquiera datos que se puedan sumar.

De lo anterior surge la idea de desarrollar sistemas en los que el conocimiento no se almacena en un punto específico sino que se difunda en todo el sistema y en un momento dado no queden hechos concretos sino la abstracción de estos hechos.

Un mecanismo orientado a este fin, es lo que se conoce como Red Neuronal, una Red Neuronal o máquina que aprende, es un mecanismo compuesto por nodos y relaciones entre los nodos, donde cada nodo tiene un conjunto de entradas que pueden tomar dos posibles valores preestablecidos para cada entrada +1 ó -1 (exitatorio e inhíbitorio), estas entradas toman su valor de acuerdo a que reciban o no una señal del exterior, esta señal puede ser 0 si reciben un 1 toman el valor 1 ó -1 según sean excitatorias o inhibitorias, la Red Neuronal suma todos los valores que entran en forma algebraica y el resultado lo compara con un umbral o valor mínimo del nodo, si la suma es mayor que el umbral entonces, del nodo sale un 1 que va a excitar o inhibir a otros nodos, si la suma es menor que el umbral entonces se manda un cero que al llegar a otros nodos los deja apagados.

La ventaja de un sistema de este tipo se encuentra en que se le puede “enseñar” a distinguir entre diferentes entradas al sistema ya que si se quiere que a ciertas señales de entrada responda con una señal específica de salida se le da la señal de entrada y si no sale la señal esperada,

se le modifican algún o algunos umbrales para que se vuelva a enviar la señal repitiéndose el proceso hasta que aprende a reconocer la señal de entrada, este método se puede utilizar para enseñarle a responder a diferentes señales de alguna forma específica y aún mas logrando que aprenda a distinguir entre diferentes tipos de entrada.

En el proceso anterior lo que se ha hecho es lograr que la Red Neuronal represente el conocimiento que se permite distinguir entre diferentes entradas, en el sistema ya que no existe una caja específica para cada entrada, sino que el conocimiento que permite distinguir entre las entradas esta dado por la configuración de la Red Neuronal y los valores del umbral.

Esta forma de representar el conocimiento se basa en un modelo del cerebro conocido como Redes Neuronales o de Mc. Culott-Pitts y este modelo del cerebro aunque burdo se acerca al funcionamiento de un cerebro humano.

Por otro lado, en la Red Neuronal cada nodo es un elemento muy sencillo y la fuerza la da la complejidad de las relaciones entre los elementos del sistema, ya que el conocimiento emerge del sistema, con lo que si se pueden tener nodos mas complejos y sistemas que permitan múltiples relaciones entre los nodos, la relevancia de la Red Neuronal o de mecanismos desarrollados sobre su línea esta asegurada. Actualmente están adquiriendo gran importancia los estudios sobre procesamiento en paralelo, procesamiento distribuido y tejidos de computadora.

Donde precisamente se empieza a trabajar sobre múltiples nodos (procesadores, computadoras, etc.) interrelacionados, sin embargo se ha seguido con un enfoque tradicional de representar el conocimiento ya que el conocimiento se almacena en puntos bien delimitados, de donde, la idea sería que se buscara almacenar el conocimiento en todo el tejido y no en un punto concreto.

Por ejemplo se tienen algoritmos que ordenan datos o calculan operaciones matriciales distribuyendo el problema entre los nodos, en este caso se tiene que el conocimiento del método de ordenamiento o de solución matricial se encuentra distribuido pero no emerge del sistema ya que existen puntos específicos para la función, por lo que a pesar de ser ya un mecanismo potente para representar conocimiento y de que ya existen Bases de Datos Distribuidas y seria conveniente contar con Bases de Conocimiento Distribuidas es necesario pensar en la manera de representar el conocimiento difundo en todo el sistema, para aprovechar las propiedades emergentes de sistemas de tanta complejidad como son por ejemplo los tejidos de computadoras.

Conclusiones

En este documento se presentaron algunas de las formas con que se cuenta para representar el conocimiento buscando enfatizar los puntos siguientes:

- 1) El concepto de conocimiento, se ha atacado desde múltiples enfoques y es conveniente tomarlos en cuenta ya que dependiendo del nivel de conocimiento que se maneja existe uno u otro mecanismo para representarlo.

- 2) El hecho de que un conocimiento no tenga un “nivel alto” (por ejemplo, semántico, o de síntesis) no indica que no sea conocimiento y el excluirlo o no tomarlo en cuenta, podría crear una barrera sobre los mecanismos de representación que se consideran “convenientes” de estudio.
- 3) El ser humano siempre ha buscado formas físicas (dibujos, letras, esculturas, etc.) de representar el conocimiento y actualmente cuenta con un cúmulo muy grande de estas formas.
- 4) Al aumentar la complejidad del conocimiento se obtiene mecanismos mas compactos para representarlo mediante un proceso de abstracción que se “olvida” de los conocimientos específicos y busca reglas generales.
- 5) Actualmente en la computación se cuenta con múltiples mecanismos para representar el conocimiento desde nivel de datos hasta relaciones semánticas y pragmáticas.
- 6) Es conveniente combinar los diferentes mecanismos con que se cuenta buscando el apoyo mutuo dependiendo de sus características ya que por ejemplo en las bases de datos se pueden almacenar las características léxicas y sintácticas de un universo conceptual y en las bases de conocimiento rutinas semánticas y álgebra cualitativa las características semánticas y pragmáticas, por lo que la combinación de estas herramientas aumenta la fuerza del sistema.
- 7) Las herramientas actuales cubren un rango muy amplio de representación, sin embargo, no tienen la capacidad de abstraer y compactar el conocimiento en reglas generales a partir de reglas o puntos específicos.
- 8) Una forma poco explotada de representación de conocimiento se presenta cuando este se difunde en todo el sistema y no se queda en un punto dado.
- 9) Con el surgimiento del procesamiento en paralelo y las redes y tejidos de computadoras se ve posible el crear sistemas en los que el conocimiento se encuentre difundido en toda la estructura.

Como se puede observar existen muchos puntos sueltos en los mecanismos de representación del conocimiento y por tal motivo las posibilidades de desarrollo en el área son muy grandes y en su momento se podría llegar a una forma más general de representación en la cual se integren todos los puntos mencionados.

Bibliografía

- [1] Bloom, Benjamin, et. Al., Taxonomía de los objetivos de la educación, editorial el ateneo, Buenos Aires Argentina, 1971.
- [2] Galindo Soria, Fernando, Comentarios sobre la informatización del derecho en Memoria del 1er congreso iberoamericano de Informática Jurídica, Santo Domingo República Dominicana, 1984.
- [3] Guzmán Arenas, Adolfo, Sistemas Expertos, Fundamentos y aplicaciones inmediatas en Revista Compumundo, vol. 1, no. 5, agosto 1984.
- [4] Guzmán Arenas, Adolfo, Inteligencia Artificial Aplicada en Revista Compumundo, vol. 1, no. 6, septiembre 1984.
- [5] Negrete, José, Apropiación del conocimiento e inteligencia Artificial (partes I-VII), en revista 010, diciembre 1983-junio 1984.
- [6] ____, ____, revista 010, marzo 1985.
- [7] Ayala, G., Representación de conocimiento, en memoria de la 1ra conferencia internacional “las computadoras en instituciones de educación superior”, México, D.F. 1985
- [8] Colme Rauer, Alain, PROLOG, Lenguaje de la Inteligencia Artificial, vol. 4, num. 41
- [9] Minsky, Marvin, Papert, Seymour, PERCEPTRONS, An introduction to computational geometry.
- [10] Kleene, S.C., Representation of events in nerve nets and finite automata, Automata Studies.
- [11] Minsky, Marvin L., Computation, Finite and Infinite Machines, Prentice Hall
- [12] Hofstadter, Douglas R., Gödel, Escher, Bach: Una eterna trenza dorada, ed. Conacyt
- [13] Quezada, J. Daniel, La Lingüística Generativo-Transformacional supuestos e implicaciones, ed. Alianza Universidad.
- [14] Bach, Emmon, Teoría Sintáctica, Editorial Anagrama.
- [15] Serrano, Sebastián, Elementos de Lingüística Matemática, Editorial Anagrama.

VII.2 Sistema Evolutivo Constructor de Sistemas Expertos

Javier Ortiz Hernández²

Fernando Galindo Soria³

Introducción

En este trabajo se presenta el uso de los sistemas evolutivos para construir sistemas expertos en forma automática, entendiéndose dicha aplicación, como un ejemplo de los sistemas evolutivos aplicados en el campo de la informática.

Para este efecto se describe en forma general la idea de construir sistemas que en lugar de tener con antelación a su explotación y ya debidamente predefinidos los datos, procesos y estructuras de control, tengan los mecanismos que les permitan la adquisición y manejo del conocimiento de una manera natural y en el lenguaje del usuario.

Es decir, con capacidad de aprender en el lenguaje del usuario, interpretando, almacenando y relacionando la información con el propósito de generar nuevo conocimiento, mismo que utiliza para comunicarse de nueva cuenta con el usuario, en un proceso dialógico que permite el consecuente enriquecimiento de los sistemas.

I Sistemas Expertos y de Toma de Decisiones

Para iniciar este tema, es necesario definir antes los conceptos de experto y sistema experto:

- 1) *Experto*. Persona de vasta experiencia y con un conocimiento detallado y especializado de los problemas que se manejan en una área determinada.
- 2) *Sistema Experto*. Programa que usa la experiencia y el conocimiento de los expertos para resolver problemas.

Llegado a este punto es claro suponer que un sistema experto es entonces capaz de “equipararse” a un experto en persona, claro con todas las posibles e imaginables deficiencias que aquel o este pudieran tener. De allí, que el hecho de desarrollar sistemas expertos debe considerar criterios de eficiencia y de costo-beneficio en relación a su homólogo.

² Javier Ortiz Hernández, Centro Nacional de Investigación y Desarrollo Tecnológico

³ Fernando Galindo Soria, UPIICSA del I P N.

I Partes Componentes de un Sistema Experto Tradicional

Fundamentalmente un sistema experto se encuentra constituido por:

- 1) *Una Base de Conocimientos*. Se conforma utilizando un conjunto de técnicas de representación de conocimiento, que pueden utilizarse solas o combinadas de acuerdo al caso, ya que cada una de ellas tiene ciertos beneficios que ofrecer. Dos requisitos muy importantes son: modularidad en el manejo de la información y la capacidad de asociación de la información con su significado.
- 2) *Una Máquina de inferencia*. Se construye con mecanismos más o menos complejos de manejo de información, basados principalmente en la teoría de resolución de problemas, particularmente en la teoría de búsqueda en un espacio de estados.

Los sistemas expertos pueden dividirse en tres tipos principales, de acuerdo a la naturaleza de los problemas que tratan:

- 1) *Sistemas Expertos de Diagnóstico*. El núcleo de este sistema es la máquina de inferencia. Utilizando tanto la base de conocimientos como los datos particulares, la máquina de inferencia considera cada una de las entradas (síntomas) y consecuentemente evalúa y reevalúa las hipótesis pertinentes según el caso. Así dicho sistema es “hecho a la medida” para una área de aplicación específica, y el diagnóstico y acciones sugeridos por el sistema se obtienen paso a paso.
- 2) *Sistemas Expertos de Planeación*. Son utilizados para resolver problemas en donde tanto el estado inicial como el final (o deseado) son conocidos, y el sistema tiene determinar la secuencia de acciones necesarias para lograr los objetivos y metas previstas.
- 3) *Sistemas Expertos de Carácter Mixto*.

II Introducción a los Sistemas Evolutivos

Un sistema evolutivo construye su versión de la realidad directamente de las fuentes de información, utilizando para el caso, el mismo lenguaje de comunicación para captar y construir una representación propia de las estructuras del objeto de conocimiento. Más aún, es capaz de interactuar con su entorno, propiciando un diálogo que procure el crecimiento intelectual del sistema.

En una aplicación práctica es capaz de comprender un lenguaje de comunicación a partir de ejemplos, y al mismo tiempo construir su base de conocimiento; proceso que le permite poco a poco aumentar su conocimiento del lenguaje, manejar eficientemente su propio conocimiento e interactuar con el usuario del sistema.

III Arquitectura de un Sistema Evolutivo

Para llevar a cabo su propósito un sistema evolutivo debe contar más que con conocimientos específicos, o estructuras específicas (datos, acciones, control) con mecanismos que le permitan construir sus propias estructuras en forma relativamente sencilla.

En términos generales, es un mecanismo inductivo-deductivo, en contraste con los que tradicionalmente se han utilizado (mecanismos deductivos) para la resolución de problemas independientemente de su complejidad y de la complejidad de dichos mecanismos (manejadores de bases de datos, nómina, contabilidad, sistemas expertos, etc.). Estos mecanismos deductivos, requieren que antes que sea utilizado el sistema, contenga todas las reglas aplicables al caso para el manejo de los datos y procesos, llámense programas, esquemas de base de datos, máquinas de inferencia, etc., convirtiendo a dichos sistemas en estructuras rígidas incapaces de adaptarse dinámicamente a los nuevos requerimientos de los usuarios y finalmente en artículos de consumo.

Los sistemas evolutivos tienen integrados en un solo mecanismo la captación, estructuración y el proceso de la información. Es decir cuentan con una interfase que interpreta directamente en términos de datos, procesos y estructuras, y otra interfase que da respuesta a alguna cuestión utilizando el conocimiento adquirido.

IV Sistemas Evolutivos Generadores de Sistemas Expertos

Una de las características de los sistemas de diagnóstico y toma de decisiones (sistemas de soporte a las decisiones, reconocedores de patrones, etc.) en general y de los sistemas expertos en particular se encuentra en que el lenguaje del usuario consta principalmente de oraciones de la forma:

SÍNTOMAS : DIAGNÓSTICO : ACCIONES o TRATAMIENTO

donde este tipo de oraciones nos permiten plantear por ejemplo la base de un conjunto de reglas de inferencia o el patrón general de un cierto tipo de figuras.

A partir de lo anterior, el problema de construir un sistema experto, o en general de toma de decisiones mediante un sistema evolutivo, es decir, la construcción automática de sistemas expertos se vuelve un problema relativamente fácil de atacar y específicamente la etapa de ingeniería de conocimiento se puede automatizar en buena medida, ya que el problema clásico de encontrar las reglas de inferencia desaparece y es sustituido por el problema más general de encontrar un conjunto de ejemplos significativos de diagnósticos o de toma de decisiones reales, en los cuales se encuentren precisamente inmersos los síntomas, diagnósticos y acciones, de donde el sistema evolutivo queda como se ve en la figura 1.

Supóngase que se tiene el siguiente conjunto de ejemplos:

a b c d e f	: D ₁ : T ₁ T ₂ T ₃
a b x y c m	: D ₂ : T ₂ T ₄
a b k l	: D ₁ : T ₁ T ₂ T ₆
d m l f g	: D ₄ : T ₆
a b k o	: D ₂ : T ₇ T ₈

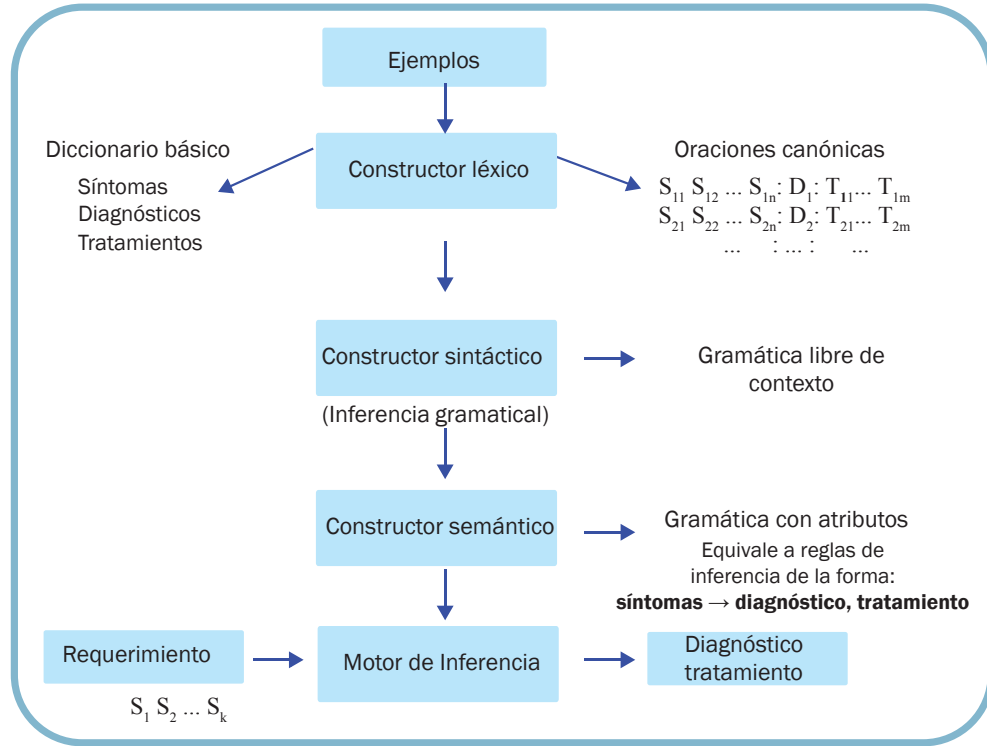


Figura 1. Sistema Evolutivo Constructor de Sistemas Expertos.

Se ordenan los síntomas en cada oración y se genera la gramática canónica:

$$\begin{aligned}
 S &\rightarrow a b c d e f : D_1 : T_1 T_2 T_3 \mid \\
 &a b c m x y : D_2 : T_2 T_4 \mid \\
 &a b k l : D_1 : T_1 T_2 T_6 \mid \\
 &d f g l m : D_4 : T_6 \mid \\
 &a b k o : D_2 : T_7 T_8
 \end{aligned}$$

Se ordenan las oraciones según sus síntomas:

$$\begin{aligned}
 S &\rightarrow a b c d e f : D_1 : T_1 T_2 T_3 \mid \\
 &a b c m x y : D_2 : T_2 T_4 \mid \\
 &a b k l : D_1 : T_1 T_2 T_6 \mid \\
 &a b k o : D_2 : T_7 T_8 \mid \\
 &d f g l m : D_4 : T_6
 \end{aligned}$$

Se factoriza la parte de los síntomas:

$$\begin{aligned}
 S &\rightarrow a b X \mid d m \mid f g : D_4 : T_6 \\
 X &\rightarrow c d e f : D_1 : T_1 T_2 T_3 \\
 &c m x y : D_2 : T_2 T_4 \\
 &k l : D_1 : T_1 T_2 T_6 \\
 &k o : D_2 : T_7 T_8
 \end{aligned}$$

Se continúa factorizando:

$$\begin{aligned} S &\rightarrow a \ b \ X \ d \ m \ l \ f \ g : D_4 : T_6 \\ X &\rightarrow c \ Y \ l \ k \ Z \\ Y &\rightarrow d \ e \ f : D_1 : T_1 \ T_2 \ T_3 \mid m \ x \ y : D_2 : T_2 \ T_4 \\ Z &\rightarrow l : D_1 : T_1 \ T_2 \ T_6 \mid o : D_2 : T_7 \ T_8 \end{aligned}$$

En este momento ya se tienen prácticamente las reglas de inferencia, sin embargo, se continúa ahora trabajando sobre los diagnósticos.

$$\begin{aligned} S &\rightarrow a \ b \ X \mid d \ m \ l \ f \ g : A_1 \\ X &\rightarrow c \ Y \mid k \ Z \\ Y &\rightarrow d \ e \ f : A_2 \mid m \ x \ y : A_3 \\ Z &\rightarrow l : A_2 \mid o : A_3 \\ A_1 &\rightarrow D_4 : T_6 \\ A_2 &\rightarrow D_1 : T_1 \ T_2 \ T_3 \mid : D_1 : T_1 \ T_2 \ T_6 \\ A_3 &\rightarrow D_2 : T_2 \ T_4 \mid D_2 : T_7 \ T_8 \end{aligned}$$

Se factorizan los diagnósticos:

$$\begin{aligned} S &\rightarrow a \ b \ X \mid d \ m \ l \ f \ g : A_1 \\ X &\rightarrow c \ Y \mid k \ Z \\ Y &\rightarrow d \ e \ f : A_2 \mid m \ x \ y : A_3 \\ Z &\rightarrow l : A_2 \mid o : A_3 \\ A_1 &\rightarrow D_4 : T_6 \\ A_2 &\rightarrow D_1 : T_1 \ T_2 \ B_1 \\ A_3 &\rightarrow D_2 : B_3 \\ B_1 &\rightarrow T_3 \mid T_6 \\ B_3 &\rightarrow T_2 \ T_4 \mid T_7 \ T_8 \end{aligned}$$

En este punto se termina la etapa de inferencia y se integran las rutinas semánticas del sistema:

$$\begin{aligned} S &\rightarrow a \ b \ X \mid d \ m \ l \ f \ g : A_1 \\ X &\rightarrow c \ Y \mid k \ Z \\ Y &\rightarrow d \ e \ f : A_2 \mid m \ x \ y : A_3 \\ Z &\rightarrow l : A_2 \mid o : A_3 \\ A_1 &\rightarrow S_1 \ D_4 : S_2 \ T_6 \\ A_2 &\rightarrow S_1 \ D_1 : S_2 \ T_1 \ T_2 \ S_3 \ B_1 \\ A_3 &\rightarrow S_1 \ D_2 : S_3 \ B_3 \\ B_1 &\rightarrow T_3 \mid T_6 \\ B_3 &\rightarrow T_2 \ T_4 \mid T_7 \ T_8 \\ S_1 &\rightarrow \text{imprime "EL DIAGNÓSTICO ES: "} \\ S_2 &\rightarrow \text{imprime "EL TRATAMIENTO ES: "} \\ S_3 &\rightarrow \text{imprime "EL TRATAMIENTO PUEDE SER CUALQUIERA DE LOS SIGUIENTES: "} \end{aligned}$$

Obtener el sistema experto a partir de la gramática anterior es trivial, ya que si por ejemplo, la representamos como un diagrama de Warnier, se tiene lo mostrado en la figura 2, y se aprovecha que el paso a un programa es directo.

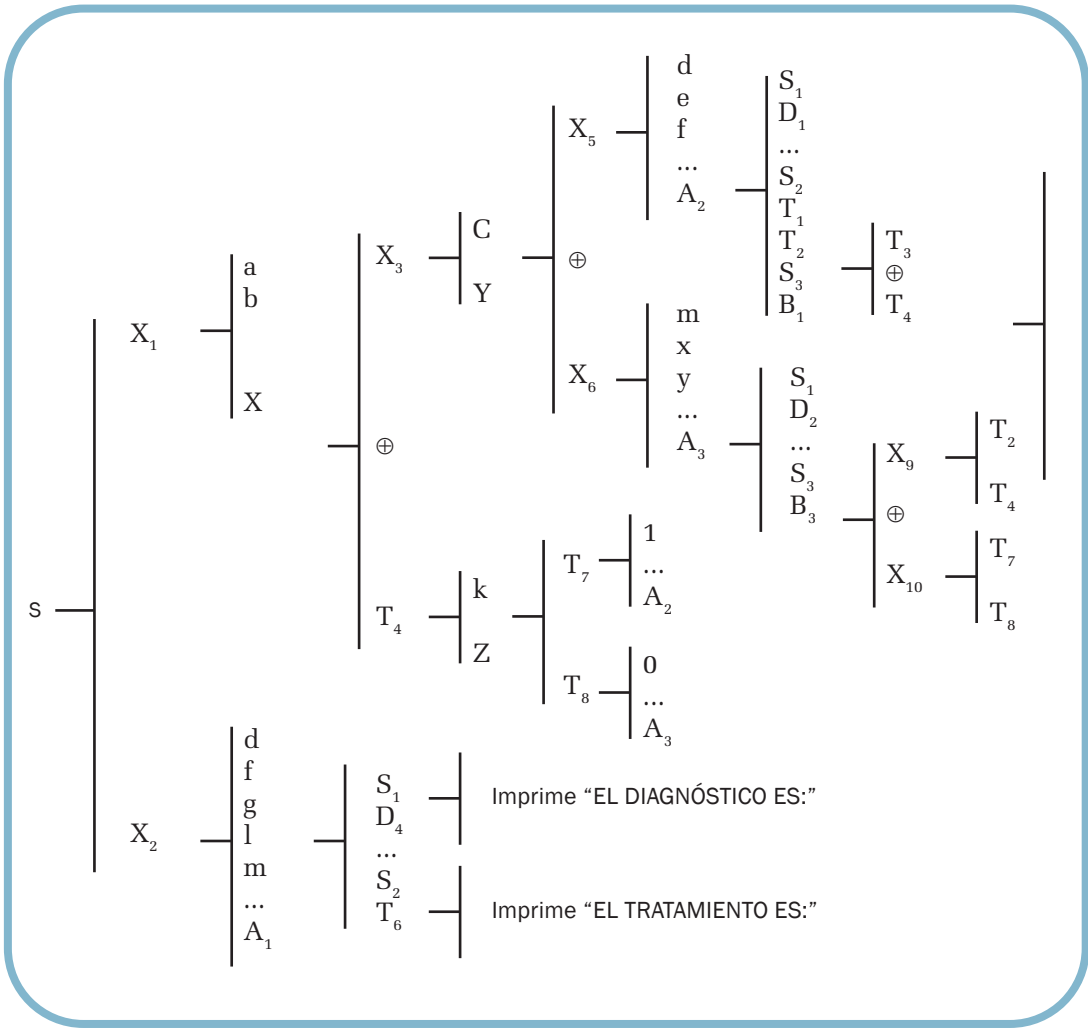


Figura 2. Sistema Experto obtenido y representado como un diagrama de Warnier.

Conclusiones

El paradigma de los Sistemas Evolutivos nos facilita la construcción de sistemas expertos como se ha visto en este trabajo.

Se presentó la manera de utilizar la inferencia gramatical para que a partir de un conjunto de ejemplos significativos del área se pueda obtener directamente las reglas de un sistema experto y luego de estas es directo pasar a un programa.

VII.3 Biblioteca Automatizada

Fernando Galindo Soria⁴

Introducción

Cuando se observan los trabajos desarrollados en el área de Automatización de Bibliotecas, se detecta que la mayoría de estos trabajos versan realmente sobre la automatización de un catálogo de títulos, autores, etc., y sobre la explotación automática de estos catálogos y con esto el concepto de Biblioteca Automatizada se queda en el limbo, ya que lo que realmente se automatiza es la parte administrativa-operativa del manejo de información bibliográfica y la biblioteca en sí no es tocada.

En este trabajo se presenta una propuesta de automatización de bibliotecas, en la cual se plantea que la biblioteca debe estar dentro de la computadora y no en estantes o libreros. Esta idea no es nueva y realmente en la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) del Instituto Politécnico Nacional (IPN) se han desarrollado algunos prototipos bajo este concepto, sin embargo, la mayoría de las personas involucradas en automatización de bibliotecas no la contemplan principalmente por dos motivos:

- 1) Rompe completamente con el esquema tradicional de una biblioteca, donde los libros se encuentran en libreros y se tiene un sistema de alta complejidad para localizarlos y mantenerlos en su lugar (es por eso, que la mayoría de aplicaciones de automatización de bibliotecas tienden a automatizar el proceso de catalogación).
- 2) *No es fácil conceptualizar a una biblioteca dentro de la computadora* y aún más se marcan una serie de problemas para su logro. Un libro ocupa mucho espacio y no cabría en la computadora, esto ya no es un argumento si se considera que un libro típico tiene 150 páginas, 40 renglones por página, 50 caracteres/renglón = 300,000 caracteres lo cual cabe sin ningún problema por ejemplo en un disco flexible. Por lo que si tomamos en cuenta que una computadora típica actual puede tener 2 Mb en memoria principal y 1000 Mb en disco, la capacidad de almacenamiento es sin problema la de una biblioteca mediana (alrededor de 3000 títulos diferentes).

⁴ Fernando Galindo Soria IPN-UPIICSA.

Otro problema que se marca mucho es el de la captación de la información, sin embargo, este problema es en sí una falacia, ya que la mayoría de las editoriales actuales capturan para su edición los libros en algún sistema de computo y posteriormente lo imprimen en papel (en la actualidad sale tan caro un ciento de hojas de papel como un disco flexible) por lo que en su momento se podría almacenar directamente el libro en la computadora, por otro lado, en el área de reconocimiento de imágenes se tienen desarrollos orientados a la “captura visual” de libros.

Poca gente tendría acceso a este tipo bibliotecas, esta es una verdad/mentira, ya que en la actualidad poca gente tiene un acceso real a cualquier tipo, de biblioteca y los libros son cada vez más caros por otra parte con introducción de las computadoras en las bibliotecas públicas (Proyecto Computación para Todos los Niños) y en las escuelas por un lado y por el otro con el acceso casero a la computadora, en algunos lugares, cada vez más se contará con herramientas para un acceso masivo.

Sin embargo, el argumento más fuerte en contra de este tipo de desarrollo se centra en la no disponibilidad de herramientas de programación orientadas a manejar una biblioteca automatizada. Es precisamente el objetivo de este trabajo, mostrar que una herramienta de este tipo es relativamente fácil de construir ya que se cuenta prácticamente con todos los ingredientes del menú y lo que se requiere es combinarlos en un buen platillo.

I Arquitectura del Sistema

Al preguntarle a varias personas cómo se imaginan la interacción con una biblioteca automatizada, se ha llegado a la conclusión de que el diálogo con esta debe ser en Español y contener oraciones como:

- 1) *Dame los libros de Fisicoquímica*
- 2) *Mira máquina quiero que me des la lista de todos los libros que escribió Cervantes*
- 3) *Imprime el contenido de “Suave Patria”*
- 4) *Dame tres temas relacionados con Histología*
- 5) *Dame los principales temas de Topología*
- 6) *Dame los fragmentos del libro de Biología Celular donde hablan de los átomos*
- 7) *Dame todos los fragmentos que tengas sobre Cuernavaca.*
- 8) *Dame diez fragmentos de temas relacionados con Pulsares*

Como se ve, este tipo de peticiones al sistema cubren una gama muy amplia y van desde un simple listado de catálogo hasta un acceso a bases de conocimiento con el fin de encontrar los temas relacionados con un tema dado y posteriormente un acceso a base de datos y la Biblioteca con el fin de acceder sólo los fragmentos solicitados.

En general la arquitectura de un Sistema de Biblioteca Automatizada capaz de soportar una interacción, en Lenguaje Natural y almacenar la biblioteca dentro del sistema es la que se muestra en la Figura 1.

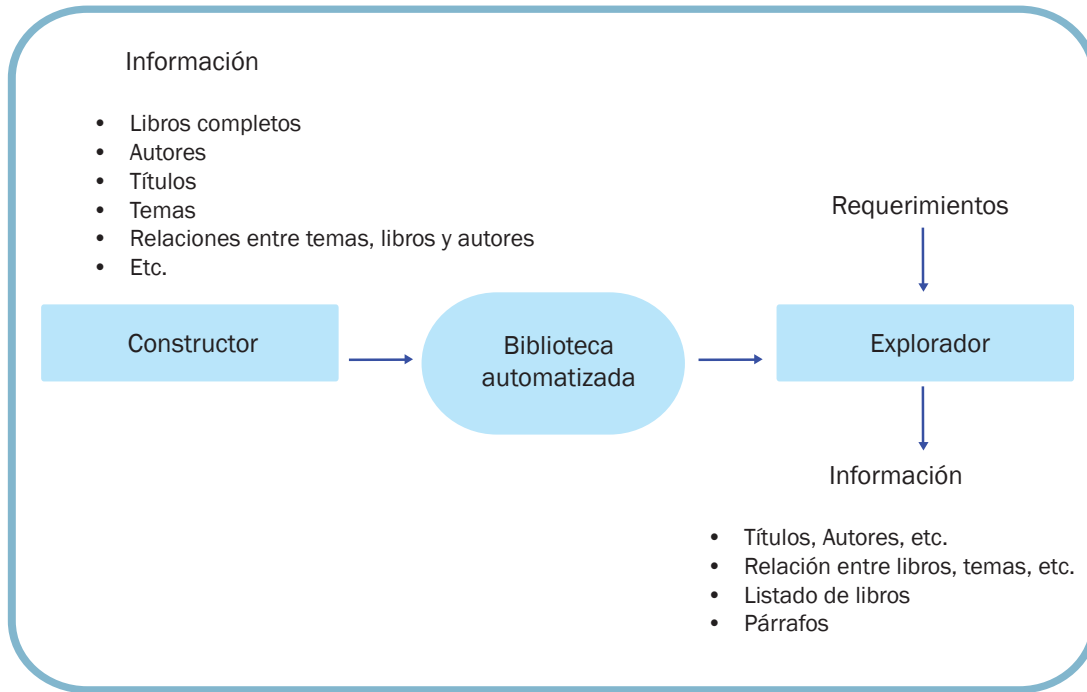


Figura 1. Componentes de un Sistema de Biblioteca Automatizada.

El esquema de la Biblioteca Automatizada se muestra en la figura 2.

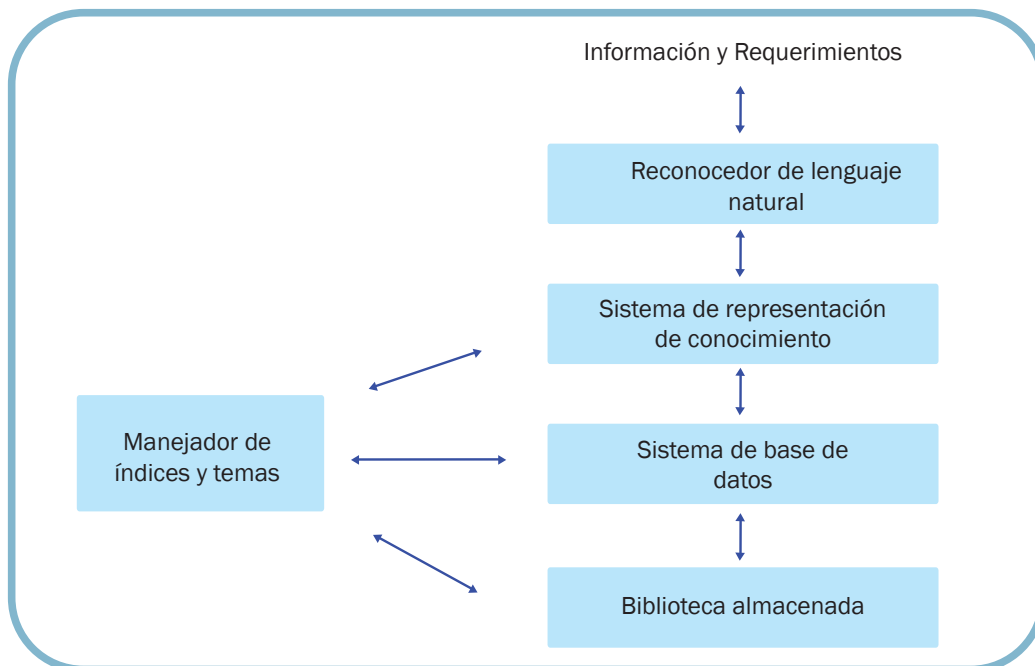


Figura 2. Esquema de la Biblioteca Automatizada.

II Reconocedor de Lenguaje

El reconocedor de Lenguaje Natural, es el responsable de detectar a partir de la información ó requerimientos del usuario, los entes y acciones involucradas.

Por ejemplo en la oración:

Dame todos los párrafos que tengas sobre temas relacionados con Antropología.

- 1) El sistema detecta una acción: **Dame**
- 2) Detecta un cuantificador: **todos**
- 3) Detecta lo que se quiere (párrafos, libro, área, etc.): **párrafos**
- 4) Detecta que no quiere el tema principal sino: **los temas relacionados**
- 5) Detecta el tema principal: **Antropología**

Este tipo de reconocedor es relativamente fácil de construir utilizando las técnicas de Programación Dirigida por Sintaxis.

III Sistema de Representación de Conocimiento

Ya que se detectaron los entes y acciones involucradas y para dar respuesta al requerimiento se interconectan tres niveles de almacenamiento de la información:

- 1) *El primero de ellos o Sistema de Representación de Conocimiento*, se tiene un mecanismo que nos permite almacenar conocimiento de la biblioteca a nivel de hechos y reglas de inferencia.

Para aclarar el punto, de las oraciones que siguen, se construye una red semántica que se muestra en la figura 3: La Citología está relacionada con la Histología. La Histología estudia los tejidos. La Citología estudia las células. Los tejidos están compuestos de células. La Histología es parte de la Biología. Biología Celular es un libro de Biología. C.U.M. Smith es el autor de Biología Celular.

Por lo que si alguien pregunta:

Dame un libro de Biología

el sistema puede responder: **Biología Celular**

Con la pregunta:

Dame un tema relacionado con Citología

el sistema puede responder: **Histología.**

Por otra parte si a lo anterior se le aumentan reglas de inferencia, se cuenta ya con una herramienta muy interesante.

Por ejemplo, si se agrega:

Si X es parte de Y, X estudia Z, entonces Y estudia Z

Si X está relacionado con Y entonces Y está relacionado con X

Si X estudia y entonces X está relacionado con Y

Si X está compuesto de Y entonces X está relacionado con Y

Si X es parte de Y entonces X está relacionado con Y

Por lo tanto, la pregunta:

Dime los temas relacionados con Biología

da como respuesta.

Histología

Tejidos

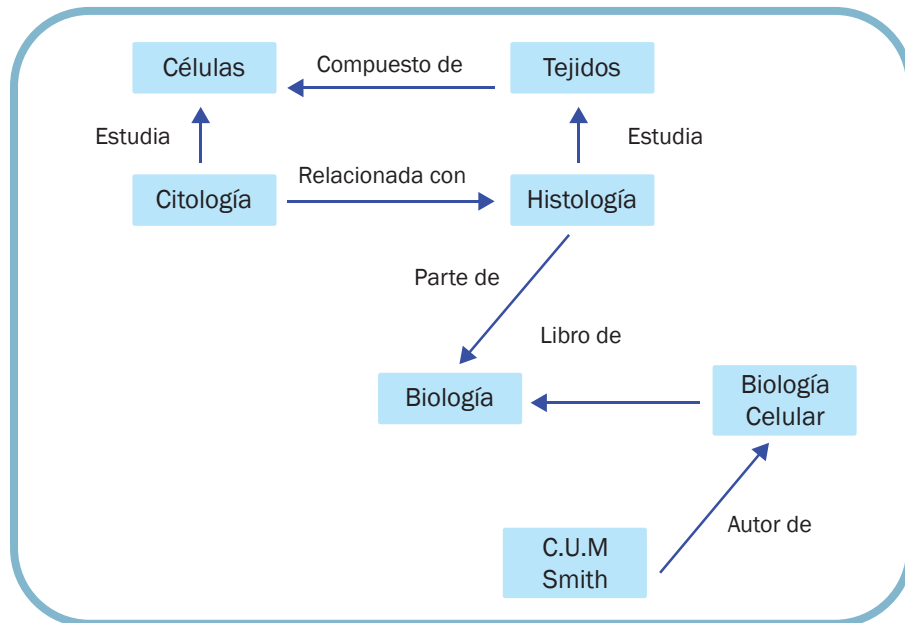


Figura 3. Una Red Semántica.

IV Base de Datos

Este primer nivel de almacenamiento es extremadamente poderoso y en su momento nos permite dar respuestas a múltiples interrogantes en las que participan distintos entes involucrados.

Sin embargo, y si recordamos el modelo de Chen de Base de Datos (de Entidad/Relación), la realidad se puede conceptualizar en términos de entidades relacionadas entre sí, donde cada entidad es a su vez descrita mediante un conjunto de atributos y hasta el momento, lo único que hemos manejado ha sido a las entidades y sus relaciones y no se ha involucrado a los atributos, por lo que oraciones como las siguientes:

- 1) *Pedro Pérez nació en la ciudad de Guadalajara*
- 2) *Pedro Pérez vive en Cerritos #38, col. Chapulín*
- 3) *Pedro Pérez nació el 20 de enero de 1950*

Como un primer enfoque “se siente” que se podría aumentar la Base de Conocimientos, sin embargo, desde un punto de vista práctico, tal vez no sea la mejor solución, ya que los atributos de un ente pueden ser muchos y en su momento sólo se refieren a ese ente en particular por lo que no se requiere ni la capacidad de relacionar todo contra todo, ni los mecanismos de inferencia.

En particular, en este trabajo se plantea la introducción de un segundo mecanismo de manejo de información fuertemente relacionado con el primero.

Este mecanismo es un sistema manejador de Base de Datos de tipo Relacional.

Mediante este enfoque, se detectan los principales tipos de Entes involucrados en el sistema (en este caso pueden ser: autor, libro, editorial, etc.) y para cada caso, sus principales atributos, por ejemplo en el caso de autor pueden ser: nombre, fecha de nacimiento, lugar de nacimiento, estudios, etc.; y a partir de estos se puede diseñar la Base de Datos (figura 4), en donde a cada entidad le correspondería una parte del esquema.

AUTOR

Nombre	Fecha-Naci	Lugar-Nac
Juan Pérez	20-ene-50	Guadalajara

LIBROS

Autor	Título	Edición	Páginas

EDITORIAL

Nombre	Dirección	Teléfono	Director

Figura 4. Ejemplo de la Base de Datos.

Mediante este mecanismo, los atributos de Juan Pérez quedan en la Base de Datos, por lo que si alguien pregunta:

¿Dónde nació Juan Pérez?

El sistema responde: **En Guadalajara**

Por otra parte, los dos mecanismos se encuentran fuertemente relacionados, ya que si por ejemplo se almacena:

Juan Pérez escribió “La Vida es Verde”

esta información se almacena en la Base de Conocimientos y si más adelante alguien pregunta:

1) *¿Quién escribió La Vida es Verde?*

el sistema contesta: **Juan Pérez** (tomándolo de la Base de Conocimientos).

2) *¿Dónde nació el autor de “La Vida es Verde”*

En este caso el sistema primero tiene que ir a la Base de Conocimientos para enterarse que Juan Pérez escribió “La Vida es Verde” y con esta información accesa la Base de Datos para saber que Juan Pérez nació **en Guadalajara**, que es lo que responde.

V Biblioteca Almacenada

Sin embargo, y a pesar de estos mecanismos, aún no se cuenta con una Biblioteca Automatizada, ya que los libros aún no están en el sistema pero esto se puede arreglar introduciendo el tercer mecanismo, con el cual se pretende almacenar precisamente los libros.

Este mecanismo es un almacén o Biblioteca Almacenada donde cada libro queda físicamente guardado.

Por lo que si alguna persona pide el contenido de un libro en particular, lo único que se requiere es desplegarlo.

Ahora bien, para lograr lo anterior, nuevamente los diferentes mecanismos deben estar fuertemente relacionados.

Por ejemplo, almacenado como atributos del libro dentro de la Base de Datos los siguientes:

1) *Nombre del Archivo Almacén*

2) *Posición Inicial del libro en el archivo*

3) *Posición Final de caracteres del libro*

con lo cual, si se pide un libro en particular se puede localizar.

Hasta este punto se puede decir que ya se cuenta con una Biblioteca Automatizada, sin embargo, es ilógico desaprovechar tan bonita oportunidad y quedarnos sólo en el umbral de lo que se puede obtener ya que, sí se involucra un nuevo mecanismo, el potencial del sistema aumenta considerablemente.

VI Manejador de Índices y Temas

Este mecanismo es un manejador de índices y temas.

Mediante este mecanismo se puede dar respuesta a planteamientos como los siguientes:

- 1) *Dime en que partes del libro de Biología Molecular se estudian las células*
- 2) *Dame los párrafos del libro Biología Molecular que tratan de las células*
- 3) *Dame todos los párrafos que tengas sobre Vicente Guerrero*

En principio un mecanismo de este tipo está formado por un directorio o índice, donde para cada tema aparecen los libros donde se encuentra y se indica en que partes del libro se menciona el tema.

Al integrar este mecanismo con los anteriores, el potencial crece ya que nos permite un acceso prácticamente a la información y realmente **el concepto de libro desaparece** ya que lo que se tiene es un gran almacén de información con múltiples mecanismos de acceso a diferentes niveles de especificidad.

Uno de los problemas planteados a este último mecanismo es el de la dificultad para obtener y almacenar este tipo de información, sin embargo, ésta no es una limitante real ya que la mayoría de los libros trae un índice por capítulos y muchos cuentan con índices por temas.

Por otro lado, esta es otra de las ventajas de una Biblioteca Automatizada, ya que al tener los libros en la computadora se puede desarrollar una herramienta que permita listar todas las palabras diferentes con las se cuenta y su posición en los diferentes documentos (en su momento se le puede dar una lista de palabras relevantes para que no las almacene, y otra de palabras compuestas v.g.: Buenos Aires que es, importante manejar en forma compuesta).

Conclusiones:

Memoria Automatizada

En el punto anterior se menciona que el potencial de la herramienta trasciende al de una Biblioteca Automatizada y a lo que realmente se llega es a un Mecanismo para manejar información masiva a múltiples niveles de especificidad.

Este es un punto sustancial del trabajo, ya que lo que se plantea es un mecanismo que lo mismo permite manejar una Biblioteca general, que una especializada (Médica, Legal, Matemática,

etc.) y con potencial de darnos acceso a la información específica, sin importar la fuente de acceso original, con lo que se cuenta con un mecanismo de representación de conocimiento capaz de entregarnos, por ejemplo, toda la información relacionada con cierto tipo de enfermedad, o también sobre un tema dado, con lo cual, el proceso tedioso de investigación documental se puede soslayar.

Aún requerimientos que se le hagan y no encuentre datos, los puede almacenar y solicitar a algún especialista la respuesta, que estaría disponible eventualmente.

En su extremo más interesante, un sistema de este tipo nos puede contestar todo lo que sabe sobre cualquier tema del que tenga información, independientemente de la fuente de consulta o de la fecha en que fue almacenada, ya que integra todo en una sola gran memoria (figura 5).

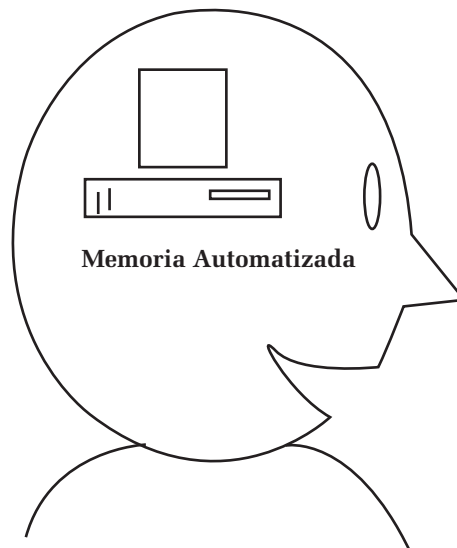


Figura 5. Conclusión: Memoria Automatizada

VII.4 Sistemas Evolutivos Basados en Conocimiento

Fernando Galindo Soria⁵

Los Sistemas Basados en Conocimiento son herramientas más generales que existen ya que por ejemplo un Sistema de Biblioteca Automatizada, un Sistema Experto o un Sistema Manejador de Base de Datos pueden verse como casos particulares de este tipo de sistemas. Sin embargo lo común es que se construyan sistemas particulares y no se ataque el caso general, por lo que, en este trabajo encontraremos como se puede construir una herramienta general para manejo de conocimiento soportado por un Sistema Evolutivo[1][2].

Como primer punto partimos de que en un Sistema Basado en Conocimiento se cuenta con diferentes mecanismos para representarlo y manejarlo, incluyendo:

- 1) *Directorios de datos*
- 2) *Archivos de datos*
- 3) *Bases de datos*
- 4) *Bases de conocimiento*
- 5) *Conjuntos de reglas de inferencia.*

y dependiendo del tipo de sistema, los medios de manejo de conocimiento se pueden Integrar de diferentes formas por ejemplo, en los sistemas de base de datos es común encontrar un directorio de datos interrelacionado con la base de datos, los sistemas expertos o de toma de decisiones integran al menos una base de conocimientos y un conjunto de reglas de inferencia, los sistemas de biblioteca automatizada están formados al menos por una combinación de bancos de información (donde están almacenados los documentos) y directorios de datos (que permiten acceder directamente a elementos específicos de los bancos de información).

I Arquitectura de Sistemas Evolutivos Basados en Conocimiento

La forma como se interrelacionan los elementos también depende de la aplicación, formando diferentes tipos de arquitectura, en particular en este trabajo partiremos de una arquitectura por capas. en la cual se tienen los diferentes componentes de un sistema basado en conoci-

⁵ Fernando Galindo Soria Instituto Politécnico Nacional UPIICSA ESCOM.

miento colocados en capas relacionadas entre si, de tal manera que una capa sirve para acceder o localizar conocimientos de otras como se ve en la figura 1.

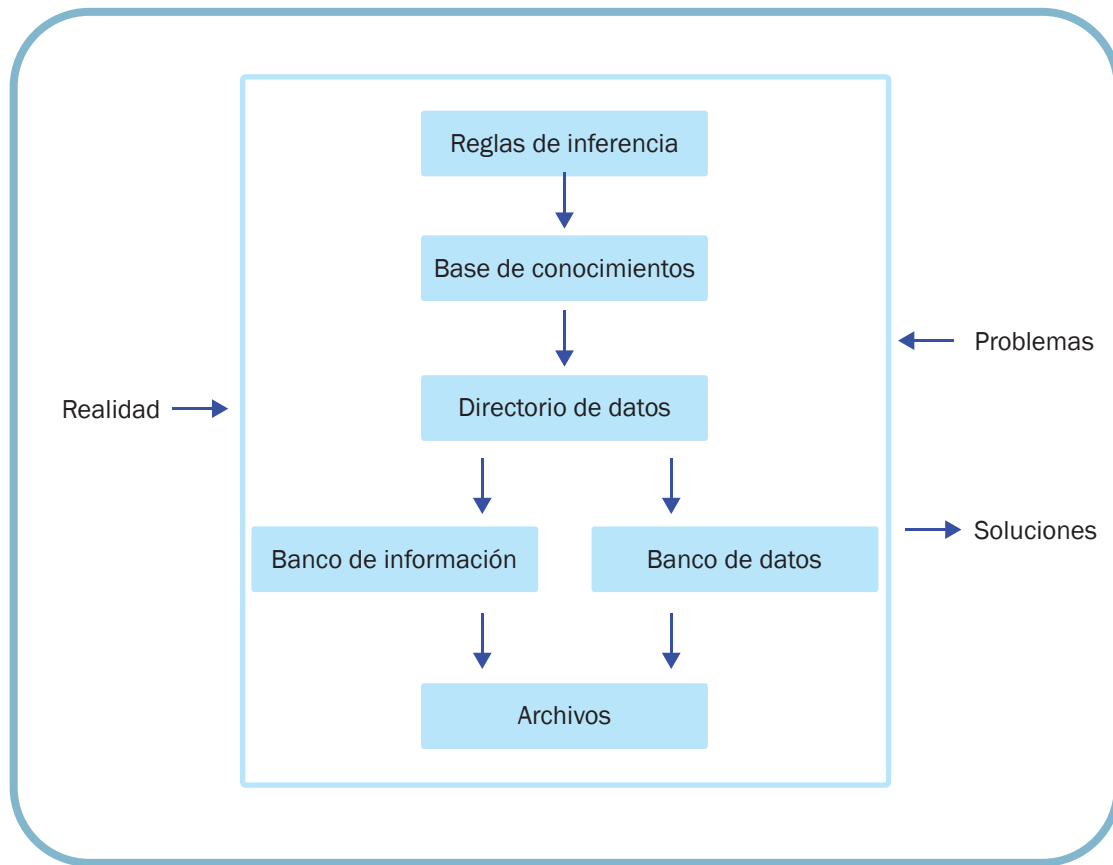


Figura 1. Arquitectura de los Sistemas Basados en Conocimiento.

Como se observa, el sistema está formado por un conjunto de mecanismos de representación de conocimiento relacionados entre sí y un conjunto de mecanismos de percepción y actuación que permiten interactuar con el sistema y mantener actualizada su información.

Sin embargo, la arquitectura no es tan simple ya que un problema muy grave de este tipo de sistemas se presenta en la adquisición, mantenimiento y explotación del conocimiento, ya que llega un momento en que el volumen de datos a manejar hace extremadamente complejo su tratamiento manual.

Por lo que, en este trabajo se plantea que el proceso de adquisición del conocimiento se debe realizar en forma automatizada, se propone un conjunto de herramientas evolutivas orientadas a la adquisición y mantenimiento del sistema basado en conocimiento y en los siguientes puntos se describirá como se puede construir un Sistema Evolutivo Basado en Conocimiento y bajo una arquitectura por capas.

II Generación Automática del Directorio de Datos

Por ejemplo la creación del directorio de datos en forma manual puede terminar siendo un proceso muy tedioso, ya que cada palabra clave en el directorio se necesita ubicar en los bancos de información para detectar en que párrafos se presenta, de tal forma que cuando un usuario pregunte por esa palabra el sistema le indique en que párrafos se encuentra.

Realizar lo anterior a mano puede ser un proceso aburrido o impráctico, por lo que desde hace muchos años se han desarrollado herramientas que construyen los directorios en forma automatizada.

Por ejemplo durante los años 70's en El Colegio de México se inicio el proyecto del Diccionario del Español en México donde se propuso hacer un diccionario con todas las palabras que utilizan los hispanohablantes en México, para realizar este proyecto se analizaron varios millones de documentos buscando las diferentes palabras y sus significados.

Definitivamente este proyecto no se hubiera podido realizar manualmente, ya que la cantidad de información a tratar era enorme, por lo que se construyó bajo la dirección del Ing. José Cen Zubieta un sistema de apoyo automatizado, al cual se le daban como entrada los documentos a analizar y el sistema generaba para cada palabra la lista de los párrafos donde aparecía esa palabra de tal forma que los lingüistas podían estudiar el uso de cada palabra en diferentes contextos y de ahí le asociaban su significado.

Basados en la idea anterior se puede construir una herramienta que analice los documentos almacenados en el banco de información y a partir de ahí la primera capa del sistema evolutivo toma los datos del banco y construye el directorio de datos, indicando para cada palabra clave la localización de los párrafos donde aparece.

III Generación Automática de la Base de Conocimientos

No sólo la creación del directorio de datos puede ser un problema oneroso, también la creación y mantenimiento de los otros componentes del sistema puede llegar a ser muy pesado para realizarse a mano, por lo que como siguiente capa del sistema se integrará un mecanismo que permita la construcción automática de la Base de Conocimientos.

La Base de Conocimientos es uno de los componentes más importantes de un sistema basado en conocimiento, ya que permite representar las interrelaciones entre las diferentes palabras clave y su significado.

Existen varios mecanismos para representar el conocimiento, pero uno de los más usados son las redes semánticas, donde una red semántica esta 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 **o r o**, 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 se realizó en la tesis de licenciatura de Jesús Olivares titulada “Sistema Evolutivo para Representación de Conocimiento” desarrollada en la UPIICSA del IPN [3], ahí se planteó un mecanismo evolutivo para construir un base de conocimientos basada en una Red Semántica Aumentada (que almacena datos, imágenes, procesos, sonido y otros elementos).

La idea de Jesús consistió básicamente en aplicar una serie de técnicas de Inferencia Gramatical basadas en la distribución lingüística [3][4], 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 **o r o** 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 aplicará 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	o1
estudia en el	es una relación	r1
Poli	es un objeto	o2
,	es un separador	+
trabaja en el	es una relación	r2
INEGI	es un objeto	o3
y	es un separador	+
vive en	es una relación	r3
Lindavista	es un objeto	o4

de donde la oración quede representada como:

$$o1(r1o2 + r2o3+r3o4)$$

El paréntesis que esta después de o1 indica que tiene relación con los elementos restantes de la oración (en la tesis de Elsa Berruecos[6] se presenta un método para encontrar en forma automática la dependencia entre los diferentes elementos de una oración).

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

$$o1 (r1o2 + r2o3 + r3o4) = o1r1o2 + o1r2o3 + o1r3o4$$

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

o1r1o2	Juan estudia en el Poli
+	
o1r2o3	Juan trabaja en el INEGI
+	
o1r3o4	Juan vive en Lindavista

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

IV Generación Automática de las Reglas de Inferencia

El conjunto de reglas de inferencia es otro de los componentes básicos de un Sistema Basado en Conocimiento nuevamente su generación manual puede ser muy tediosa por lo que nos basaremos en un trabajo desarrollado junto con Javier Ortiz en el CENIDET de Cuernavaca [5] para la generación automatizada de reglas de inferencia para lo cual partiremos que en general una regla de inferencia es un regla de la forma:

Si A entonces B

esto es, si se cumple A entonces B es cierto donde A en general es un conjunto de condiciones o síntomas sobre un problema y B en general se puede ver como una acción o conjunto de acciones que se ejecutan si se cumpla A, en particular B es el diagnóstico y tratamiento asociado con los síntomas de A.

Para encontrar las reglas de inferencia a partir de un documento se toma del documento un párrafo donde se presente un conjunto de síntomas asociados con un diagnóstico y con un conjunto de tratamientos, se sustituye cada elemento por una representación interna donde se indica si un elemento es síntoma, diagnóstico o tratamiento e ignorando los elemento que no tengan significado por lo que al final queda una representación de la forma:

S1 S2 S3 ... Sn D1 T1 T2 T3 ... Tm

Esto se realiza para todos los párrafos del documento y al final se cuenta con un conjunto de reglas como:

S11 S21 S31 ... Sn1 D1 T11 T21 ... Tm1

S12 S22 S32 ... Sn2 D2 T12 T22 ... Tm2

...

S1k S2k S3k ... Snk Dp T1k T2k ... Tmk

Ahora bien asociada con la distribución lingüística se encuentra otra operación algebraica conocida como factorización lingüística y que es inversa de la distribución o sea que si factorizamos o1 de la oración:

o1r1o2 + o1r2o3 + o1r3o4

nos queda:

o1 (r1o2 + r2o2 + r3o3)

Lo interesante es que si mediante la distribución se pueden encontrar los componentes de la base de conocimientos, mediante la Factorización Lingüística se pueden encontrar las reglas de Inferencia del Sistema Basado en Conocimiento, ya que sólo se necesitan tomar las reglas encontradas anteriormente verlas como una expresión algebraica y factorizar los elementos comunes, de tal manera que los síntomas comunes a varios diagnósticos quedan juntos y los síntomas particulares de un diagnóstico quedan asociados con él.

Conclusiones

Como se puede ver en este documento partimos de que se cuenta con un banco de información al que se le aplican técnicas evolutivas para construir el Directorio de Datos, la Base de Conocimiento y el conjunto de Reglas de Inferencia.

Lo maravilloso de este enfoque es que a partir de la misma fuente de información se pueden encontrar diferentes niveles de representación de conocimiento y la que se esta realizando es un diálogo con el interior del Banco de Información para encontrar reglas y conocimiento que se encuentra inmerso en las fuentes de información y que por lo común se pierden.

Con este trabajo se quiere mostrar que en los bancos de información se tiene una gran cantidad de información oculta, que sólo es necesario construir herramientas capaces de realizar procesos de diálogo con los almacenes de información y estos mecanismos son capaces de encontrar una gran cantidad de información inmersa y que hasta el momento no se aprovecha por no contar con las herramientas adecuadas, pero principalmente por desconocimiento de que estamos sentados sobre una mina de información y no sabemos aprovecharla o siquiera que existe.

Bibliografía

- [1] Galindo Soria, Fernando, Sistemas Evolutivos: Nuevo Paradigma de la Informática, Memorias de la XVII conferencia Latinoamericana de Informática Caracas. Ven. Julio. 1991.
- [2] Galindo Soria, Fernando, Sistemas Evolutivos en Boletín de Política Informática, INEGI-SPP, México. Sept 1986.
- [3] Olivares Ceja, Jesús Manuel, Sistema Evolutivo para Representación del Conocimiento, UPIICSA-IPN, México, D.F., 1991.
- [4].Galindo Soria, Fernando, Algunas Propiedades Matemáticas de los Sistemas Lingüísticos, Memorias del Symposium Nacional de Computación, México, nov. de 1992.
- [5] Ortiz, Javier, Galindo Soria, Fernando, Sistemas Evolutivos Constructores de Sistemas Expertos, CENIDET-DGIT. UPIICSA-IPN, Cuernavaca. Mor. 1988.
- [6] Berruecos Rodríguez, Elsa, Sistema Evolutivo Generador de Esquemas Lógicos de Bases de Datos, UPIICSA-IPN, México. 1990.

VII.5 Sistema Evolutivo Generador de Esquemas Lógicos de Bases de Datos

Elsa Berruecos Rodríguez⁶

Resumen

La construcción del Sistema Evolutivo Generador de Esquemas Lógicos de Bases de Datos se basa en el método de “Programación Dirigida por Sintaxis”, el cual, nos permite construir sistemas con la capacidad de inducir reglas generales a partir de ejemplos específicos para posteriormente aplicarlas a la solución de problemas particulares. Tal es el caso del sistema que se presenta en este trabajo, el cual a partir de ejemplos significativos del lenguaje del usuario (Lenguaje Natural), induce su gramática generativa para posteriormente poder interpretar oraciones que describan un ambiente y generar a partir de dicha descripción el esquema lógico deseado.

Palabras clave: Sistemas Evolutivos, Base de Datos, Lenguaje Natural, Programación Dirigida por Sintaxis, Gramáticas, Gramática con Atributos, Semántica, Forma Canónica, Recursividad.

Introducción

A raíz de la aceptación que han tenido los Sistemas Manejadores de Base de Datos (SMBD o DBMS por sus siglas en inglés), las compañías que desarrollan estos sistemas invierten cada año gran cantidad de recursos humanos y materiales para perfeccionar su funcionamiento, sin embargo, poco se ha hecho para facilitarle al usuario de estas herramientas la difícil tarea de diseñar las bases de datos que se implantarán en esos sistemas.

Existen actualmente varias metodologías para elaborar el diseño lógico y físico de una base de datos, sin embargo, a pesar de hacer uso de ellas, el tiempo que se debe dedicar a esta tarea continua siendo elevado y no puede llevarse a cabo sin tener un conocimiento profundo sobre diseño de base de datos. La herramienta que se propone para ayudar a solucionar este problema es un Sistema Evolutivo que genera o actualiza esquemas lógicos de bases de datos a partir de la descripción en lenguaje natural que el usuario haga del ambiente que desea manejar en dicha base de datos.

⁶ Elsa Berruecos Rodríguez realizó este trabajo en su seminario de titulación en la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) del Instituto Politécnico Nacional (IPN) el 5 septiembre de 1988.

La construcción de este sistema se basa en uno de los métodos más innovadores que en materia de desarrollo de sistemas se haya conceptualizado, la “Programación Dirigida por Sintaxis”, este método permite construir sistemas que tienen la capacidad de inducir reglas generales a partir de ejemplos específicos para posteriormente aplicarlas a la solución de problemas particulares, tal es el caso del Sistema Evolutivo Generador de Esquemas Lógicos de Bases de Datos, el cual a partir de ejemplos significativos del lenguaje del usuario induce su gramática generativa para posteriormente interpretar oraciones que describen el ambiente del que se obtiene el esquema de base de datos requerido.

El objetivo de este sistema es generar el esquema lógico de una Base de Datos que refleje el ambiente que el usuario le describa mediante oraciones en lenguaje natural. Pretende cubrir las necesidades de aquellas organizaciones que ya tienen instalado algún Sistema Manejador de Bases de Datos (SMBD, DBMS por sus siglas en inglés) y que ahora se enfrentan al problema de tener que capacitar a sus analistas para diseñar bases de datos.

La idea es que este sistema pueda ser utilizado por personas que aún sin tener un conocimiento profundo sobre Bases de Datos puedan hacer uso de ellas para aprovechar las facilidades que esta tecnología les brinda.

El sistema también se plantea el objetivo de ayudar a los diseñadores de bases de datos a definir y modificar esquemas fácilmente y sin que esto les consuma demasiado tiempo en el análisis.

Por el momento se tiene considerado que el sistema produzca esquemas de tipo relacional y que le proporcione al usuario su descripción en términos de las tablas, columnas, llaves primarias y llaves foráneas que constituyen su base de datos, además siguiendo los lineamientos que se plantean en este trabajo se pueden implantar otros modelos de base de datos como el orientado a objetos.

I Arquitectura del Sistema

En la figura 1 se muestra el Diagrama de Flujo de Datos del sistema, los procesos que lo componen se describen en forma general en las secciones de este capítulo.

1 Determinación de Unidades Léxicas

Este proceso se encarga fundamentalmente de pasar cada oración proporcionada por el usuario en lenguaje natural a su correspondiente forma canónica, sustituyendo cada una de las palabras por su tipo de unidad léxica. Las unidades léxicas que maneja el sistema son las que se muestran en la figura 2. Debido a las restricciones que tiene el proceso de comunicación con el sistema, es necesario que el usuario tome en cuenta los siguientes lineamientos:

- 1) *Las palabras que definan una sola entidad o atributo, deben unirse por guiones, ejemplos: cuenta-de-cheques, saldo-mensual.*

2) Las palabras que indiquen el número de ocurrencias de una entidad que está siendo descrita, también deberán estar unidas por guiones, ejemplos: un-sólo, un-único, uno-o-varios.

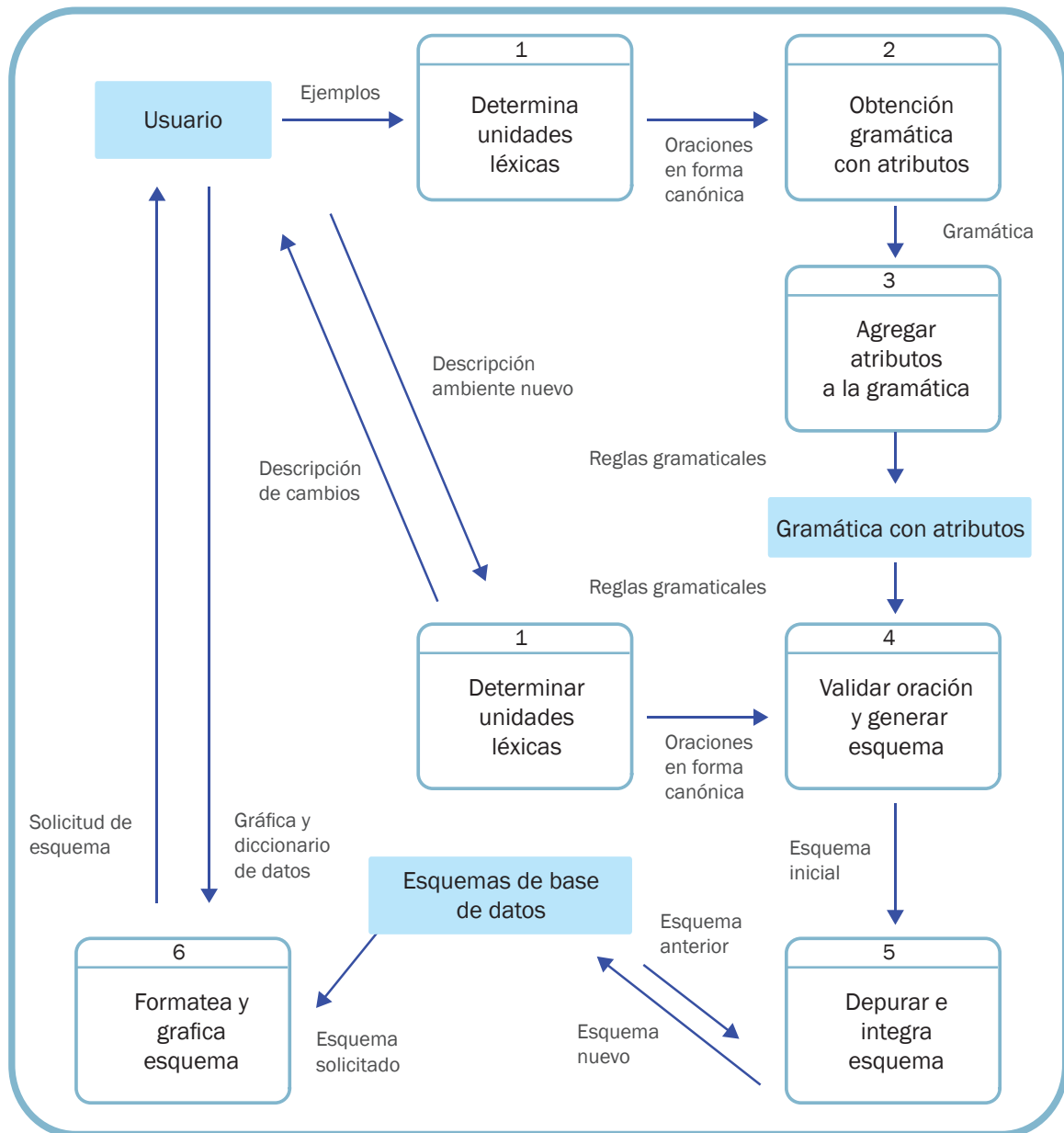


Figura 1. Arquitectura del Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos.

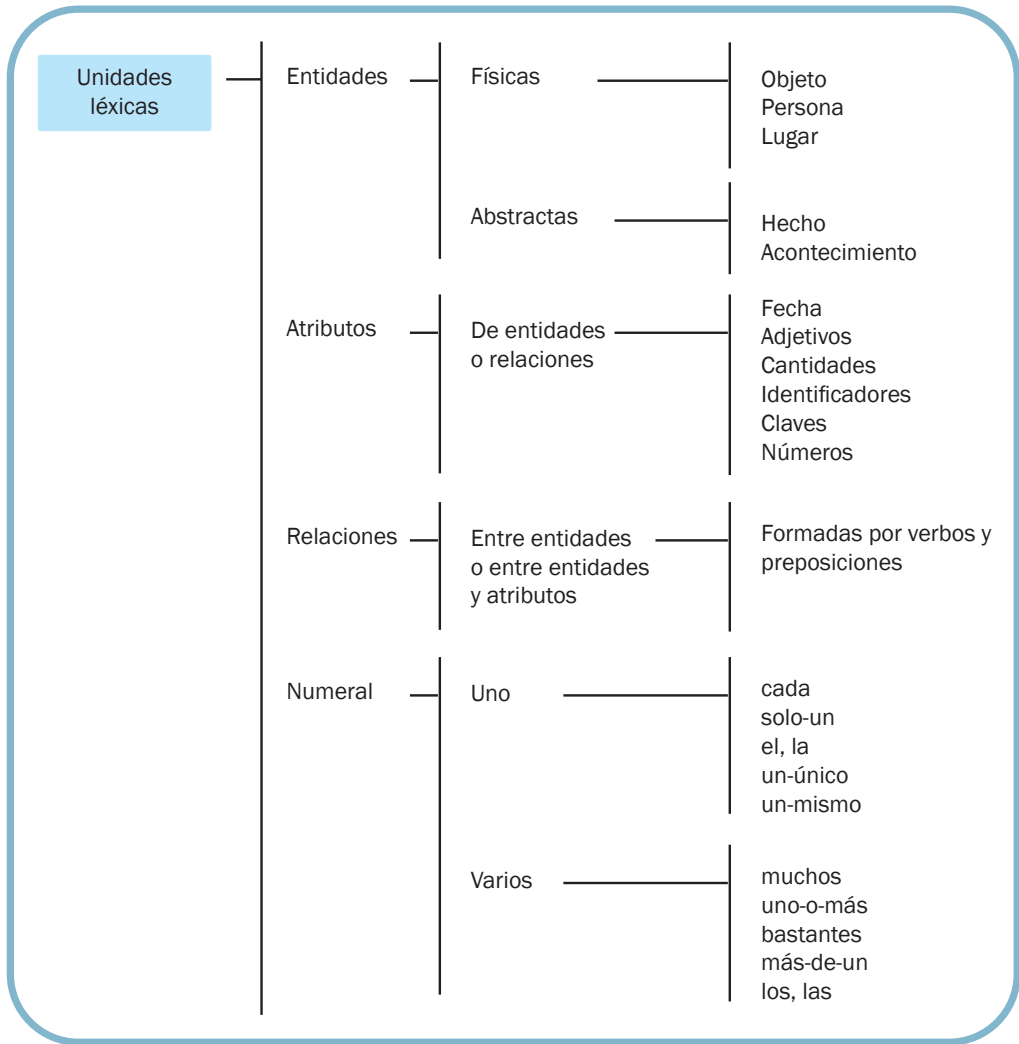


Figura 2. Unidades Léxicas que utiliza el Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos.

Hay dos tipos de oraciones de entrada:

1) *Oraciones que definen alguna relación*, ejemplo: Cada empleado es asignado a un-sólo departamento.

En base a este tipo de oraciones se induce la gramática del lenguaje del usuario.

2) *Oraciones que ayuden al sistema a identificar el tipo de unidad léxica que le corresponde a un palabra que se haya mencionado en alguna oración*, por ejemplo:

Sistema: ¿Qué es un empleado?

Usuario: es una persona

“persona” es una palabra clave que se considera en este ejemplo como sinónimo de Entidad.

2 Obtención de la Gramática Generativa

El objetivo de este proceso es obtener la gramática de las oraciones de ejemplo proporcionadas por el usuario. Esta gramática se obtiene.

1) *Detectando estructuras recursivas*. Para lograrlo se hace un barrido de cada oración para determinar cuantas veces dentro de una oración ocurre en forma contigua una unidad léxica. Las unidades que aparezcan más de dos veces en forma contigua se consideraran como estructuras recursivas.

2) *Factorizando oraciones detectando estructuras intermedias*. Para lograrlo se recorren las oraciones por pares de unidades léxicas. Se marcan las parejas que se repiten más veces a lo largo de todas las oraciones. Luego se recorren las oraciones pero ahora para detectar tercias de unidades léxicas. Se hace lo mismo que en el inciso anterior pero ahora con cuartetos. Los de mayor ocurrencia se sustituyen por variables no terminales y se define una regla de producción para ella. Se hace esto desde los cuartetos hasta los pares mientras quede alguno.

3 Agregar Atributos a la Gramática para Incluirle la Semántica

En este proceso se agregan las rutinas semánticas a la gramática que se obtuvo en el proceso anterior, en este caso las rutinas están relacionadas con el manejo de una pila.

4 Se Valida la Oración y se Genera el Esquema de Base de Datos Inicial

Este proceso está constituido por dos partes: la primera parte usa las reglas gramaticales obtenidas en el proceso descrito en C y en base a ellas se determina si una oración de entrada es valida y entendible para el sistema. La segunda parte, a partir de cada oración de entrada empieza a formar o modificar el esquema lógico de la Base de Datos que refleja el ambiente que el usuario le este describiendo.

5 Depuración e Integración del Esquema Definitivo

Si el usuario esta solicitando al sistema una modificación a un esquema ya almacenado, en este proceso se integraran las nuevas entidades, atributos y relaciones que se encontraron en el proceso descrito en D. Después de esto, se pasa por un proceso de normalización que permite depurar el esquema eliminando las anomalías que llegase a tener. Por otra parte, si el usuario esta solicitando la generación de un esquema por primera vez este proceso solo se encarga de normalizar el esquema inicial y de almacenarlo ya normalizado en el archivo de esquemas reconocidos por el sistema para que pueda ser consultado y/o modificado posteriormente por el usuario.

6 Formateo del Esquema y Graficación

En este proceso se accesa el archivo que tiene el esquema que desea ver el usuario y se formatea en un vocabulario entendible para él. La salida que se presenta al usuario esta compuesta básicamente por lo siguiente:

Un listado del esquema (entidades y relaciones) describiendo los objetos que lo constituyen.

ESQUEMA: nombre de la base de datos

ENTIDAD 1: nombre de la entidad

nombre de la columna 1

nombre de la columna 2

...

nombre de la columna n

LLAVE PRIMARIA: lista de columnas

LLAVES FORANEAS:

lista 1 de columnas

lista 2 de columnas

...

lista m de columnas

ENTIDAD 2: nombre de la entidad

columna1 nombre de la columna

columna2 nombre de la columna

...

columna n nombre de la columna

LLAVE PRIMARIA: lista de columnas

LLAVES FORANEAS:

lista 1 de columnas

lista 2 de columnas

...

lista m de columnas

Otra salida que se obtiene es la gráfica del modelo entidad-relación formado con rombos y cuadrados correspondientes a relaciones y entidades respectivamente.

II Obtención de la Gramática con Atributos

El problema fundamental del Lenguaje Natural es la complejidad de las oraciones que puede llegar a recibir el sistema, puesto que a pesar de tratarse de oraciones meramente declarativas presentan peculiaridades tales como:

1) *Símbolos iguales que tienen diferente significado en función de la posición en que se encuentren, por ejemplo:*

Cada empleado tiene número de empleado, dirección y teléfono y es asignado a varios proyectos en este caso la primera aparición de la letra “y” tiene la función de enlazar dos atributos que corresponden a una misma entidad, mientras que la segunda aparición de esta letra cumple la función de enlazar una oración con otra para proporcionar información adicional al respecto a la entidad principal de la oración.

2) *Símbolos que a pesar de ser diferentes pueden llegar a significar lo mismo, por ejemplo:* Cada empleado tiene número de empleado, dirección y teléfono como se puede observar la unidad léxica “,” y la unidad léxica “y” en el ejemplo cumplen la misma función, ambas actúan como nexos entre varios atributos que corresponden a una misma entidad.

3) *Recursividad a varios niveles*, en donde el primer nivel se puede ver en el ejemplo siguiente:

Oración en el lenguaje natural	Oración Canónica
Cada empleado tiene número-de-empleado.	C e r a.
Cada empleado tiene número-de-empleado, dirección.	c e r a , a
Cada empleado tiene número-de-empleado, dirección, edad.	c e r a , a , a
Cada empleado tiene número-de-empleado, dirección, edad, estado civil.	c e r a , a , a , a

donde: a=atributo, e=entidad, r=relación, c=cuantificador

La aparición consecutiva de la unidad léxica “a” separada por “,” puede considerarse recursiva y por lo tanto ser representada con las siguientes reglas de producción.

$$S \rightarrow c e r a A$$

$$A \rightarrow . \mid , a A$$

El segundo nivel de recursión se visualiza el ejemplo siguiente:

Oración en el lenguaje natural	Oración Canónica
Cada empleado es-asignado-a un departamento.	c e r c e .
Cada empleado es-asignado-a un departamento y a un proyecto.	c e r c e , c e , c e
Cada empleado es-asignado-a un departamento, a un proyecto y a un grupo	c e r c e , c e , c e

$$S \rightarrow A . \mid A , B . \mid A , B , B .$$

$$A \rightarrow c e r c e$$

$$B \rightarrow c e$$

simplificando

$$S \rightarrow A X$$

$$X \rightarrow . \mid , B X$$

$$A \rightarrow c e r c e$$

$$B \rightarrow c e$$

A su vez dentro de A y B puede haber un primer nivel de recursividad.

El tercer nivel de recursión se ejemplifica a continuación:

Oración en el lenguaje natural	Oración Canónica
Cada empleado es-asignado-a un departamento; cada departamento maneja varios proyectos; cada proyecto tiene número-de-proyecto, fecha-de-inicio y fecha-de-terminación .	c e r c e ; c e r c e ; c e r a , a y a .

$S \rightarrow A ; A ; A .$

$A \rightarrow c e r c e \mid c e r c e \mid c e r a , a y a$

simplificando

$S \rightarrow A X$

$X \rightarrow . \mid ; A X$

a su vez dentro de A puede presentarse el segundo nivel de recursividad.

III Módulos del Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos

Las funciones del Sistema Generador del Esquemas Lógicos de Base de Datos son fundamentalmente dos, aprender el lenguaje del usuario y utilizar el conocimiento adquirido para generar un esquema de Base de Datos de acuerdo a los requerimientos del usuario.

Para aprender el lenguaje del usuario se usan los módulos:

- 1) *Módulo de transformación de oraciones a su forma canónica*
- 2) *Módulo de obtención de la gramática generativa*
- 3) *Módulo de obtención de la gramática con atributos*

Para obtener el esquema de base de datos se usa el módulo:

- 1) *Módulo generador de esquemas de base de datos*

1 Módulo de Transformación de las Oraciones a su Forma Canónica

El objetivo de este módulo es identificar las unidades léxicas de cada oración de entrada para obtener su forma canónica formateada de tal manera que pueda ser de utilidad para la inducción de la gramática como para la generación del esquema de base de datos.

Los pasos que sigue el para transformar la oración en lenguaje natural a su correspondiente forma canónica son:

1) *Generación de la oración canónica inicial.* Consiste en sustituir cada palabra de la oración en lenguaje natural por su tipo de unidad léxica correspondiente.

Evidentemente el sistema en un principio no puede reconocer todas las palabras que contiene la oración puesto que solo cuenta con el vocabulario mínimo indispensable para realizar sus funciones, sin embargo tiene la capacidad de ir incrementando ese vocabulario conforme se vayan utilizando nuevas palabras dentro de las oraciones.

Cuando el sistema no reconoce alguna palabra, le pregunta al usuario de que tipo de unidad léxica se trata y la almacena como tal para poder reconocerla la próxima vez que la utilice el usuario, por ejemplo:

En cada departamento se tienen varios empleados con nombre, edad, sexo, dirección y además tienen varios proyectos con clave, nombre, fecha-de-inicio y fecha-de-terminación.

Unidad léxica	Nombre tipo	Tipo
En	preposición	p
cada	cuantificador	c
departamento	entidad	e
se	pronombre	o
tienen	verbo	v
varios	cuantificador	c
empleados	entidad	e
con	preposición	p
nombre	atributo	a
,	coma	,
edad	atributo	a
,	coma	,
sexo	atributo	a
,	coma	,
dirección	atributo	a
y	nexo	y
además	adverbio	d
tienen	verbo	v
varios	cuantificador	c
proyectos	entidad	e
con	preposición	p
clave	atributo	a
,	coma	,
nombre	atributo	a
,	coma	,
fecha-de-inicio	atributo	a
y	nexo	y
fecha-de-terminación	atributo	a
.	punto	.

2) *Aprendizaje de nuevas relaciones.* En esta etapa la oración no sufre ningún cambio puesto que el sistema sólo analiza la palabra o conjunto de palabras que el usuario haya marcado explícitamente como relaciones. Este análisis está encaminado a descubrir la estructura interna de las unidades léxicas Relación para darles de alta en la tabla de gramática de relaciones para que el usuario en futuras oraciones no tenga que unir con guiones aquellas palabras que conforman una relación, por ejemplo:

Cada empleado es-asignado-a un proyecto

Unidad léxica	Nombre tipo	Tipo
Cada empleado es-asignado-a un proyecto	cuantificador entidad relación cuantificador entidad	c e r c e

Si un usuario marca explícitamente la unidad léxica Es-asignado-a como una relación, se hace el análisis siguiente:

i) *Se separa la relación eliminando lo guiones*

es
asignado
a

ii) *Se determina el tipo de unidad léxica de cada palabra de la relación:*

Unidad léxica	Nombre tipo	Tipo
es asignado a	verbo verbo preposición	v v p

iii) *Se busca en la tabla TipRel la combinación v v p y si no se encuentra, se inserta esa combinación como un renglón nuevo de esa tabla.*

TipRel (antes)				TipRel (después)			
v	v			v	v	p	
v				v	v		
				v			

De esa manera la próxima vez que el usuario inserte una relación del tipo v v p, ya no será necesario que una las tres palabras con guiones.

3) *Sustitución de relaciones*. Esta etapa consiste en aplicar el conocimiento adquirido en la etapa anterior respecto a la estructura interna que guardan las unidades léxicas Relación. Para ello el sistema va recorriendo la oración canónica tratando de encontrar varias unidades léxicas contiguas que puedan agruparse para formar una Relación, de ser así sustituye las unidades léxicas correspondientes por la unidad léxica Relación, por ejemplo:

Tomando como base la oración canónica que se obtuvo en el punto 1), las unidades o, v y la unidad v fueron sustituidas por una unidad r.

4) *Depuración de la oración*. Una vez que han sido detectadas las oraciones dentro de la oración canónica, el siguiente paso es eliminar las preposiciones, artículos y demás unidades léxicas que se considera no serán determinantes durante la generación del esquema de lógico que corresponda a esa oración.

2 Módulo de Obtención de la Gramática Generalizada Mediante Inferencia Gramatical

Este módulo se encarga de inferir la gramática generativa del lenguaje del usuario a partir de un conjunto de oraciones que se alimentan al sistema durante su etapa de aprendizaje. Para lograr esto, efectúa el siguiente proceso sobre las oraciones canónicas que se obtuvieron como salida del modulo descrito en A.

1) *Fraccionar oraciones complejas en oraciones mas simples*. Inicialmente el sistema toma cada oración canónica y la desglosa en oraciones mas simples pero completas, sustituye en la oración original cada oración simple por un símbolo no terminal y graba cada fracción de oración como una opción de la regla de producción que corresponde a ese símbolo no terminal. Por ejemplo, considérense las oraciones siguientes:

Oración en lenguaje natural	Oración canónica
Cada cliente tiene una cuenta-de-cheques; cada cuenta pertenece a una sucursal; cada sucursal pertenece a una plaza.	c e r c e # @ ; c e r c e # @ ; c e r c e # @ .
Un cliente puede tener varias cuentas-de-cheques y varias cuentas-de-vpf; cada cuenta-de-vpf está asignada a un cliente.	c e r c e # y c e # @ ; c e r c e # @ .

Un cliente tiene una cuenta-de-cheques, varias cuentas-de-cartera y varias cuentas-de-bursatil.	c e r c e # y c e # y c e # y c e # @ .
Un cliente con nombre, dirección, teléfono y edad puede tener una cuenta-de-cheques y una cuenta-de-bursatil.	c e a , a , a , a r c e # y c e # @ .
Cada cliente tiene nombre y dirección	c e r a , a # @ .

Fraccionando las oraciones originales en oraciones más simples:

$S \rightarrow A ; A ; A . \mid A ; A . \mid A . \mid A .$
 $A \rightarrow c e r c e \# @ ; c e r c e \# @ ; c e r c e \# @ . \mid$
 $c e r c e \# y c e \# @ ; c e r c e \# @ . \mid$
 $c e r c e \# y c e \# y c e \# y c e \# @ . \mid$
 $c e a , a , a , a r c e \# y c e \# @ . \mid$
 $c e r a , a \# @ .$

2) *Detección del primer nivel de recursividad.* Una vez desglosadas las oraciones, se analiza la primera regla de producción y se determina si el símbolo no terminal aparece suficientes veces como para considerar que puede expresarse con una regla recursiva, de ser así, la primera regla se sustituye por dos para expresar esa recursividad.

Después de esto, se analiza cada una de las oraciones simples que ya se separaron de su oración original y se verifica si aun pueden simplificarse mas por estar compuestas de segmentos de oración; de ser así, se desglosa cada oración simple y se genera un nuevo símbolo no terminal para colocarlo en sustitución de cada segmento de oración.

En base a la gramática obtenida en el paso anterior se tiene:

$S \rightarrow A B$
 $A \rightarrow C @ \mid C y D @ \mid C y D y D y D @ \mid C y D @$
 $B \rightarrow ; A B \mid .$
 $C \rightarrow c e r c e \# \mid c e a , a , a , a r c e \# \mid c e r a , a \#$
 $D \rightarrow c e \#$

3) *Detección del segundo nivel de recursividad.* Este paso consiste en determinar si el segundo símbolo, no terminal (D) que fue generado en el paso anterior aparece suficientes veces como para considerarlo recursivo, de ser así, la regla de producción en donde aparece se transforma en dos reglas que expresen esa recursividad.

$$A \rightarrow C @ \mid C y D @ \mid C y D y D y D @ \mid C y D @$$

queda como:

$$\begin{aligned} A &\rightarrow C E \\ E &\rightarrow y D E \mid @ \end{aligned}$$

4) *Detección del tercer nivel de recursividad.* Debido a que el tercer nivel de recursividad se puede dar específicamente en el caso de la unidad léxica Atributo, y puesto que esta recursividad esta empotrada dentro de cada segmento de oración, si se detecta que aparece consecutivamente tres veces o mas, se asume que es recursiva y se extrae de la oración junto con la unidad léxica que le sucede para que sirva como símbolo de terminación de la recursividad y en el sitio donde se extrajo el conjunto de atributos, se incluye un nuevo símbolo no terminal y se genera su regla de producción correspondiente.

$$C \rightarrow c e r c e \# \mid c e a , a , a , a r c e \# \mid c e r a , a \#$$

queda como:

$$\begin{aligned} C &\rightarrow c e r c e \# \mid c e F c e \# \mid c e r F \# \\ F &\rightarrow a G \\ G &\rightarrow , a G \mid r \mid \# \end{aligned}$$

5) *Depuración de la gramática.* Finalmente se ordenan las alternativas de cada regla de producción y se eliminan aquellas que sean iguales para dejar simplificada la gramática.

La gramática resultante del ejemplo es:

$$\begin{array}{ll} S \rightarrow A B & D \rightarrow c e \# \\ A \rightarrow C E & E \rightarrow y D E \mid @ \\ B \rightarrow ; A B \mid . & F \rightarrow a G \\ C \rightarrow c e r c e \# \mid c e F c e \# \mid c e r F \# & G \rightarrow , a G \mid r \mid \# \end{array}$$

3 Módulo para Obtener la Gramática con Atributos

La gramática con atributos es una de las herramientas que nos permiten asignar y manejar la semántica de un lenguaje. Este tipo de gramática se caracteriza por llevar intercalados en sus reglas de producción llamados a rutinas semánticas. Las rutinas semánticas se representan como símbolos no terminales y su función es la de llevar a cabo alguna acción sobre los datos, por ejemplo:

$$S \rightarrow c a r S1 c a . S2$$

donde S1 y S2 son rutinas semánticas.

El ejemplo indica que si se esta en el símbolo S y llega la señal r, entonces se manda a ejecutar la rutina S1 y se verifica si llega la señal c.

Las rutinas semánticas del sistema evolutivo en cuestión se basan en el manejo de una pila que se utiliza como entrada de la oración que se va a reconocer y en función del contenido de esta pila se van llenando las estructuras de datos de la figura 3.

Entidades		Atributos	
	entidad	atributo	entidad
1	cuenta-de-cheques	saldo-mensual	1
2	sucursal	número-de-cuenta	1
3		número-de-sucursal	2
4			

Relaciones				
c1	e1	relación	c2	e2
u	1	es-abierta-en	u	2
u	2	maneja	m	1

donde c1=cuantificador1, c2=cuantificador2, e1=entidad1, e2=entidad2, u=uno, m=muchos

Figura 3. Estructuras de datos utilizadas para generar y modificar los esquemas de base de datos

Un ejemplo de las rutinas semánticas de este sistema es el siguiente:

S1: Rutina semántica que se ejecuta ante la entrada de la unidad léxica CUANTIFICADOR
ALGORITMO:

Si la pila esta vacía o en el tipo de la pila hay una relación o una entidad
entonces

 almacena el cuantificador en la pila

sino

 si en el tope de la pila hay una “y” entre oraciones

 entonces

 saca la “y” de la pila

 almacena el cuantificador en la pila

La complejidad del diseño de las rutinas semánticas de este sistema radica en el hecho de que cada atributo debe ser asociado con la entidad que le corresponde y las entidad que intervienen en la oración deben ser correctamente relacionadas unas con otras, independientemente de la estructura que guarde la oración de entrada.

4 Módulo de Generación del Esquema de Base de Datos

Este modulo cumple dos funciones básicas:

- 1) *Determinar si una oración de entrada corresponde al lenguaje que aprendió el sistema.*
- 2) *Interpretar cada oración de entrada y construir el esquema lógico que la representa*

Estas dos funciones se llevan a cabo simultáneamente gracias a la gramática con atributos que se obtuvo en el modulo anterior, puesto que al ir siguiéndola es posible determinar si la oración esta bien construida al mismo tiempo que se van ejecutando las rutinas semánticas que llenan las estructuras de datos que reflejan el esquema lógico correspondiente.

Al final de este modulo existe una rutina que se encarga de formatear el esquema obtenido, de manera que sea entendible para el usuario.

Ejemplo: Un usuario podría describir su ambiente problema de la manera siguiente:

Un cliente puede tener varias cuentas-de-cheques. Cada cuenta-de-cheques tiene número-de-cuenta, saldo, número-de-retiros y número-de-depósitos y puede pertenecer a uno-o-mas clientes. Un cliente es identificado por su nombre y RFC. Cada cliente tiene dirección y número-telefónico. Cada cuenta-de-cheques es abierta en una sucursal y cada sucursal puede abrir varias cuentas-de-cheques.

El esquema lógico que propone el generador es el siguiente:

Entidades

Cliente

	nombre RFC dirección número-telefónico identificador-cliente
--	--

Cuenta-de-cheques

	número-de-cuenta saldo número-de-retiros número-de-depósitos identificador-cuenta-de-cheques
--	--

Sucursal

	identificador-sucursal
--	------------------------

Relaciones

1 Cliente	Puede-tener	M Cuentas-de-cheques
1 Cuenta-de-cheques	Puede- pertenecer es-abierta-en puede-abrir	M Clientes 1 Sucursal
1 Sucursal		M Cuentas-de-cheques

Ejemplos de oraciones que se pueden utilizar para actualizar un esquema de base de datos son:

Desde mañana los clientes tienen derecho a darle una extensión de su cuenta de cheques a sus hijos. En el cálculo de impuestos se maneja una tabla de subsidio que tiene los mismos atributos que la tabla actual del cálculo del ISR.

IV Obtención Automática de Llaves y Tipos de Datos (Propuesto por Jesús Manuel Olivares Ceja)

Las llaves de una entidad se pueden obtener en base al tipo de consultas que haga el usuario, por ejemplo si se solicita:

Dame las personas cuyo apellido inicia con L. Busca los datos del empleado Jiménez.

El sistema genera un índice por apellido para facilitar la atención a estos requerimientos.

El tipo y la longitud de los atributos se puede obtener en base a ejemplos de datos que se maneja para cada uno, de donde el sistema infiere el tipo y longitud más apropiados. En un momento dado puede efectuar un cambio tanto en tipo como en longitud si los datos que llegan son diferentes:

Por ejemplo si se da: *Algunos nombres de personas son: Juan Pérez Herrera, Claudia Bracamontes.*

El sistema asigna un atributo de tipo alfabético de la longitud máxima del nombre que tenga registrado.

En otro ejemplo, cuando se da: *El sexo únicamente puede ser masculino o femenino.*

El sistema puede asignar una variable que tome dos valores como por ejemplo un bit.

VII.6 Sistema Evolutivo para Representación del Conocimiento

Jesús Manuel Olivares Ceja⁷

Resumen

Se describen los elementos de un Sistema Evolutivo que usa para representar conocimiento: una Red Semántica Ampliada (RSA), la cual toma de oraciones declarativas hechos formados por dos objetos unidos por una relación, para obtener los hechos explícitos se utiliza la distribución lingüística. La RSA crece mediante un proceso de agregación. En la RSA mediante un proceso de factorización se encuentran hechos comunes que pueden asignarse a objetos genéricos y de esta forma simplificar la red, al tiempo que se tienen hechos nuevos. La RSA se consulta mediante oraciones interrogativas e imperativas.

Palabras clave: Sistemas Evolutivos, Redes Semánticas, Inteligencia Artificial, Lingüística.

Introducción

En los Sistemas Evolutivos se considera la integración del enfoque inductivo y deductivo en la solución de los problemas informáticos; es de esto de donde surge su arquitectura general [3] [4] [6] [25] (figura 1).

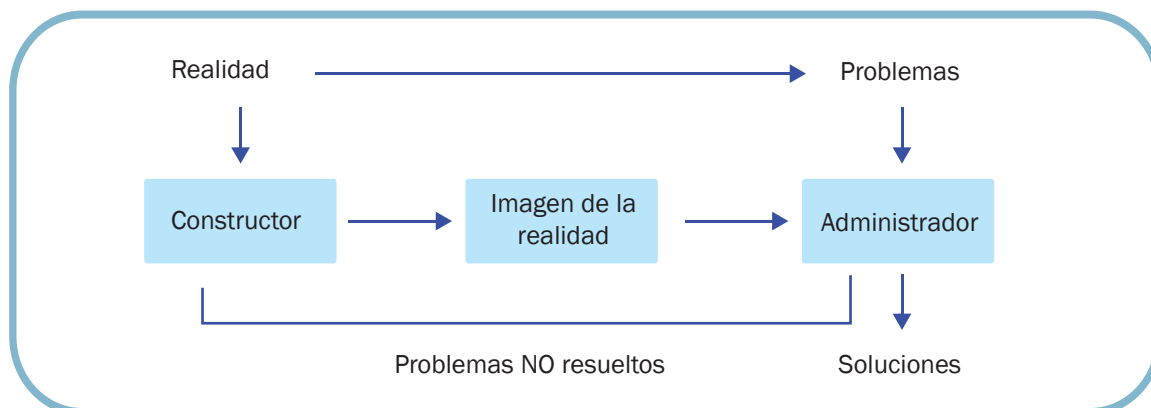


Figura 1. Arquitectura General de un Sistema Evolutivo.

⁷ Jesús Manuel Olivares Ceja escribió este trabajo como parte de su tesis de licenciatura en ciencias de la informática en la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) presentada en abril de 1991.

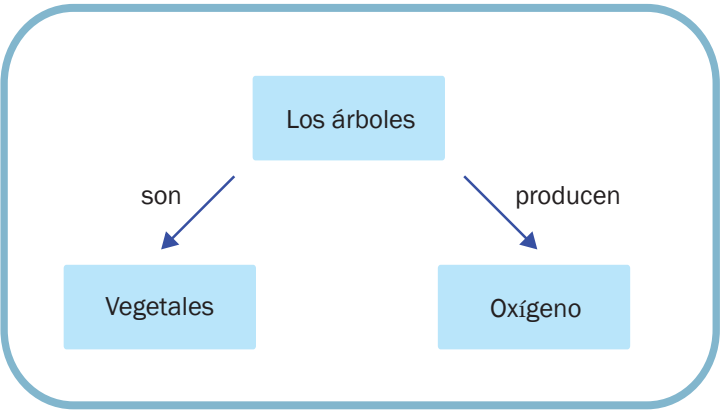
El sistema presentado se encuentra enmarcado en la implementación de los mecanismos que permiten obtener una herramienta evolutiva, en la que una aplicación puede ser la enseñanza automatizada. Algunos de estos pueden ser:

Captación de la realidad	Conocimiento	Presentación y aplicación
<ul style="list-style-type: none">• voz• imágenes• señales• texto• sonido	<ul style="list-style-type: none">• registro• inducción• deducción• planeación• analogía	<ul style="list-style-type: none">• animación• texto• señales de control• sonido• voz

El sistema que se presenta pertenece al de conocimiento. Apoya su representación de la realidad en una Red Semántica Ampliada (RSA) que se forma tomando hechos (formados por objetos relacionados) de oraciones declarativas en algún lenguaje como Español, Inglés, etc. En la RSA utilizando la factorización se pueden inducir hechos generales asignándolos a un objeto genérico. La información registrada se puede solicitar mediante consultas sencillas con oraciones interrogativas e imperativas.

Ejemplo 1: Dada la oración declarativa:

LOS ÁRBOLES SON VEGETALES Y PRODUCEN OXÍGENO se obtiene:



donde una posible consulta y su respuesta es:

¿QUE SON LOS ÁRBOLES?

son VEGETALES

I Representación de Conocimiento

El conocimiento se maneja aquí como el conjunto de información que alguien (o algo) tiene y le permite interactuar, obrar y hacer en su realidad [25], esta puede ser de múltiples formas: imágenes, señales, etc.

Se emplea el concepto de Red Semántica Ampliada (RSA) (figura 2) como una extensión de las Redes Semánticas en cuanto a la información que se registra en sus nodos, pudiendo ser: etiquetas, figuras, conceptos, procesos, etc. La semántica de cada nodo está dada por las relaciones que tiene con los demás [1] [9] [21] [25].

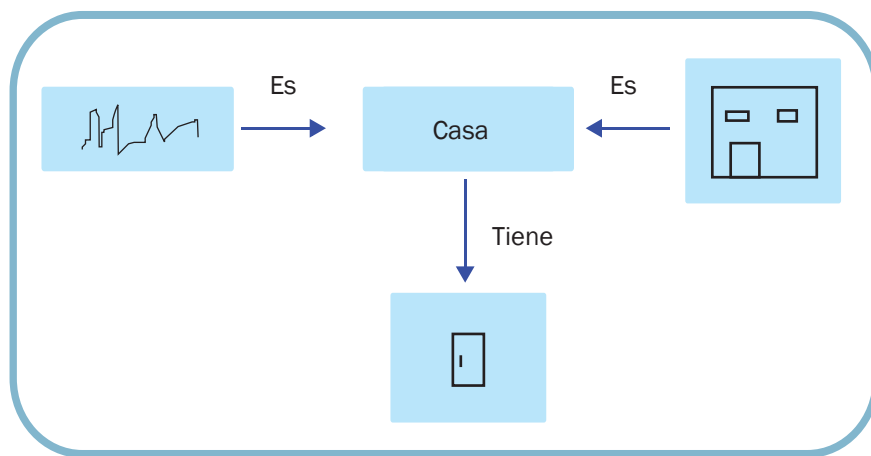


Figura 2. Una Red Semántica Ampliada.

Se ha propuesto un modelo [25] que ilustra el crecimiento de la RSA por agregación de objetos, y donde se aprecia que las estructuras obtenidas son semejantes entre sí, lo que es característico de los fractales.

II Componentes del Sistema

La arquitectura general del sistema descrito se muestra en la figura 3.

El CONSTRUCTOR LÉXICO [4] identifica las palabras presentes en las oraciones que se dan, le asigna un tipo a cada una y obtiene su estructura u oración canónica.

Los principales tipos que se usan para obtener los hechos de las oraciones declarativas son: objeto (o) y relación (r).

Ejemplo 2: De la oración:

LOS ÁRBOLES son VEGETALES y producen OXÍGENO

se obtienen las unidades léxicas siguientes:

Unidad léxica	Tipo
LOS ÁRBOLES	o
son	r
VEGETALES	o
y	i
producen	r
OXÍGENO	o

junto con la oración canónica:

o1 r1 o2 r2 o3

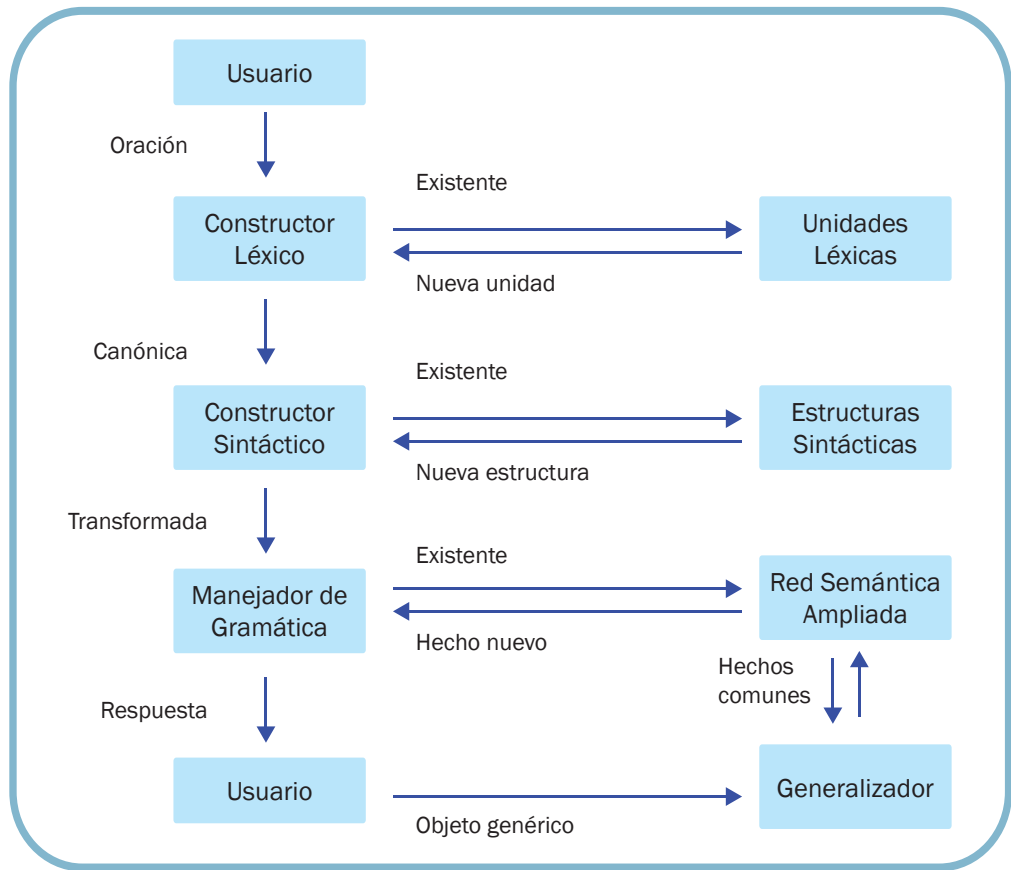


Figura 3. Arquitectura del SERC.

El Constructor Sintáctico aplica alguna transformación a las oraciones canónicas del constructor anterior, en caso de requerirse. Una de esas operaciones es la distribución lingüística [16] [25] con la que se obtienen explícitos los hechos de las oraciones declarativas. Esta operación se basa en su similar algebraica, ejemplo:

$$a(b + c) = a b + a c$$

En el ámbito lingüístico se usa el operador “+” (y), los paréntesis para agrupación de operandos y el operador implícito “*” (concatenación).

Ejemplo 3: Aplicando distribución al ejemplo 2, se tiene:

LOS ÁRBOLES son VEGETALES y producen OXÍGENO

o1 r1 o2 r2 o3

se asignan los operadores lingüísticos en la oración canónica:

(o1) (r1 o2 + r2 o3)

se distribuye o1:

o1 r1 o2 + o1 r2 o3

de donde resultan los hechos que se registran en la RSA:

LOS ÁRBOLES son VEGETALES ;

o1 r1 o2 ;

LOS ÁRBOLES producen OXÍGENO

o1 r2 o3

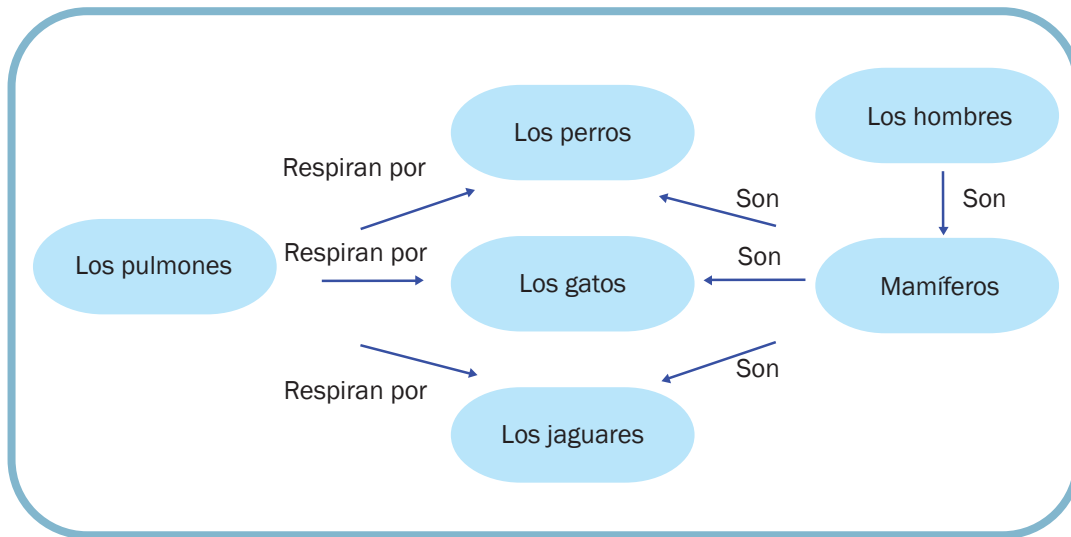
El MANEJADOR DE GRAMÁTICA CON ATRIBUTOS interpreta las oraciones transformadas del constructor sintáctico e invoca a las rutinas semánticas correspondientes según la oración escrita, pudiendo ser:

- 1) *Las que integran los hechos en la RSA.*
- 2) *Las que dan la respuesta a requerimientos.*

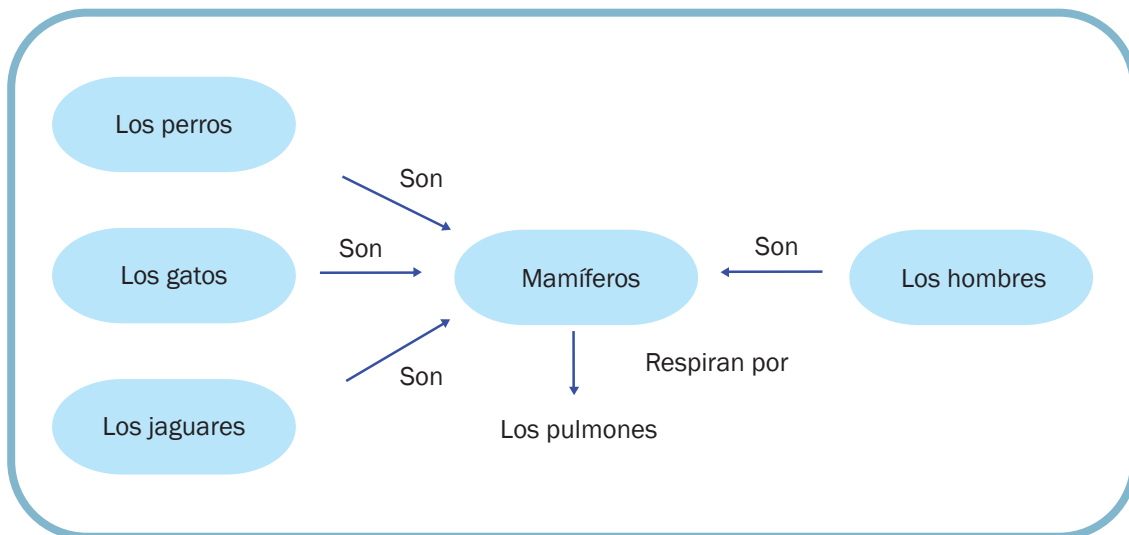
El GENERALIZADOR usa la factorización de la inferencia gramatical [12] [25] y le permite al usuario asignar los hechos comunes a un objeto genérico que él define.

La generalización, en un momento dado se puede plantear como regla del tipo: SI premisa ENTONCES conclusión.

Ejemplo 4: De la siguiente RSA:



sacando como FACTOR COMÚN los hechos “son MAMÍFEROS” y “respiran por LOS PULMONES” y asignando a MAMÍFEROS como objeto genérico se obtiene:



o se puede proponer:

SI X son mamíferos ENTONCES respiran por los pulmones con lo anterior se puede inducir que LOS HOMBRES respiran por LOS PULMONES por estar relacionados con MAMÍFEROS.

La parte correspondiente al ADMINISTRADOR [25] usa los programas del constructor y las rutinas semánticas que dan respuesta a los requerimientos.

Conclusiones

Se presentó un Sistema Evolutivo con el que el usuario puede interactuar para registrar información o realizar consultas mediante oraciones declarativas, interrogativas o imperativas, la información se registra en una Red Semántica Ampliada, sobre la que se aplican las consultas y la búsqueda de objetos y relaciones genéricos que dan lugar a generalizaciones útiles para simplificar la red y reducir el número de casos particulares. Es una herramienta de apoyo al aprendizaje interactivo.

Bibliografía

- [1] GALINDO Soria, Fernando, SISTEMAS EVOLUTIVOS DE LENGUAJES DE TRAYECTORIA en VI Reunión de Inteligencia Artificial, Memorias, Presidente José Negrete M., Ed. Limusa, Junio 1989, Querétaro, Qro.
- [2] GALINDO Soria, Fernando, SISTEMAS EVOLUTIVOS, IPN-UPHCSA Unidad de Investigación y Desarrollo, Junio 1985, México.
- [3] GALINDO Soria, Fernando, CONSTRUCCIÓN DE SISTEMAS EVOLUTIVOS, IPN-UPHCSA.
- [4] GALINDO Soria, Fernando, CONFERENCIA TUTELAR, Introducción a los Sistemas Evolutivos, IPN-UPHCSA Departamento de Computación.
- [5] BERRUECOS Rodríguez, Elsa, SISTEMA EVOLUTIVO GENERADOR DE ESQUEMAS LÓGICOS DE BASES DE DATOS, IPN-UPHCSA, México.
- [6] VICARIO Solorzano, Marina, AGUILAR Vallejo, Francisco Javier, EVA: EVOLUCIÓN APLICADA, México, D.F., 1989.
- [7] AGUILAR Y AGUILAR, Araceli, Et. al, AVANCES Y EJEMPLOS DE SISTEMAS EVOLUTIVOS, IPN-UPHCSA Lic. en C. de la Informática, Julio 1988.
- [8] GALINDO Soria, Fernando, Notas del Seminario de Titulación "CONSTRUCCIÓN DE SISTEMAS EVOLUTIVOS", IPN-UPHCSA, Octubre 1989 a Marzo 1990. México.
- [9] AGUILAR Y AGUILAR Araceli, Et. al, SISTEMAS EVOLUTIVOS, Conferencia en IPN-CENAC, 5 Agosto 1988. México.
- [10] ORTIZ Hernández, Javier, GALINDO Soria, Fernando, SISTEMAS EVOLUTIVOS CONSTRUCTORES DE SISTEMAS EXPERTOS, Centro de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México.
- [11] GALINDO Soria, Fernando, PROGRAMACIÓN DIRIGIDA POR SINTAXIS, IPN-UPHCSA-Lic. en C. de la Informática, Dic. 1985.
- [12] GALINDO Soria, Fernando, INFERENCIA GRAMATICAL (Hacia la matemática de la Informática), IPN-UPHCSA, México, D. F., DGIT-CENIDET, Cuernavaca, Mor., Oct. 1988.
- [13] BERRUECOS Rodríguez, Elsa, SISTEMA EVOLUTIVO GENERADOR DE ESQUEMAS LÓGICOS DE BASE DE DATOS, IPN-UPHCSA, México 1990.
- [14] GALINDO Soria, Fernando, REPRESENTACIÓN DEL CONOCIMIENTO, IPN-UPHCSA, Jul 1985, México, D.F.
- [15] CHOMSKY, Noam, ESTRUCTURAS SINTÁCTICAS, 9a edición, introducción, notas, apéndice y traducción de C. P. Otero, Ed. Siglo XXI, México, 1987.
- [16] GALINDO Soria, Fernando, Comunicación personal sobre DISTRIBUCIÓN DE CADENAS LINGÜÍSTICAS, Diciembre 1990, IPN-UPHCSA, México.

- [17] RICH, Elaine, ARTIFICIAL INTELLIGENCE, Mc. Graw Hill, Singapoure, 1985.
- [18] GANASCIA, Jean-Gabriel, LA CONCEPCIÓN DE LOS SISTEMAS EXPERTOS en Mundo Científico No. 53.
- [19] FAGOT Largeault, Anne, LA SIMULACIÓN DEL RAZONAMIENTO MEDICO en Mundo Científico No. 53.
- [20] LOPEZ DE MANTARAS Badia, Ramón, MODELOS DE RAZONAMIENTO APROXIMADO en Mundo Científico No. 53.
- [21] GALLAIRE, Hervé, LA REPRESENTATION DES CONNAISSANCES en La Recherche No. 170, Octubre 1985.
- [22] JULLIEN, Rémi, Botet, Robert, Kolb, Max, LOS AGREGADOS en Mundo Científico No.75.
- [23] BOTET, Robert, Jullien, Rémi y Skjeltorp, Arne T., LA FORMA RESULTADO DEL CRECIMIENTO en Mundo Científico No. 75.
- [24] LESORT, Marc, COMO DESCUBRI LOS FRACTALES entrevista a Benoit Mandelbrot en Mundo Científico No.58.
- [25] OLIVARES Ceja, Jesús Manuel, SISTEMA EVOLUTIVO PARA REPRESENTACIÓN DEL CONOCIMIENTO, IPN-UPICSA, México, 1991.
- [26] TEC-COMP 91, APPROACHES TO NON-CONVENTIONAL COMPUTING TOWARDS INTELLIGENT SYSTEMS, Proceedings, April 15th thru 18th, 1991, World Trade Center, Mexico City, ITESM-CEM.

VII.7 Zanya: Compositor Evolutivo de Música Virtual

Horacio Alberto García Salas⁸

Introducción

La Informática es una disciplina que ha permitido grandes avances en muy corto tiempo y hay que destacar que ha tomado una gran aceleración con la aparición de herramientas como las computadoras y una serie de dispositivos electrónicos que hacen que el mundo en el que vivimos se vuelva cada vez más pequeño.

Prácticamente, todas las áreas del conocimiento humano se han visto apoyadas con el uso de estas nuevas tecnologías, tal es el caso de las Bellas Artes y en particular de la Música.

De esta manera, tenemos por un lado a la *informática*, que ha revolucionado el mundo drásticamente y por otro lado la *Música*, que es tan antigua como el mismo ser humano. Unidas ambas áreas con el paso del tiempo, han creado el marco dentro del cual fue desarrollado este trabajo, “*La Música Informática*”.

En el área de desarrollo informático musical se encuentran diferentes corrientes de desarrollo, así pues encontramos a quienes se dedican a fabricar software de edición musical, reproductores de música, secuenciadores, etc.

Y por otro lado, otra de las corrientes, hasta ahora más de tipo experimental, es la dedicada a la obtención de software para la *composición musical automática*, “el sueño” de la máquina con la capacidad de componer música por siempre exquisitamente diferente.

Con la aparición de los *fractales*[1] y la aplicación de las teorías de *caos*[8], el problema de crear una máquina capaz de hacer composición musical automática, se centró en lo natural. Y no es para menos, los hermosos paisajes generados por el científico Benoit Mandelbrot, han mostrado que la mejor manera de hacer modelos es imitar el comportamiento de la Naturaleza [4].

Aplicando teorías fractales, el científico Richard F. Voss[2], ha hecho desarrollo musical informático, obteniendo compositores verdaderamente hermosos. En su trabajo ha desarrollado generadores de ruido browniano, ruido blanco y resalta uno que genera música muy agradable,

⁸ Horacio Alberto García Salas.

que se denomina ruido $1/f$ [5], que de acuerdo a algunos estudios que ha realizado, música como el jazz, la clásica, el blues etc., se comportan de manera muy similar al ruido $1/f$. La cuestión es un poco más trascendente cuando se descubre que muchos fenómenos naturales, como la aparición de manchas solares, el crecimiento de las poblaciones, la formación de nubes, etc., tienen un comportamiento similar al ruido $1/f$.

Escuchar el algoritmo para la generación de ruido $1/f$ es bastante agradable.

Por otro lado, el investigador Fernando Galindo Soria ha desarrollado una herramienta informática que tiene una amplia gama de aplicaciones, a la que ha denominado “*Sistemas Evolutivos*” [3].

Los Sistemas Evolutivos modelan una de las principales características de la Naturaleza, la *Evolución*.

La Naturaleza ha mostrado desde hace largo tiempo, que los sistemas que evolucionan son los que tienen la capacidad de adaptarse al medio que los rodea. De tal forma que si lo que se pretende al hacer un modelo es imitar a la realidad, ¿porqué no desarrollar modelos o sistemas con la capacidad de adaptarse al medio que los rodea? En general, este tipo de sistemas son hermosos, basta con echar una mirada a nuestro alrededor para percatarnos de los bellos sistemas evolutivos que la Naturaleza tiene por gusto crear, y es que prácticamente todo a nuestro alrededor evoluciona, sean animales, plantas o minerales, en general sufren modificaciones del medio que los rodea y a su vez modifican al medio que los contiene, creándose procesos de coevolución[6], convirtiendo a la evolución en un proceso Universal.

De esta manera, cuando se desarrolla un sistema evolutivo, hay que pensar en un sistema que tenga la capacidad de interrelacionarse con el medio ambiente y que aprenda a través de él. Aplicando esta filosofía, se han desarrollado impresionantes Sistemas Evolutivos, algunos de ellos aplicados a la generación de paisajes.

Aplicando la misma filosofía, en este trabajo presentamos el desarrollo de Zanya, un Sistema Evolutivo que tiene la capacidad de aprender y evolucionar de manera permanente, con el fin de hacer composiciones musicales.

1. Sistema compositor

1.1 Generación de las notas musicales

Como primer paso en el desarrollo de un sistema informático musical, como el que se presenta en este trabajo, es interesante hablar de las notas musicales, que representan la base sobre la cual está sustentada la música.

En términos de la Física, las notas musicales son ondas que están comprendidas en un rango de frecuencias, relativamente pequeño, para el cual el oído humano está perfectamente adaptado. El ser humano tiene la capacidad de escuchar las frecuencias de entre 20 y 20000 Hz[7] y

sólo ciertas frecuencias se consideran como notas musicales puras. Cada instrumento musical emite las frecuencias puras de las notas musicales acompañadas por algunas otras frecuencias, denominadas armónicas, que permiten distinguir cuando se trata de un piano, un violín, un saxofón o cualquier otro de los instrumentos musicales que existen.

De esta forma lo primero que se necesita construir es una representación de las notas y se puede lograr almacenando en un arreglo los valores de las frecuencias de las notas, creándose de esta manera un “piano virtual” sobre el cual se pueden interpretar melodías y desde luego, hacer composiciones musicales.

Cuando se duplica el valor de una nota, *Do* por ejemplo, que tiene una frecuencia de 65 ciclos/seg, se obtiene una nota que se encuentra una octava o escala más aguda, sin embargo, esta nota también es *Do*, sólo que con una frecuencia de 130 c/s y si volvemos a duplicar este valor, se obtiene otro *Do*, pero aún más agudo, con una frecuencia de 260 c/s. De igual forma se puede hacer con todas las notas musicales. En la tabla 1 se pueden ver los valores de las frecuencias de 5 escalas cromáticas.

Do	#Do	Re	#Re	Mi	Fa	#Fa	Sol	#Sol	La	#La	Si
65	69	73	77	82	87	92	98	103	110	116	123
130	138	146	154	164	174	184	196	206	220	232	246
260	276	292	308	328	348	368	392	412	440	464	492
520	552	584	616	656	696	736	784	824	880	928	984
1040	1101	1168	1232	1312	1392	1472	1568	1648	1760	1856	1968

Tabla 1 Frecuencias de las notas musicales en ciclos/seg.

1.2 Mecanismo general de composición

A continuación se describirá la forma en la que se generan las composiciones musicales. Para representar la información musical se utilizan dos matrices una que se ha denominado *Matriz Evolutiva Aleatoria* y otra que se obtiene a partir de esta, llamada *Matriz Evolutiva de Frecuencias Acumuladas*. Dichas matrices son arreglos de 60 renglones por 60 columnas, sin embargo, para la explicación de su funcionamiento, utilizaremos matrices de 7 renglones por 7 columnas.

a) Construcción de la Matriz Evolutiva Aleatoria

Para ejemplificar el funcionamiento de estas matrices, primeramente, se construye un arreglo cuadrado llamado *Matriz Evolutiva Aleatoria*, utilizando como etiquetas de las columnas y renglones las notas musicales. Después, se llenarán algunas de las casillas con números aleatorios como se ve en el ejemplo de la fig. 1.

	<i>Do</i>	<i>Re</i>	<i>Mi</i>	<i>Fa</i>	<i>Sol</i>	<i>La</i>	<i>SI</i>
<i>Do</i>	0	0	90	0	30	40	0
<i>Re</i>	10	0	15	45	20	70	80
<i>Mi</i>	5	25	0	60	0	30	90
<i>Fa</i>	0	80	10	40	15	95	30
<i>Sol</i>	25	15	0	40	65	70	0
<i>La</i>	0	35	5	10	0	0	0
<i>Si</i>	20	30	0	0	60	0	70

Fig 1. Matriz Evolutiva Aleatoria.

b) *Construcción de la Matriz Evolutiva de Frecuencias Acumuladas*

A partir de la matriz evolutiva aleatoria se construye la *Matriz Evolutiva de Frecuencias Acumuladas* y se hace de la siguiente manera:

Se agrega una columna llamada *totales*, que originalmente está llena de ceros, a la derecha en la matriz evolutiva aleatoria.

En cada renglón vamos a hacer un recorrido de izquierda a derecha ignorando las casillas con valor cero.

El primer número (más a la izquierda) del renglón, que se encuentre diferente de cero se suma a *totales* y el resultado sustituye a este primer número.

El segundo número diferente de cero se suma a *totales* y el resultado sustituye al segundo número.

Así sucesivamente con todos los valores diferentes de cero, el *i*-ésimo número diferente de cero se suma a *totales* y el resultado sustituye al *i*-ésimo número. Agotados estos, la suma debe quedar almacenada en la columna *totales* del renglón.

Por ejemplo, el renglón original es

	Do	Re	Mi	Fa	Sol	La	Si	Totales
Do	0	0	90	0	30	40	0	0

Y al aplicar el algoritmo queda:

	0+9				90-30	120+40	90+30+4	
	Do	Re	Mi	Fa	Sol	La	Si	Totales
Do	0	0	90	0	120	160	0	160

De esta manera, al recorrer de izquierda a derecha cada uno de los renglones de la matriz evolutiva aleatoria, en la columna *totales* de la matriz evolutiva de frecuencias acumuladas quedará almacenada la suma de los valores diferentes de cero de cada renglón y en cada casilla con valor diferente de cero se almacenará el valor acumulado durante el recorrido.

En esta forma se obtiene la *matriz evolutiva de frecuencias acumuladas*, como se ve en la figura 2.

	Do	Re	Mi	Fa	Sol	La	Si	Totales
Do	0	0	90	0	120	160	0	160
Re	10	0	25	70	90	160	240	240
Mi	5	30	0	90	0	190	210	210
Fa	0	80	90	130	145	240	270	270
Sol	25	40	0	80	145	215	0	215
La	0	35	40	50	0	0	0	50
Si	20	50	0	0	110	0	180	180

Fig. 2 Matriz Evolutiva de Frecuencias Acumuladas.

c) Generación de la Música

Los números contenidos en la matriz anterior, representan la probabilidad de pasar de una nota a otra y se van a utilizar para hacer la composición musical mediante el siguiente proceso:

-Se escoge aleatoriamente uno de los renglones de la matriz evolutiva de frecuencias acumuladas y este representa la primer nota de la composición.

-Se utiliza el valor almacenado en la columna totales de ese renglón para generar un número aleatorio entre cero y dicho valor.

-El número aleatorio generado se empezará a comparar de izquierda a derecha con los valores distintos de cero de ese renglón, hasta que se encuentre uno que sea mayor o igual. La nota de la columna donde esté almacenado este número, indica la segunda nota de la composición y el siguiente renglón a procesar.

-Se escoge el renglón correspondiente a la nota anterior y nuevamente se genera un número aleatorio entre cero y el valor contenido en la columna totales de ese renglón. Se compara de izquierda a derecha con los valores diferentes de cero almacenados en las casillas de ese renglón, hasta que alguno de ellos sea mayor o igual al número aleatorio. La columna indica la siguiente nota de la composición y el renglón al que se le aplicará el mismo proceso, que se repite indefinidamente.

Por ejemplo, tómese al azar una nota, *Fa* por ejemplo, será la primer nota de la melodía; se toma el valor de la columna *totales* del renglón *Fa*, en este caso 270 (véase la fig. 2, matriz de frecuencias acumuladas) y se genera un número aleatorio entre cero y este número. El número aleatorio así obtenido, se compara en orden de izquierda a derecha, con los valores del renglón *Fa*; al llegar a un valor que sea mayor o igual al número aleatorio, se toma la nota de la columna como la nota segunda de la melodía. Por ejemplo, si se obtiene el número 157, como este es mayor que 145 (nota Sol) y menor que 240 (nota La), se toma la nota *La* como la segunda nota de la melodía, llevando *Fa, La,...*

Se repite el procedimiento anterior, sólo que ahora se toma el renglón de la nota anterior, *La*, que tiene en la columna totales un valor de 50, por lo tanto, se genera un número aleatorio entre 0 y 50; por ejemplo el 18, 35 es mayor que 18, luego *Re* es la tercer nota de la melodía y así va *Fa, La, Re,....* Para obtener la 4ª, 5ª, 6ª y n-ésima nota, basta repetir el mismo procedimiento:

-Se genera un número aleatorio entre cero y el valor de la columna totales del renglón de la última nota obtenida.

-Se compara el número así obtenido con los valores de ese renglón, hasta que alguno sea mayor o igual. La columna en la que se encuentre este valor, representa la siguiente nota de la melodía.

2. Componente evolutiva

A continuación se explicará el funcionamiento de la componente evolutiva del sistema, que le da la capacidad de componer música de cualquier tipo, lo que es una ventaja; ya que, se pueden implementar diferentes tipos de algoritmos para la generación de música, sin embargo, habría que desarrollar algoritmos específicos para cada tipo de música y más complejo aún, habría que hacer un algoritmo específico que refleje las características propias de composición de cada autor, convirtiendo a esta en una tarea que sin duda alguna requeriría mucho de tiempo de programación.

Por lo tanto, la gran ventaja de utilizar las técnicas evolutivas, es que permiten que sea el mismo sistema quien encuentre las reglas de composición y de esta forma, para que el sistema aprenda a componer como algún autor en especial, basta con proporcionarle ejemplos de composiciones musicales de dicho autor, evitando el arduo trabajo de desarrollar miles de algoritmos.

Este sistema evoluciona en base a dos procesos, uno de *aprendizaje*, que le da la *capacidad de aprender a través de ejemplos de música* y por otro lado un proceso de aplicación que mientras el sistema genera alguna composición musical *le permite irse modificando a sí mismo en tiempo real*, como si la música generada fueran nuevos ejemplos de música que se le estuvieran proporcionando.

2.1 Proceso de aprendizaje

La función que desempeña el *proceso de aprendizaje* es encontrar los valores apropiados con los que se han de llenar las matrices evolutivas, esto es, qué casillas deberán de contener valores y qué valores se deberán de almacenar en ellas, de manera tal que reflejen las características de algún autor o algún tipo de música en particular.

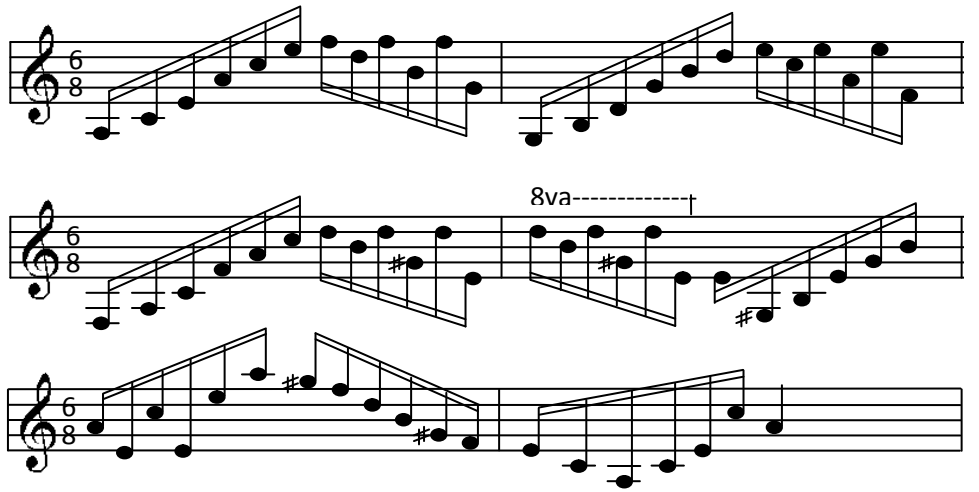
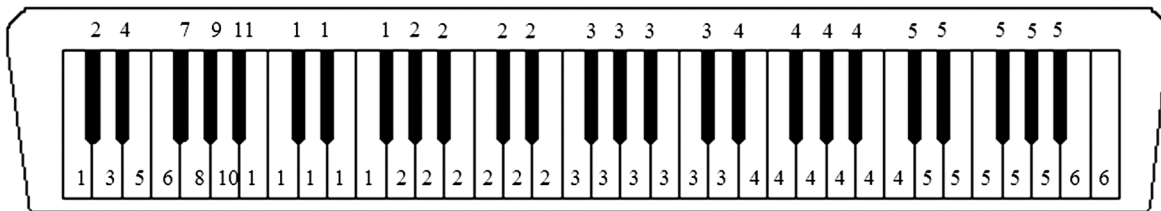


Fig. 3 Fragmento de la obra Los 24 Caprichos de Niccoló Paganini.

Para explicar el proceso de aprendizaje se utilizará un breve fragmento de la obra “Los 24 caprichos” del maestro Nicoló Paganini.

Basándonos en el siguiente teclado,



los números correspondientes a las notas de esta composición son los siguientes:

10(La),13(Do),17(Mi),22(La),25(Do),29(Mi),30(Fa),27(Re),30(Fa),24(Si),30(Fa),20(Sol),8(Sol),12(Si),15(Re),20(Sol),24(Si),27(Re),29(Mi),25(Do),29(Mi),22(La),29(Mi),18(Fa),6(Fa),10(La),13(Do),18(Fa),22(La),25(Do),27(Re),24(Si),27(Re),21(#Sol),27(Re),17(Mi),15(Re),12(Si),15(Re),9(#Sol),15(Re),5(Mi),17(Mi),9(#Sol),12(Si),17(Mi),21(#Sol),24(Si),22(La),17(Mi),25(Do),17(Mi),29(Mi),34(La),33(#Sol),30(Fa),27(Re),24(Si),20(Sol),18(Fa),17(Mi),13(Do),9(#Sol),13(Do),17(Mi),25(Do),22(La).

Conforme se vaya avanzando en el ejemplo se podrá observar que el aprendizaje consiste en encontrar las notas que ha utilizado el maestro Paganini, así como contar las veces que se repiten y saber que notas ha usado después de otras con mayor regularidad.

Al iniciar el proceso de aprendizaje, la matriz evolutiva aleatoria se encuentra vacía, es decir llena de ceros.

El primer renglón de la Matriz Evolutiva Aleatoria del sistema (ver fig. 4) que sufrirá algún cambio, es el que corresponde a la primer nota del fragmento, es decir el renglón 10 equivalente a la nota *La*; recorriendo este renglón de izquierda a derecha, se encuentra la columna correspondiente a la nota *Do*, segunda nota del fragmento, que es la columna 13 de la matriz.

Entonces, el valor de la casilla que se encuentra en la intersección del renglón 10 (*La*) con la columna 13 (*Do*) se incrementa en una unidad. Tomando en cuenta que el valor anterior era cero, queda almacenado el valor uno. De esta forma se ha incrementado la posibilidad de que la nota *Do* sea tocada después de la nota *La*.

Para continuar, se utiliza la nota *Do* como el número del siguiente renglón que se ha de modificar. Es decir, en el renglón 13 (*Do*), la segunda nota leída, se incrementará el valor de la casilla, ubicada bajo la columna que corresponde a la tercer nota leída 17 (*Mi*) en una unidad.

Se repite el mismo procedimiento, modificando en esta ocasión el valor de la casilla que se encuentre en la intersección del renglón 17 (*Mi*) con la columna 22 (*La*), que es la cuarta nota del ejemplo.

El mismo procedimiento se lleva a cabo con el resto de la melodía y de igual forma se procede con todos los ejemplos musicales que se le proporcionen al sistema. De esta manera, la matriz se va modificando conforme se le vayan presentando ejemplos.

Después de haber leído toda la partitura, se tiene la matriz lista para calcular la matriz de frecuencias acumuladas, la columna de *totales* y finalmente permitir que el sistema produzca su primera composición. La matriz que se presenta en la fig. 4, muestra como quedan los valores de las diferentes notas después de haber aprendido el fragmento de Paganini. (Debido a que la matriz que utiliza el sistema es de 60 renglones por 60 columnas, no fue posible incluir toda la información contenida en la matriz, sin embargo sólo se han quitado algunos renglones y columnas llenos de ceros).

2.2 Proceso de aplicación

La rutina generadora de música, se encarga de llevar a cabo el *Proceso de Aplicación*, dentro del cual, cada vez que se genera una nota, se modifican las matrices evolutivas, de manera tal que aumenta la probabilidad de que esa nota vuelva a ser tocada, tal y como sucede cuando un músico aprende a tocar.

Un ser humano, tiene que practicar durante “largas horas” para poder interpretar música en algún instrumento musical, repitiendo una y otra vez ejercicios y melodías, con el fin de aumentar su habilidad y de grabar en su memoria los ejercicios y melodías practicadas. De esta forma, las notas que más utiliza son de las que más se acuerda, de hecho, después de mucho practicar puede llegar a tocar sin tener que voltear a ver que nota está tocando.

De igual manera, *lo que hace el generador de música cada vez que genera una nota, es aumentar el valor de la probabilidad de la casilla que corresponde a esa nota, provocando que el sistema evolucione indefinidamente* y que las notas que más toca sean de las que más se acuerde.

En la fig. 5 se muestra el pseudocódigo del proceso de aplicación, donde se puede ver como se modifican las matrices mientras se genera la música, propiciando la evolución de ésta.

En la fig. 6 se presenta la partitura de una composición producida por el sistema, después de haber aprendido 4 ejemplos de Paganini.

```

MusicaPorcentual()
{
    a1 = nota_siguiente
    zz = random( Bethoven [a1] [62] )+1;    //Genera un número aleatorio entre el valor almacenado en el
                                           //(renglón a1, columna 62) y cero.

    a2 = 1;
    while( Bethoven[a1][a2] < zz && a2<=60 ) a2++;    //encuentra la columna a2 del renglón a1, que contiene
    nota_siguiente = a2;    //un valor mayor o igual al número aleatorio, el
                           // número de esa columna es la siguiente nota.

    Bethoven1[a1][a2] += 1;    //Actualiza (evoluciona) la Matriz Evolutiva Aleatoria.

    if( Bethoven1[a1][a2] > 500 )
    {
        for( a1 = 1; a1 <= 60; a1++ )
            for( a2=1; a2 <= 60; a2++ )
                if( Bethoven1[a1][a2] > 1 )
                    Bethoven1[a1][a2] /= 2;    //Mecanismo de olvido, cada vez que algún valor de la
                                                //matriz llega a 500, a todos los valores se les divide
                                                //entre 2 y aquellos menores a 1 se igualan a 1.
                else
                    Bethoven1[a1][a2] = 1;
        EscalaPorcentual();
    }
    while ( a2 <= 62 )
    {
        if( Bethoven[a1][a2] != 0 )
        {    //Actualiza (evoluciona) la Matriz Evolutiva de
            Bethoven[a1][a2] += 1;    // Frecuencias Acumuladas
        }
        a2++;
    }
}

```

Conclusiones

Hoy día que se descubre que el Caos hace presa de todos los habitantes de este planeta y quizá de este Universo, resulta muy interesante utilizar herramientas como los Sistemas Evolutivos que generan caos de manera cotidiana y permiten modelar los fenómenos de manera maravillosa.

La versatilidad de los Sistemas Evolutivos ha permitido el desarrollo de Zanya, un sistema compositor que da la oportunidad de escuchar composiciones caóticas por siempre diferentes, pero similares a las de ejemplos musicales de los cuales tiene la capacidad de aprender. Esto provoca un efecto maravilloso, pues abre la posibilidad de escuchar como si estuvieran presentes a autores como Bethoven o Paganini.

Por otro lado, Zanya se puede usar como un apoyo para aprender a hacer composición musical, con la ventaja de contar con la asesoría directa de los grandes maestros o de cualquier autor que uno prefiera.

Queda abierto el camino, ya que este trabajo es parte de un proceso general de composición, en donde se busca mejorar los diferentes sonidos y efectos que se puedan obtener, además de utilizar matrices de 3, 4 ó n dimensiones en donde se puedan reflejar las tantas variables que intervienen en una obra maestra.

Este breve trabajo se verá completo si logra inspirar a alguien a hacer desarrollo musical-informático, para que pueda crecer esta área que se puede considerar nueva.

Conclusión: Rumbo a la Competitividad Internacional en Informática

Fernando Galindo Soria¹

Introducción

La Informática es una de las áreas de mayor desarrollo a nivel mundial y su mercado es de cientos de miles de millones de dólares, sin embargo prácticamente pocos compiten con sus productos en ese mercado internacional.

La anterior situación se da porque la mayoría de los involucrados en el área de la informática nos dedicamos principalmente a comprar las herramientas informáticas (computadoras, satélites, líneas de comunicación, etc.) y no tenemos una contraparte que produzca y distribuya mundialmente este tipo de herramientas, somos simples importadores y usuarios de una tecnología que no dominamos ni desarrollamos.

Y lo que es más grave, se cree que nos estamos modernizando tecnológicamente sólo porque compramos e instalamos una gran cantidad de herramientas para automatizar fábricas o empresas. Es como creer que en un país está desarrollado porque compra y tiene automóviles de lujo último modelo, aunque no tengan ni idea de como se construyen y sólo sepan manejarlos.

En la actualidad un país desarrollado tecnológicamente, no es aquel que compra las mejores herramientas tecnológicas, sino el que las produce, por lo que es una falacia decir que nos estamos modernizando porque compramos herramientas modernas y las metemos a nuestras empresas y fábricas.

I Desarrollo informático

Se requieren acciones que nos permitan llegar a un nivel real de competencia dentro de la economía mundial de las nuevas tecnologías y en particular de la Informática. Algunas de estas acciones incluyen:

- 1) *El fomento a la investigación y desarrollo de tecnología avanzada.*
- 2) *La creación de una base de industrias sobre nuevas tecnologías.*
- 3) *La tecnologización e informatización de la sociedad.*

¹ Fernando Galindo Soria es profesor-investigador de la Escuela Superior de Computo (ESCOM) del Instituto Politécnico Nacional (IPN)

Resumidas en la triada:

- 1) *Investigación.*
- 2) *Industrialización.*
- 3) *Informatización.*

Y como centro de esto, el desarrollo de un proceso educativo que nos apoye en la formación de los agentes de cambio que nos ayuden a realizar las otras acciones y funcionen como núcleo del cambio.

Necesitamos contar con gentes capaces de hacer investigación, desarrollar industrias y realizar procesos de informatización, ya que, si solo se cuenta con investigadores, éstos seguirán diluyéndose en la inmensidad de los usuarios, una industria no puede subsistir sin una base de investigación y tanto unos como otros, requieran de un país tecnológizado.

Para lo cual, se requieren escuelas donde se les forme. Por lo que, el objetivo general de una institución académica debe contemplar la formación de esas gentes. Es decir que, debemos incluir dentro de las currículas actividades que apoyen la formación de los investigadores, industriales, informatizadores y educadores, además estas actividades se deben integrar en forma natural dentro del proceso académico de los estudiantes y no como un agregado lateral.

II Globalización integral

Por otro lado, si no queremos ser arrasados tenemos que tomar en cuenta que estamos en un proceso de globalización, asumir que ese proceso no tiene que ver con nosotros es como la política del avestruz que por esconder la cara cree que no pasa nada.

El mundo ya está aquí y si no somos copartícipes de este proceso otros tomarán las decisiones por nosotros y nos impondrán su visión del futuro, con lo que, más que un proceso de globalización en el que todos participemos, se puede convertir en un proceso de transculturización, en el cual otros nos impongan su manera de pensar, su cultura, su tecnología y su visión del mundo y en el cual nosotros seamos simples seguidores pasivos.

Necesitamos integrarnos en forma activa en este nuevo espacio con nuestras propias ideas y experiencias, buscando que, tanto nuestra cultura base, como las locales, no desaparezcan, sino que trasciendan, sean conocidas por otras comunidades y contribuyan e impacten en la creación de los entornos mundiales.

Por lo que una estrategia debe desarrollarse en tres planos integrados:

- 1) *Regional.*
- 2) *Nacional.*
- 3) *Internacional.*

Lo cual significa que debemos interactuar en forma natural dentro de estos niveles y la formación de los estudiantes debe contemplar su integración e impacto en los tres planos.

Ya no existe diferencia real entre estos tres espacios, un producto que penetra en el mercado global en forma natural esta dentro de los entornos locales. Por lo que, al hablar de entornos mundiales no nos referimos exclusivamente al espacio internacional sino también al nacional y al local.

Algunos argumentan que si a duras penas podemos con el espacio nacional, como podemos pensar en el internacional. Lo anterior es un error porque un producto que compite únicamente en un espacio local tiene muy poca posibilidad de sobrevivencia, en cuanto una empresa internacional penetra en ese espacio el producto local es desplazado fácilmente por el internacional.

Podemos construir el mejor manejador de bases de datos del mundo, pero si solo lo usamos en nuestra comunidad, en el momento que entra un paquete de manejo de datos a nivel internacional, tarde o temprano tenemos que usarlo o corremos el riesgo de que nuestras aplicaciones queden rápidamente desplazadas, porque dejan de ser compatibles con los estándares internacionales.

Una opción que aumenta la posibilidad de sobrevivir es pegarnos al carro del mundo y que nuestros productos sean compatibles con los estándares y plataforma marcados por otros.

Pero lo que nos da mayor posibilidad de competencia se centra en trabajar asumiendo que nuestro mercado es el mundo, en su momento, si nuestro producto es bueno no nos tenemos que preocupar de que la competencia nos desplace del espacio local ya que sí ellos están en nuestro mercado, nosotros estamos en el suyo y en su momento el estándar podría ser marcado por nosotros si logramos absorber una masa crítica del mercado.

Es difícil establecer una masa crítica de usuarios de un producto sólo con el mercado nacional, por ejemplo, nosotros somos alrededor del 0.5% del mercado mundial por lo que, como dicen por ahí, ni sumamos.

Aunque nuestro producto sea mejor, en el momento que entra el estándar de mercado es desplazado. O sea que, de todos modos Juan te llamas, queramos o no a nivel local ya estamos compitiendo en mercados internacionales, si no queremos que desplacen a nuestros productos, aunque sean locales, tienen que ser desarrollados con niveles internacionales de calidad y competitividad, ya que, el mercado local es una parte natural del mercado global y no es un espacio separado e independiente.

III Industrialización

Podremos hablar de un proceso de modernización tecnológica cuando entre otras cosas tengamos industrias que desarrollen y compitan con productos de alta tecnología tanto en el mercado interno como en el mercado mundial.

Por lo que se necesita una política de desarrollo tecnológico basada no en la importación de nuevas tecnologías para modernizar las fábricas sino en la construcción de las industrias del futuro, donde se tenga un campo real de trabajo y donde se produzcan los productos de nueva tecnología que nos permitan competir y vivir en el mundo del futuro.

Tenemos que crear un futuro mejor porque es cruel pensar que nuestros hijos (sí, esos niños de 5 años) cuando estudien Informática o Computación van a terminar únicamente como paqueteros, instaladores de redes o lo que este de moda en ese momento; los biólogos, químicos o biotecnólogos pueden terminar como vendedores de medicinas o prótesis.

En fin, los ingenieros mecánicos o electrónicos pueden terminar como técnicos en reparación de automóviles, televisión o computadoras, etc., no porque quieran trabajar en eso, sino porque no tienen otra alternativa a menos que construyamos las industrias donde ellos se puedan desarrollar.

Tenemos que crear industrias que generen productos informáticos, de biotecnología, de nuevos materiales, etc. y que no simplemente usen esos productos. Si no se hace ahora de todos modos el problema queda vigente y la bronca de crear las industrias será para nuestros hijos, pero cada vez va a ser más difícil.

Urge un cambio de mentalidad, pero no solo del gobierno sino de todos y cada uno de nosotros y dejar de pensar en volvernos expertos en el siguiente producto de moda y mejor pensar en construir nuestros propios productos a nivel de competencia internacional.

IV Estrategia industrial

En general un ciclo industrial consta de tres fases fundamentales:

- 1) *Investigación y desarrollo.*
- 2) *Producción.*
- 3) *Distribución y ventas.*

Y estas tres líneas se tienen que manejar en paralelo ya que si falta alguna es difícil competir realmente en el mundo.

1 Investigación y desarrollo

De estos aspectos el enfocado a la investigación y desarrollo es un punto que múltiples veces ha demostrado su bondad, durante muchos años se ha visto como los investigadores han desarrollado productos del mas alto nivel de competencia internacional y en algunos casos han hecho el primer ejemplo o el mejor sistema desarrollado a nivel mundial (paquetes de graficación, manejadores de base de datos, reconocedores de imágenes, generadores de sistema, etc., etc.)

También en forma cotidiana muchos de esos trabajos se han guardado o tirado a la basura y rebasados tarde o temprano por el desarrollo mundial mientras nosotros nos hemos quedado nomás milando.

Y la gente que los desarrolló ha terminado en muchos casos como expertos en paquetería (algunos han llegado a puestos de dirección pero se han vuelto directores de expertos en paquetería) o han continuado desarrollando el siguiente producto mágico y el siguiente y el siguiente, sin salir nunca al mercado.

Es triste ver la cantidad de estudiantes que se forman en nuestras universidades y que van a terminar como doctores en paquetería (sino, observen las ofertas de empleo nacionales y verán que la mayoría solicitan expertos en paquetería.).

Es cruel verlos construyendo robots, sistemas de control distribuido, juegos por computadora, etc., sabiendo que fuera de las escuelas no interesa que sepan eso (busquen cuántos están solicitando expertos para desarrollar productos de tratamiento de imágenes, proceso distribuido, realidad virtual robótica, etc.), sólo requieren que manejen el paquete de moda.

Con lo que la opción trivial es tirar todo a la basura y dedicarnos a formar buenos paqueteros, pero además de tonta, esta opción es suicida, ya que, el mundo migra rápidamente a las nuevas tecnologías. Sólo basta observar las revistas internacionales en el área como el Communications of the ACM para darnos cuenta de la otra cara de la moneda y ver un mundo donde se pelean por esos niños genios.

O sea que, por el lado de la investigación y desarrollo tenemos mucha capacidad, pero necesitamos mecanismos que nos permitan encontrar y recuperar esos resultados, incluyendo la creación de grupos de exploradores y cazadores especializados en localizar nuevos productos y en la transición de los resultados de los grupos de investigación a los centros de promoción y distribución, ya que, esta creatividad se está perdiendo y no existen los espacios donde se pueda capitalizar.

2 Producción

En el aspecto de producción, la mayoría de los productos informáticos no requieren de muchos recursos, sin embargo, lograr que una investigación termine en un producto de mercado puede costar cientos de veces mas que hacer el prototipo.

Necesitamos los centros de producción donde se tenga la infraestructura que permita producirlos a un costo accesible para pasar de un prototipo a miles o cientos de miles de copias (normalmente un investigador aislado no la tiene, muchas veces ni siquiera sabe que se necesita y mucho menos tiene la capacidad para adquirirla), de tal manera que el precio final al consumidor sea competitivo para ventas masivas.

En este punto se presenta otro de nuestros errores, muchas personas que se desarrollan independientemente en el área piensan en una sola copia cada vez y cobrar miles de dólares por cada solución, aplicación o sistema específico.

Esta es una visión muy local, cuando se piensa en mercado internacional se debe pensar en términos de miles o cientos de miles de copias y en venderlas en unos cuantos dólares para que lleguen a mercados masivos.

3 Distribución

Es ahí donde entra el tercer aspecto, una gran parte del problema de desarrollar una industria internacional se centra en su capacidad de penetración, y específicamente en su capacidad de distribución y mercadotecnia, por lo que, necesitamos aprender a competir con nuestra propia tecnología, cultura y pensamiento a nivel mundial.

Es necesario contar con las líneas de distribución adecuadas y con una mercadotecnia agresiva en el entorno internacional, ya que si no, es difícil penetrar en el mundo.

Este último punto es nuestro Talón de Aquiles, ya que no contamos con estas líneas de penetración (recuerden que somos usuarios y no desarrolladores, o sea que no nos hemos preocupado de crear esas líneas) por lo que es urgente empezar a trabajar sobre este aspecto.

Como se puede ver no es raro que lo que se hace en los grupos de investigación se pierda, si no contamos con la infraestructura, ni la experiencia de producción, ni de distribución y mercadotecnia.

Aunque inicialmente un producto tiende a penetrar el mercado local no necesariamente es así, ya que eso depende de la estrategias de distribución, por lo que necesitamos redes de distribución y mercadotecnia que en forma natural nos permitan penetrar en los mercados locales y globales.

Por ejemplo una empresa que cotidianamente vende en mercados internacionales, en forma natural introduce los nuevos productos en esos mercados.

Lo interesante es que ya existe una comunidad industrial madura en el país y mucha de esta infraestructura ya existe para los productos tradicionales, por lo que tal vez, lo único que se requiere es que se adquiera conciencia de que no tenemos por que ser simples receptores de la tecnología.

Conclusiones

Necesitamos adquirir conciencia de que, los países desarrollados tecnológicamente no son los que compran la tecnología para estar a la moda, sino los que la desarrollan.

Por lo que, debemos lograr que se construyan industrias integradas a su ambiente, que desarrollen productos de alta tecnología y los distribuyan en forma natural a nivel local, nacional e internacional.

Y debemos de luchar por llegar a ser realmente actores del proceso mundial y no simples receptores de algo que tal vez nos esta destruyendo sin saber y sin posibilidad de actuar para cambiar o revertir, ya que no somos copartícipes del futuro.