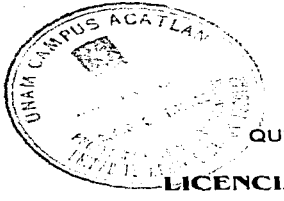
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ACATLAN

"ALGUNAS APLICACIONES DE LAS REDES SEMANTICAS"



TESINA

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

ELIZABETH INFANTE COLINA

ASESOR : ING. RUBEN ROMERO RUIZ

NAUCALPAN DE JUAREZ, EDO.MEX. 1997

A las Escuelas:
Que me han permitido la prepación con sencilles y respeto.

A los Maestros:
Por la enseñanza y la orientación que me han dado.

A mi Familia:
Como una muestra de mi cariño y agradecimiento, por todo el amor y el apoyo brindado; les agradezco la orientación que siempre me han otorgado.
Con admiración y respeto.

SUMARIO

INTRODUCCIÓN

Nació en mí la inquietud de investigar en qué parte de la Inteligencia Artificial se utiliza lo estudiado en la carrera , en especial los tres últimos semestres como lo fue en las asignaturas:

A) Teoría de la computación I, con el tema Gramáticas enfocadas a los autómatas.
B) Teoría de la computación II, con el tema Máquina Universal de Turing.
C) Compiladores, con el tema Gramática Independiente del Contexto.

En ocasiones la profesora Judith Jaramillo López que impartió la materia de Teoría de la computación I, nos indicó que las gramáticas se utilizan en la Inteligencia Artificial, lo cual me ayudó a decidir investigar por mi parte sobre lo dicho por la profesora.

Recurrí a distintos libros relacionados con la Inteligencia Artificial, en los cuales me encontré que una de las herramientas para su desarrollo son las redes semánticas, que sirven para representar conocimientos a través de las producciones para poder llegar a deducciones.

Me dediqué a buscar en las tesis de la carrera el tema que me interesaba, acudí a la hemeroteca y encontré los siguientes temas relacionados con la Inteligencia Artificial:

1. Planteamiento de un sistema experto para la obtención de indicadores diagnósticos que determinen una posible alteración neurológica.
2. Forma de aprendizaje de las redes neuronales aplicando el algoritmo regla delta generalizada a la representación del conocimiento.
3. Inteligencia Artificial:Representación de conocimiento.
4. Principios del lenguaje natural por computadora con aplicación a la generación de programas D'Base.
5. Diseño de un sistema experto para el diagnóstico y tratamiento del edema cerebral causado por traumatismo craneoencefálico.

Con esto vi que el tema *'Algunas aplicaciones de las redes semánticas'* es un tema nuevo en la carrera.

Esta Tesina reúne y explica algunas aplicaciones de las redes semánticas, que se desarrollan para la representación tanto de ciertas teorías de la computación, como de herencias en este campo. Para lograr este fin se utilizan ejemplos gráficos que ayuda al lector a adquirir una mejor comprensión del trabajo.

A lo largo de este trabajo se usan términos computacionales, por esta razón se orienta esta Tesina a lectores que tengan desde conocimientos básicos de computación en adelante.

CAPITULO I

INTELIGENCIA ARTIFICIAL

I.1 INTRODUCCIÓN

Durante mucho tiempo, la gente soñaba con tener una máquina que le facilitara tareas (tales como cálculos matemáticos, almacenamiento de información y su recuperación rápida entre otras tareas manuales que eran tardadas), pero nunca se imaginó que esto se haría realidad con las ideas y sueños de muchos investigadores que serán mencionados en este capítulo.

Podemos ver que uno de los avances que favoreció en gran medida a las empresas relacionadas con números fue la calculadora, y posteriormente la computadora. Simplemente con la calculadora la gente común sentía que era el más grande avance, sin saber su funcionamiento interno. Cuando llegó la computadora la gente sintió miedo al usarla, ya que no cualquiera la podía manejar, sin saber de lenguajes de programación , del software y hardware. Con las escuelas de computación estas máquinas llegaron a formar parte de la vida diaria, tanto que actualmente a los niños de pre-primaria se les enseña a utilizarla.

Gran parte de la investigación se dirige en la actualidad al desarrollo de las computadoras que forman parte de los sistemas artificiales, para que puedan operar en la misma forma en que la inteligencia humana lo hace.

Hablaremos de lo que es la inteligencia en los seres humanos, ya que pueden darnos una idea sobre las posibilidades y limitaciones de las máquinas.

Solemos imaginar que la inteligencia consiste en la habilidad para resolver problemas, planear, hacer sumas, aprender, salir airoso ante lo desconocido. En cierto modo, al hablar diferenciamos tipos de inteligencia. Así hablamos de juicio, sabiduría, perspicacia, percepción y aprendizaje. Veremos cómo estos matices son importantes para comprender la Inteligencia Artificial.

La inteligencia biológica ha evolucionado a lo largo de millones de años. Desde la infancia hasta la madurez, el ser humano desarrolla y amplia sus capacidades cognoscitivas.

Una evolución similar es discernible en la inteligencia computarizada, que como se hablará posteriormente está basada en el modelo de la inteligencia del ser humano. Una de la críticas a la Inteligencia Artificial es que sus programas van dirigidos a tareas específicas e independientes -juegos, demostración de teoremas, narración de historia, etc.- y carece por lo tanto de universalidad que caracteriza a la auténtica inteligencia.

I.2 PLANTEAMIENTO COGNOSCITIVO

El modelo cognoscitivo de la mente humana está basado en la idea del proceso de la información.

"La información del mundo exterior incide sobre los sentidos receptores los cuales captan los cambios en fenómenos naturales tales como la presión del aire, temperatura, la distribución de las moléculas, y la radiación electromagnética. En un principio la información se almacena a corto plazo sensorialmente en la memoria sensorial intermedia. Esta memoria tiene gran capacidad ya que puede guardar toda la información recibida por los receptores sensoriales, pero únicamente por un intervalo de tiempo muy corto.

En función de necesidades inmediatas del individuo, este atiende o ignora la información que por un instante guardó en la memoria sensorial. La memoria a corto plazo tiene bastante menos capacidad que la memoria sensorial intermedia. La memoria de trabajo funciona como un block mental de notas sobre el que se pueden efectuar tareas intelectuales; al igual que la memoria a corto plazo, la información se borra rápidamente de la memoria de trabajo si no se adopta una estrategia de repetición mental. Se puede transferir información desde la memoria a corto plazo a la memoria a largo plazo. La memoria a largo plazo puede guardar grandes cantidades de información y poca o ninguna información se desvanece con el tiempo. Si la memoria está despejada, se puede mejorar la capacidad de recuperación de información mediante estrategias de búsqueda"[1]

Con lo anterior podemos darnos cuenta de que en la Inteligencia Artificial se emplea una estructura de memorias similar a la usada por la inteligencia humana. La memoria de trabajo RAM (Random Access Memory) permite el almacenamiento temporal, tanto escritura como lectura que al apagar la computadora desaparecen. En el caso del ser humano la información almacenada en la memoria de corto plazo se borra si no se le hace pasar (mediante la repetición) a la memoria de largo plazo; de forma parecida surge la necesidad en las computadoras de almacenar programas y datos a largo plazo nos dá como consecuencia el uso de un almacenamiento secundario, capaz de guardar información hasta que vaya a procesarse teniendo como medios de almacenamiento secundario el disco magnético, disco compacto, entre otros.

Así nos damos cuenta que hay estrategias de búsqueda de la información como sucede en la mente humana, pero la diferencia que hay entre los procesos de la mente humana y de la computadora, es que el almacenamiento de la información en una computadora está condicionada a la capacidad del almacenamiento primario y secundario, lo que no sucede con la mente humana ya que como se mencionó antes puede guardar grandes cantidades de información y poca o ninguna se desvanece con el tiempo.

[1] Simons G. L. , "Introducción a la Inteligencia Artificial", Ed. Díaz De Santos, México 1990, pag .20

I.3 LA IMPORTANCIA QUE TIENE EVALUAR LA INTELIGENCIA NATURAL PARA LA INTELIGENCIA ARTIFICIAL.

Desde el planteamiento de la Inteligencia Artificial resulta interesante poder medir la inteligencia natural. Los intentos de medir o cuantificar la inteligencia han facilitado su eficaz implementación en sistemas artificiales. Los primeros test de capacidad mental fueron efectuados por Sir Francis Galton[2] (1883) quien intentó demostrar que los atributos mentales eran heredados y por J. Mck Catell colaborador de Galton en los Estados Unidos, con el fin de evaluar parámetros sicológicos tales como tiempos de reacción, agudeza visual y auditiva, sensibilidad al dolor, memoria y representaciones cognoscitivas.

Alfred Binet[3] (1857-1911), gran pedagogo y presidente de la Sociedad Libre para el Estudio Psicológico del Niño en Francia, sugiere que es equivoco considerar la inteligencia como atributo inalterable de la persona.

La tarea de evaluar la inteligencia de una máquina puede ser técnicamente difícil. Actualmente hay computadoras inteligentes, pero lo importante es saber cuál es su inteligencia real y qué nivel de inteligencia pueden alcanzar. La capacidad de los sistemas artificiales, no se puede evaluar emotivamente y mucho menos en un contexto carente de imparcialidad.

Algunos aspectos de la inteligencia natural como son intuición, conocimiento, no son aún comunes en las computadoras, las cuales son corrientes máquinas que pueden recordar, computar e iniciar acciones destinadas a cumplir determinados objetivos, pero sorprendentemente, también existen en la actualidad sistemas artificiales capaces de aprender, desempeñar actividades creativas, o manipular información para sintetizar nuevos conocimientos.

También podemos ver que el aprendizaje depende de la capacidad de memorizar un elemento clave en los sistemas inteligentes biológicos y artificiales.

Las computadoras están aprendiendo a aprender, principalmente mediante programas heurísticos y adaptables que permiten a las máquinas aprender de la experiencia. El aprendizaje comprende varias etapas: adquisición de información, retención, generalización y retrospección, asimilación y aplicación como resultado del aprendizaje. Casi todas las actividades en Inteligencia Artificial tienen una connotación cognoscitiva, es decir, se asemejan al proceso de información relativo a la aceptación de los datos, rememorización, resolución de los problemas, etc.

---

[2] Simons G. L. , "Introducción a la Inteligencia Artificial", Ed. Díaz De Santos, México 1990.

[3] Alfred Binet, "Las ideas modernas sobre los niños", Ed.Fondo de cultura económica, S.A de C.V., México 1985, pgs.358.

I.4 HISTORIA DE LA INTELIGENCIA ARTIFICIAL

La Inteligencia Artificial suele ser considerada como una rama más de la Computación. También se ha considerado conveniente excluir ciertas actividades informáticas tradicionales (las típicas del proceso de datos) del campo de la Inteligencia Artificial.

Ahora hablaremos brevemente de los inicios de la computación como antecedente de la inteligencia artificial.

La computadora nace de la necesidad de crear máquinas que calcularan operaciones que simplificarían el trabajo de los científicos.

Charles Babbage, nació en 1792, suele ser considerado el padre de la moderna informática. Inventó dos máquinas de calcular, posteriormente desarrolló con relativo éxito dos máquinas : la máquina diferencial y analítica. Ello fue muy importante para el futuro de la computación electrónica, pues quedaron demostrados, aunque en términos mecánicos, los componentes esenciales en cualquier sistema de computación de propósito general : entrada o introducción de los números en la máquina, almacenamiento, unidad aritmética para efectuar los cómputos, la unidad de control para controlar la ejecución de las diferentes tareas bajo la dirección del programa almacenado, y la salida para poner los resultados del proceso a disposición de los usuarios.

El tabulador que diseñó el Dr. Herman Hollerith como medio eficaz para la realización del censo americano de 1890, fue la primera máquina de cómputo que usó medios no mecánicos. Su enfoque fue hacer perforaciones en una tarjeta, cada perforación generaba una corriente electrónica que hacía avanzar un paso a un contador.

La compañía IBM creó la computadora MARK I presentada en 1943, la cual usaba relés electromagnéticos. En 1946 aparece la primera computadora electrónica llamada ENIAC diseñada con válvulas termoiónicas.

Después John Von Neumann comenzó el diseño de EDVAC (ordenador electrónico automático de variable discreta). Por primera vez se incluía en el proyecto del diseño de un ordenador digital electrónico el concepto de control por programa almacenado. Von Neumann contribuyó en el tema de la Inteligencia Artificial al introducir el concepto de sistema informático.

Alan Turing contribuyó notablemente a la aparición de la Inteligencia Artificial. En 1937 publicó una ponencia sobre "números computables", donde expuso por primera vez el concepto de "máquina universal de Turing", un dispositivo teórico. En él expone que la máquina puede desarrollar cualquier procedimiento matemático siempre que se le proporcione una tabla adecuada de instrucciones (ver Capítulo III, subtema III.4 en donde se define a esta máquina).

Alan Turing tuvo la oportunidad de poner la teoría en práctica. Como preveía una invasión por parte de las fuerzas de Hitler, el gobierno británico reunió un equipo de matemáticos e ingenieros con la misión de descifrar el código alemán. Bajo la dirección de Turing, el grupo construyó Colossus, una máquina de propósito específico que en la actualidad se considera como el primer computador digital electrónico. Desde que se completó el Colossus en 1943 hasta que terminó la guerra, descifró con éxito los códigos nazis, hechos que los británicos ocultaron hasta mucho después del fin de la guerra mundial.

Turing de hecho inauguró el campo de la inteligencia artificial con su artículo "Computing Machinery and Intelligence", publicado en 1950. En donde hablaba de si una máquina era inteligente o no. El fue coautor del primer programa para jugar ajedrez.

Turing fue un individuo poco convencional y extremadamente sensible. Su vida profesional y social fue aniquilada al ser arrestado en 1952 por violación a las leyes antihomosexuales británicas. Según las apariencias, este genio de 41 años, se suicidó en 1954, tres años antes de que el gobierno hiciera del conocimiento público su heróica contribución durante la guerra.

El término Inteligencia Artificial se cree que fue utilizado por primera vez en 1956 por John McCarthy, profesor auxiliar de matemáticas de Dartmouth College en Hanover, New Hampshire, Estados Unidos. Convocó a una conferencia considerada como el comienzo de la Inteligencia Artificial como disciplina independiente de la Informática. Varios de los asistentes fueron Allen Newell, Herbert Simon, Marvin Minsky y John McCarthy.

Posteriormente Marvin Minsky, que trabajó junto con Claude Shanon en los laboratorios Bell, estimuló en el desarrollo del primer proyecto de la Inteligencia Artificial. Posteriormente John Mcarthy desarrolló el lenguaje LISP uno de los lenguajes más utilizados en la inteligencia artificial.

Actualmente se ha obtenido un avance considerable en diferentes campos de la inteligencia artificial :

- Traducción automática con vocabulario restringido o especializado.
- Desarrollo en partidas de ajedrez, póker, etc.
- Demostración de teoremas en lógica simbólica y geometría elemental.
- Lectura de Caracteres caligrafiados o mecanografiados.
- Identificación de los componentes gráficos de una fotografía o dibujo.
- Identificación de rostros humanos aun en diferentes expresiones.
- Identificación de palabras habladas en una conversación ordinaria.
- Comprensión del lenguaje natural.

Todas estas actividades en desarrollo, entre otras.

CAPITULO II

REDES SEMÁNTICAS

II.1 INTRODUCCIÓN

Como se mencionó en el Capítulo I, la Inteligencia Artificial trata de imitar a la inteligencia humana, pero para lograrlo se necesita que el sistema artificial tenga una memoria. Esto llevó a la búsqueda de una buena representación de la memoria, dando así paso al entendimiento de su funcionamiento y posteriormente su codificación a un sistema artificial.

Las redes semánticas nos ayudan a la representación del conocimiento en la memoria, de las cuales hablaremos en este Capítulo con la ayuda de ejemplos gráficos para una mejor compresión.

II.2 LA SICOLOGÍA Y EL CONOCIMIENTO.

El objetivo principal de la inteligencia artificial es modelar, asimilar, imitar, o duplicar los fenómenos sicológicos del ser humano.

Los sistemas artificiales tratan de imitar a la inteligencia humana, pero para lograrlo se necesita que la máquina del sistema artificial tenga una memoria en la cual se encuentren conocimientos, los cuales antes de utilizarlos hay que buscar su ubicación, considerando a la memoria como una colección de conceptos que van asociados unos con otros a través de una lista de características. Por lo tanto, en los sistemas artificiales existe la necesidad de codificar las reglas para manipular los conocimientos a fin de inferir otros nuevos a partir de los ya contenidos en la base de conocimientos; las inferencias y los aspectos semánticos han de ser controlados. Además de todos los requerimientos está la necesidad de desarrollar métodos de extracción de conocimientos del experto humano, para almacenarlos en la base de conocimientos. Esta es una de las primeras tareas de la Inteligencia Artificial.

Localizar los conocimientos dentro de la memoria. Ello supone retener en memoria una gran cantidad de conceptos, expresados con nombres, verbos, adjetivos, etc., y poder recuperarlos en un tiempo.

Un planteamiento inicial es que las palabras suelen estar definidas normalmente por otras palabras, ello sugiere que el significado de un concepto está en parte relacionado con otros conceptos en memoria. Por ejemplo 'Zanahoria' es un concepto que está relacionado con 'Vegetal' y este a su vez se relaciona con 'Comestible', de la misma forma cada uno de estos conceptos está relacionado con otros conceptos. Un concepto, además pertenece a una clase y tiene propiedades como se muestra en el Capítulo IV.

No cabe duda que la base del conocimiento de memoria se puede utilizar de modos diferentes, puede funcionar como una red semántica cuyos nodos representan conceptos y acciones, y crear una estructura de memorización interconectada por medio de relaciones entre nodos. La memoria se puede considerar también como una colección de conceptos, cada uno de los cuales van asociados a una lista de características.

II.3 REGLAS DE PRODUCCIÓN

Las arquitecturas de sistemas de producción son otro destacado método de representación de conocimientos en sistemas de Inteligencia Artificial. Originalmente fueron expuestos por Allen Newell en 1973, como modelos de razonamiento humano. El planteamiento de los sistemas de producción está ligado al concepto de regla, cada una de las cuales es esencialmente un par operativo "patrón -> acción" en un entorno de conceptos

relevantes, en otras palabras una regla de producción son oraciones válidas. Por ejemplo podemos representar bloques Begin-End de Pascal con una producción de la siguiente manera:

bloque → **Begin** proposiciones_opciones **end**

donde es posible leer la flecha como ' puede tener la forma'. Dicha regla se denomina producción.

En una producción se tienen el lado izquierdo que es la palabra la cual puede ser sustituída por el lado derecho de la producción el cual es señalado por la flecha.

La idea ha demostrado ser especialmente fructífera para la manipulación de conocimientos en sistemas expertos.

Un sistema de Inteligencia Artificial puede acumular conocimientos en forma de reglas de producción. Esto es útil por que el conocimiento de un especialista se puede aumentar y puede expresarse en una forma fácilmente comprensible por un experto familiarizado con la ingeniería del conocimiento. Las reglas se recogen del experto, se comprueba su coherencia, y se programa en el sistema de Inteligencia Artificial, simultaneamente se analiza qué reglas son apropiadas o inadecuadas.

II.4 REDES SEMÁNTICAS

Las redes semánticas se utilizan frecuentemente en los sistemas de la inteligencia artificial como formalismos para representar conocimientos. Para comprender que significa red semámtica se definen por separado las dos palabras:

Red: Conjunto de nodos relacionados entre sí por medio de arcos. Expresada en otros términos, puede ser un conjunto de reglas (producciones) relacionadas entre sí.

Semántica: Es la signifficación de un contexto.

Con estas definiciones deducimos que una *red semántica* es un conjunto de producciones relacionadas unas con otras formando una red qué al recorrerla podemos obtener conceptos, deducciones y herencias.

Hay muchos tipos diferentes de redes semánticas, pero todas ellas comparten una notación común, nodos y arcos (que entrelazan nodos). Los nodos (representados normalmente por cajas o círculos) indican objetos, conceptos o situaciones en un campo determinado. Los arcos representan relaciones entre los nodos. Como otros medios de la inteligencia artificial, las redes semánticas pueden representar tanto modelos sicológicos de memoria como representaciones funcionales de sistemas artificiales.

Una representación semántica sencilla se ve en la siguiente figura :

figura 2.1

Si deseamos hablar de un coche determinado, la representación se puede ampliar a la figura 2.2 como se muestra a continuación.

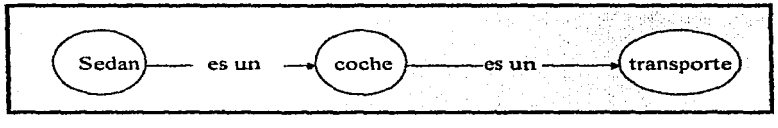


figura 2.2

Este tipo de representación facilita la obtención de deducciones válidas con sólo seguir las flechas. La facilidad con que se puede hacer tales deducciones mediante jerarquías heredadas justifica la popularidad de las redes semánticas. La representación anterior que se puede ampliar aún más incluyendo otro tipo de enlace para significar la posesión de una cualidad.

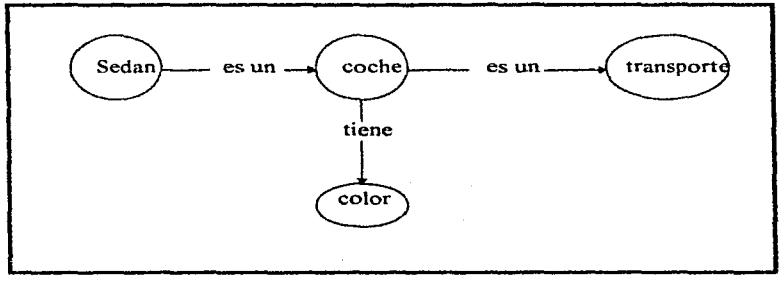


figura 2.3

De este modo indicamos que los coches tienen color y es fácil deducir que esta es una característica de Sedan, así las redes semánticas se pueden ampliar para representar relaciones complejas, entre objetos, acciones, etc., y para significar conceptos tales como "cualidad de", "poseídas por" etc.

Consultando la red semántica de la figura 2.3, se observa que la mayoría de los nodos representan objetos, pero otros nodos representan atributos de los objetos con los que se relacionan como tamaño, color o especificaciones.

Una característica importante de las redes semánticas es que algunos nodos pueden heredar propiedades o características de otros. Dado que las redes semánticas se usan para representar información jerárquica, algunos nodos se encontrarán con una escala jerárquica menor y heredarán propiedades de los que tienen más alta jerarquía.

Para resolver problemas con la ayuda de una red semántica, se formulan preguntas relativas al dominio que se está representando. El programa de inferencia recorre distintos nodos y arcos en busca de las palabras clave de la cuestión. Si el conocimiento se encuentra en el sistema, este podrá proporcionar respuestas adecuadas.

Viendo el ejemplo del esquema de la figura 2.4, una pregunta posible sería *¿Qué es un coche?* la cual dará como respuesta "*Herramienta de transporte*". La pregunta *¿De qué color es el coche?* dará como respuesta "*Negro*" *¿Quién fabricó el coche?* podría contestar "*SEAT*" o "*VOLKSWAGEN*" ya que este último tiene una orden de sucesión (herencia).

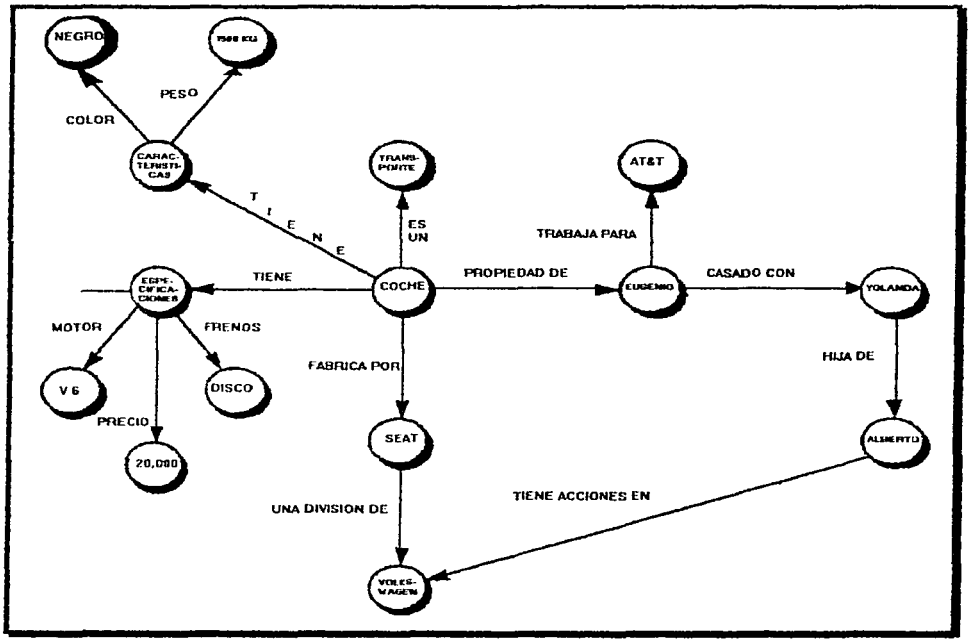


figura 2.4. Esquema de una red semántica

CAPITULO III
APLICACIONES EN LA TEORÍA DE LA COMPUTACIÓN

III.1 INTRODUCCIÓN

Los sistemas pueden ser interpretados, ya que a sus elementos se les puede asignar un significado y se pueden tomar como símbolos del mundo exterior.

Interpretar es comprender el sentido. Interpretar un sistema de señales o elementos como símbolos, es comprender el sentido de ellos dándoles significados.

Las teorías que muchos investigadores realizaron en alguna notación, las podemos transformar a reglas de producción con el propósito de ver su utilidad y poder interpretar de otro modo las ideas que ellos querían transmitir para dar paso a los procesos entrono de las computadoras actuales.

A los elementos de un sistema computacional (programa) se les pueden asignar un significado y se pueden tomar como símbolos del mundo exterior. La interpretación y la significación simbólica es la semántica; y el significado asignado, así como cualquier relación o característica que dependa de esas asignaciones son propiedades semánticas de los elementos interpretados. Por ejemplo, si 'negro' fuera un elemento interpretado como el nombre de un color en particular, entonces su relación sería una propiedad, por otro lado el elemento compuesto 'negro es un color ' refleja las interpretaciones de éste término como : verdadero o falso, siendo estas también propiedades semánticas.

Los elementos interpretados que se utilizaran para este capítulo tienen dos tipos de vida:

Vida Sintáctica : se mueve de acuerdo con las reglas producción, pudiéndose pensar que son marcadores sin sentido a simple vista, pero esto llega a cambiar con la vida semántica.

Vida Semántica: el conjunto de relaciones simbólicas llegan a tener un significado explícito para representar algunas teorías o ideas.

III.2 EL JUEGO DE LAS LETRAS DE ATHOL

Athol descubrió un juego extraño en el cual sus elementos son las quince letras del abecedario de la A a la O; al inicio del juego generó 8 renglones de combinaciones de las quince letras permitidas, a los cuales les llamó los 8 juegos como se muestra a continuación.

CUADRO III.1

Juegos *Combinación de letras*

J1	OEOA N
J2	NIBMA G
J3	HCHCHA KON
J4	KEKDOFA F
J5	MMCNA JJ
J6	OODFA OO
J7	IDLA M
J8	NBNA O

Esto lo hizo con el fin de asignarle un valor a cada letra, ya sea de un dígito (que después se tomaría como operando) ó un operador (+,-,*,/ ó =), pero para poder asignarle un valor a cada letra, él utilizó la notación de las producciones en donde el símbolo '->' significa *traducido a*, con la finalidad de crear ecuaciones con sentido.

CUADRO III.2

Producciones

A->1	F->6	K->+
B->2	G->7	L->-
C->3	H->8	M->*
D->4	I->9	N->/
E->5	J->0	O->=

Con los 8 juegos del cuadro III.1 y las producciones del cuadro III.2, generó el primer esquema de traducción que se muestra a continuación.

Primer esquema de traducción

```
J1    =5= 1   /
J2    /92x1   7
J3   838381
=/
J4  +5+4=61  6
J5    xx3/1  00
J6    = 461  = =
J7     94-1
*
J8     /2/1
=
```

Como podemos ver este primer esquema está completamente sin sentido, y es que existen 1.3 billones de alternativas para representar quince letras en diez dígitos y 5 operadores.

Se presenta otra de las tantas alternativas que hay para las producciones como veremos en el cuadro III.3.

CUADRO III.3

Producciones

A->=	F->0	K->5
B->+	G->1	L->6
C->-	H->2	M->7
D->*	I->3	N->8
E->/	J->4	O->9

Se puede apreciar que Athol pone los signos y dígitos en los lugares adecuados de tal manera que los juegos parecen ecuaciones como se ve en el segundo esquema.

Segundo esquema de traducción

J1	9/9=	
8		
J2	83+7=	1
J3	2-2-2=	98
J4	5/5*90	0
J5	77-8	44
J6	99*0	99
J7	3*6	2
J8	8+8=	
9		

Por desgracia lo que parece como ecuaciones en el segundo esquema sería falso.

Después de la búsqueda del esquema con coherencia y verdad Athol pudo llegar a un conjunto de producciones del cuadro III.4 que generan un esquema en donde sus productos no parecen ecuaciones, si no que lo son y resultan verdaderas, como veremos a continuación en el tercer esquema.

CUADRO III.4

Producciones

A->=	F->0	K->5
B->/	G->9	L->4
C->*	H->8	M->3
D->-	I->7	N->2
E->+	J->6	O->1

Tercer esquema de traducción

J1	1+1=	2
J2	27/3=	9
J3	8*8*8=	512
J4	5+5-10=	0
J5	33*2 =	66
J6	11-0=	11
J7	7-4=	3
J8	2/2=	1

Claramente este esquema justifica la idea de Athol, quien pensó que unos resultados verdaderos hacen conveniente la interpretación de los mismos, y de él nace la inquietud de representar ecuaciones con sentido verdadero a través de las reglas que son proposiciones aceptadas como ciertas llamadas axiomas, y algunos axiomas sobre los cuales se pudo haber basado como guía en el trabajo con las ecuaciones son las siguientes:

"a)Propiedad reflexiva de la igualdad:

Para todo número a, a=a

b)Propiedad de simetría de la igualdad:

Para cualquiera números a y b si a=b , entonces b=a

c)Propiedad transitiva de la igualdad:

Para cualquiera números a, b y c si a=b y b=c, entonces a=c

d)Propiedades de cerradura bajo la adición:

Para todo número a y todo número b enteros, la suma a+b , da como resultado siempre un número entero.

Para todo número a y todo número b la suma a+b es un único número.

e)Propiedad de cerradura bajo el producto:

Para toda a y todo b, el producto ab es un único número.

Para toda a entero y todo b entero, el producto ab es un número entero.

f)Propiedad multiplicativa del 1:

Para toda a, 1*a = a*1 = a

g)Propiedad aditiva del cero:

Para toda a, 0+a = a+0 = a.

h)Propiedad multiplicativa del cero:

Para toda a, a*0 = 0*a = 0

i)El conjunto de los números enteros no es cerrado bajo la división .

j)No se debe dividir entre cero, por lo tanto a/0 no tiene valor ó tiene valor indefinido."[1]

Athol utilizó las reglas de inferencia (producciones), caminando hacia un descubrimiento importante que fue los sistemas axiomáticos formales; explicando de otra forma que un sistema axiomático es un conjunto de axiomas más un conjunto de reglas de inferencia que permiten derivar teoremas.

[1] Mary P. Dolciani, "Algebra Moderna Estructura y Método", Publicaciones Cultural, S.A., México, 1989, pags. 69-84.

III.3 PROGRAMA DE LA MÁQUINA ANALÍTICA.

Babbage es recordado por sus máquinas calculadoras. En realidad diseñó dos máquinas bastantes diferentes. La primera (ca. 1823-1833), llamada Máquina Diferencial, en la cual se construyeron muchos componentes, pero la Máquina Diferencial no se terminó.

La segunda máquina fue la Máquina Analítica mostrada en el esquema de la figura 3.1, mantuvo ocupado a Babbage, su diseño incorporaba dos ideas muy profundas que juntas sientan las bases de toda la ciencia de la computación :

1. Sus operaciones son completamente programables.
2. Los programas pueden tener ramificaciones condicionadas.

Al igual que la Máquina Diferencial tampoco llegó a estar concluída aunque estuvieran un millar de dibujos técnicos, diagramas de regulación y de flujo.

Por muchos años, la única noticia publicada sobre la nueva máquina fue una pequeña memoria. En 1842 L. F. Menabrea Ingeniero italiano publicó un resumen de alguna de las conferencias que Babbage había dado en forma privada en Nápoles. Al año siguiente, Ada Augusta, una joven talentosa tradujo al inglés algunas notas explicativas más elaboradas que el artículo original.

La Máquina Analítica tiene tres componentes principales :

La fábrica (unidad aritmética lógica): Ahí se pueden efectuar cuatro operaciones aritméticas usando cualquier lugar en el almacén para los operandos y resultados, además de realizar las operaciones lógicas.

El almacén (la memoria): Es en donde se guardan los programas, operandos y resultados.

La unidad de control: Es la unidad que lleva el control de la secuencia de las instrucciones del programa almacenado, además de llevar el control de los datos que provienen de los dispositivos de entrada, así como de los datos que se deben entregar a los dispositivos de salida, y de la información que se almacenará en la memoria.

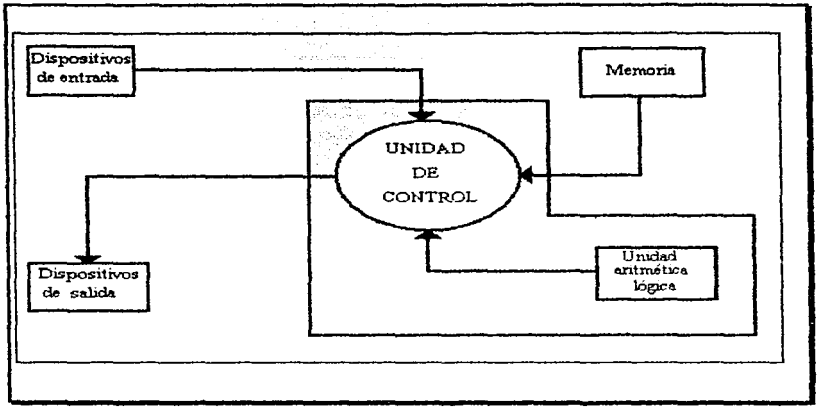


figura 3.1. ***Esquema de la Máquina Analítica de Babbage.***

Dentro de ciertos límites, la Máquina Analítica jugará cualquier cosa que le digamos, es decir, nosotros especificamos las reglas y así, solo con manejar las diferentes definiciones podemos hacer que la Máquina Analítica sea el sistema formal que queramos (dentro de sus límites). Este ejercicio de elaborar un sistema automático específico simplemente con describirlo de manera apropiada para un sistema de propósito general, a esto se le llama programar.

El segundo gran invento de la Máquina Analítica es la ramificación condicionada :

Una orden que se da para que se mueva otra parte del programa dependiendo del resultado de alguna comprobación. Los programas de Babbage son esencialmente listas ramificadas, tales ramificaciones se necesitan para muchos cálculos numéricos (ver el cuadro III.5). El control condicionado es sumamente importante, también en otros manipuladores generales de símbolos, incluyendo a los que tienen que ver con la inteligencia artificial.

CUADRO III.5

Consideremos que los lugares en el almacén tienen nombres :
V1, V2, V3, ...

La instrucción "*V1+V2→V3*" significa tomar los números almacenados en *V1* y *V2*, sumarlos y poner la suma en *V3*.

El siguiente programa computa el valor de ar^m-r^n. Antes de empezar, hay que considerar los lugares almacenamiento con los valores iniciales como sigue :

V1 con *a*, *V2* con *r*, *V3* con *m*, *V4* con *n* y *V5*, *V6*, *V7* con la constante 1 ya que serán variables auxiliares para las operaciones que se encuentran en iteraciones, y así poder llegar al resultado deseado que será almacenado en *V8*.

PROGRAMA	EXPLICACIÓN
EMPIEZA V5xV2→V5	Multiplica V5 por r (se tiene que repetir *m* veces).
V3-V7→V3	Resta 1 de *m* (cuenta las iteraciones que deben hacerse para lograr r^m).
SI V3 > 0 REGRESA DOS LINEAS	Condicionante que verifica si se ha multiplicado *m* veces a *r*, almacenando el resultado en V5.
V1xV5→V5	El valor obtenido en V5 (r^m) se multiplica por el coeficiente *a* y este valor es almacenado en V5 (ar^m).
V6xV2→V6	Multiplica V6 por *r* (se tiene que repetir *n* veces).
V4-V7→V4	Resta 1 de *n* (lleva la cuenta de las iteraciones que deben hacerse para lograr r^n).
SI V4 > 0 REGRESA 2 LÍNEAS	Condicionante que verifica si se ha multiplicado *n* veces a *r*, almacenando el resultado en V6.
V5-V6→ V8	Resta lo contenido en V5 (ar^m) menos V6 (r^n), y este resultado se almacena en V8.
IMPRIME V8 TERMINA	Imprime el resultado final.

El programa anterior lo podemos usar con los siguientes valores (ver Cuadro III.6):

$$5(7^2) - 7^1$$

$a=5$
$r=7$
$m=2$
$n=1$

Cargando los valores en los siguientes lugares de almacenamiento :
V1 con *5*, *V2* con *7*, *V3*con *2*, *V4* con *1*, *V5*, *V6*, *V7* con la constante 1 y *V8* será el resultado final.

CUADRO III.6

PROGRAMA	EXPLICACIÓN
EMPIEZA V5xV2→V5 1xV2→V5 1x7→V5 7→V5	Con ésta instrucción se multiplica 7 por sí mismo hasta que elevemos 7 a la segunda potencia. Sustituimos los valores almacenados. El resultado es almacenado en V5.
V3-V7→V3 2-V7→V3 2-1→V3 1→V3	Contador que indica cuantas veces faltan elevar a 7 a la segunda potencia. En este caso V3 nos indica que faltan una iteración.
V3 > 0 REALIZAR OTRA ITERACIÓN CON LAS	Condicionante que verifica si se ha multiplicado 2 veces a 7 , almacenando el resultado en V5.
SIGUIENTES INSTRUCCIONES: V5xV2→V5 , V3-V7→V7, V5xV2→V5 7xV2→V5 7x7→V5 49→V5	
V3-V7→V3 1-V7→V3 1-1→V3 0→V3 YA QUE V3=0 CONTINUAMOS CON LAS SIGUIENTES OPERACIONES.	
V1xV5→V5 5x49→V5 245→V5	El valor obtenido en V5 (7^2) se multiplica por 5 y este valor es almacenado en V5 ($5(7^2)$).
V6xV2→V6 1xV2→V6 1x7→V6 7→V6	Multiplica V6 por 7 (se tiene que repetir 1 vez).
V4-V7→V4 1-V7→V4 1-1→V4 0→V4 V4 = 0 CONTINUAMOS CON LAS SIGUIENTES OPERACIONES.	Resta 1 de 1 (lleva la cuenta de las iteraciones que deben hacerse para lograr 7^1). Condicionante que verifica si se ha multiplicado 1 vez a 7, almacenando el resultado en V6.
V5-V6-> V8 245-V6→V8 245-7→V8 238→V8	Resta lo contenido en V5 ($5(7^2)$) menos V6 (7^1), y este resultado se almacena en V8.
IMPRIME *238* TERMINA	Imprime el resultado final.

Los programas de la Máquina Analítica no se guardan en el almacén sino que están codificados en una secuencia de tarjetas perforadas unidas entre si por cordones. Los cordones permiten que la máquina regrese y repita un grupo de instrucciones una y otra vez, lo que es esencial para llevar a cabo los procesos iterativos, la disposición básica se inspiró en un sistema usado para controlar los diseños de brocada que fabrican los telares de Jacquard y eso a su vez, inspiró una famosa comparación de Lady Lovelace :

"La Máquina Analítica teje los modelos algebraicos tal como el telar de Jacquard teje las flores y las hojas "[2] .

III.4 LA MÁQUINA UNIVERSAL DE TURING.

Alan Turing (1912-1954), fue un matemático inglés como Babbage. Durante la Segunda Guerra Mundial hizo unas aportaciones importantes al desarrollo experimental británico de las computadoras electrónicas. Y aún antes de la guerra, Turing desarrolló la primera teoría matemáticamente elaborada de la computación, incluyendo algunos impresionantes y profundos descubrimientos sobre las aptitudes de las máquinas. Decisivo para esos descubrimientos teóricos fue la invención de un nuevo tipo de computadora que incluía lo que ahora llamamos Máquina de Turing. Existen distintas máquinas de Turing, aunque sólo las podemos considerar dispositivos hipotéticos; lo que importa son los diferentes tipos de teoremas generales que Turing demostró, las Máquinas de Turing son asombrosamente simples. Turing estaba interesado únicamente en cuestiones teóricas abstractas. Por eso sus diseños son muy elegantes para describir y apropiados para probar procedimientos.

Una máquina de Turing es una máquina de estados finitos que consta de dos partes:

1. ***Una cinta***: es sólo un medio de almacenamiento pasivo; a lo largo está dividida en casillas, cada una de las cuales puede contener un elemento a partir de un alfabeto finito especificado con anterioridad. Se toma encuenta que el número de casillas de la cinta es ilimitado, pero que sólo un segmento finito de ellas esté ocupado en cada ocasión; es decir, todas las demás están vacías o contiene únicamente un elemento blanco, especial. La cinta se usa para las entradas y salidas, escribiendo elementos en ellas antes de que empiece la máquina y leyendo los resultados después de que para la máquina.
2. ***Una cabeza***: es la parte activa de la máquina que va recorriendo la cinta hacia la derecha o hacia la izquierda, casilla por casilla y mientras lee o escribe los elementos. La máquina es asociada con la cinta a través de la cabeza. La cabeza tiene tres funciones asociadas en cada ciclo de operación de la máquina de estados finitos:

[2] John Haugeland, "La Inteligencia Artificial", Siglo Veintiuno Editores, México 1988, pag. 126.

a) *Leer* la casilla de la cinta que le corresponde, que es la única que puede leer hasta que pasa a otra.
b) *Escribir* sobre la casilla un símbolo permitido en la cinta que por las condiciones de la cabeza corresponde.
c) *Mover la cabeza* hacia la izquierda o hacia la derecha un cuadro adyacente.

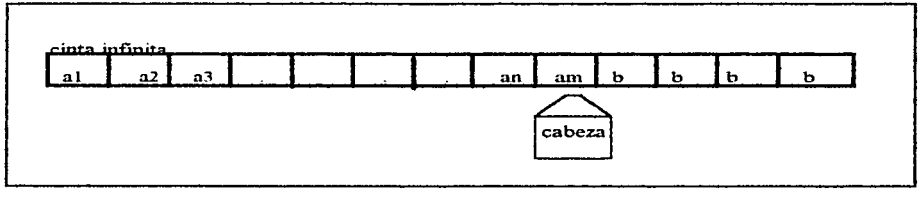


figura 3.2 Esquema gráfico de la Máquina de Turing.

También en cada paso, la propia cabeza se encuentra en un estado interno particular. Este estado cambia de un paso al siguiente; pero con ciertas condiciones la cabeza entrará en un estado final llamado "*parar*", en cuyo caso la máquina se detiene dejando sus salidas en la cinta.

Un movimiento de la Máquina de Turing depende del símbolo leído por la cabeza y el estado de la cabeza.

Formalmente una Máquina de Turing es denotada por:

$$\text{“}M=(Q,\Sigma,\Gamma,\delta,\_,q_0,F)$$

donde

Q: Conjunto de estados finito: estados en los que la cabeza se encontrará según el símbolo leído.
Σ: Es un subconjunto de Γ no incluyendo el blanco y es el conjunto de símbolos de entrada.
Γ: Conjunto finito de símbolos permitidos en la cinta.
δ: Es la función del siguiente movimiento mapeando QxΓ={(estado, símbolo permitido)} lleva a Γx{ D,I }xQ= {(símbolo permitido, un movimiento hacia la derecha o izquierda, estado)}. Delta puede estar sin embargo definida por algunos argumentos.
_: Es un símbolo de Γ y es un blanco.
q_0: Es el estado inicial.

F: Conjunto de estados finales."[3]

Así todo el funcionamiento de una máquina de Turing se puede especificar en una sola tarjeta bidimensional δ, con un renglón para cada elemento, que la cabeza debe encontrar, y una columna para cada estado en que puede estar la cabeza, como podemos ver a continuación en el ejemplo del cuadro III.7.

CUADRO III.7

Máquina de Turing sencilla

En el diagrama siguiente, los estados están enumerados a lo ancho de la parte superior, el alfabeto está listado hacia abajo, a la izquierda. Correlativamente con cada estado y elemento está también en el diagrama una entrada de tres caracteres o la palabra ***Parar***. El primer caracter es el elemento que hay que escribir; el segundo es la I o una D, para moverse a la izquierda o a la derecha y el tercero es el número de estado siguiente.

$$M=(Q,\Sigma,\Gamma,\delta,\_,q_0,F)$$

donde

$Q=\{ q_0, q_1 \}$
$\Gamma=\{ A,B,C,\_ \}$
$\Sigma=\{ A,B,C \}$
$F=\{ q_1 \}$

δ	q_0	q_1
A	AIq_0	BDq_1
B	BIq_0	ADq_1
C	CIq_0	CDq_1
_	$\_Dq_1$	***Parar***

δ contiene tres tipos en su alfabeto además del blanco que se indica con la letra del subrayado.

Damos por sentado que empieza el estado q_0 y que va explorando alguna casilla en el centro de la cadena de As, Bs, Cs. Primero se mueve hacia la izquierda, sin cambiar nada en la cinta. Pero cuando llega al extremo izquierdo de la cadena, o sea el primer blanco, pasa al estado q_1 y empieza moverse a la derecha, cambiando todas las As por Bs y las Bs por As sin tocar las Cs. En el extremo derecho se para.

Podemos transformar esta matriz δ a las producciones que veremos a continuación en el cuadro III.8.

[3] John E. Hopcroft, "Introducción a la teoría de los autómatas, lenguajes y computación", México,Compañía Editorial Continental S.A. de C.V., 1993.

CUADRO III.8

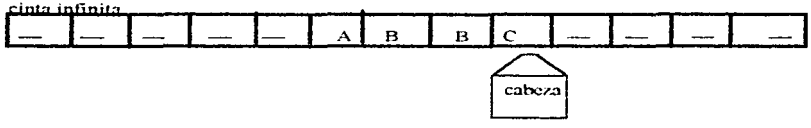
$q_0A \rightarrow Aq_0$
$q_0B \rightarrow Bq_0$
$q_0C \rightarrow Cq_0$
$q_0\_ \rightarrow \_Dq_1$
$q_1A \rightarrow BDq_1$
$q_1B \rightarrow ADq_1$
$q_1C \rightarrow CDq_1$
$q_1\_ \rightarrow$ ***Parar***

De ésta manera podemos comprender más fácilmente el diagrama δ.

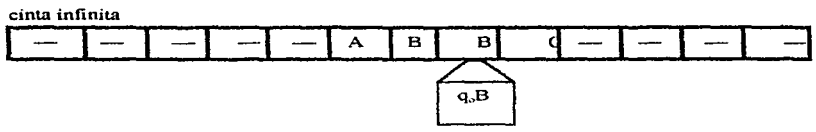
A continuación gráficamente en la figura 3.3 veremos cuál es el proceso de la Máquina de Turing con la cinta que tiene la cadena _ABBC_ , al seguir la sustitución ya sea de la matriz δ del cuadro III.7 o de las producciones del cuadro III.8.

figura 3.3. Ejemplo gráfico de un proceso de la Máquina de Turing.

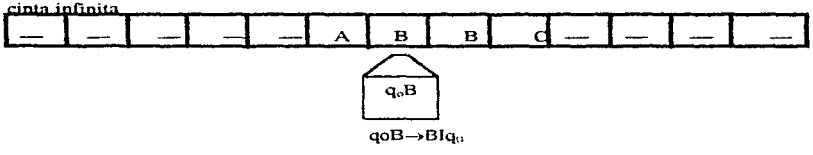
El segmento de casillas ocupado en esta ocasión lo vemos en el dibujo siguiente y sobre el cual se harán los cambios que indica la tarjeta bidimensional δ , pero nuestro propósito, es el de presentar la utilidad de las producciones que en conjunto nos llevan a una red semántica, el proceso lo ejemplificaremos retomando las producciones del cuadro III.8.

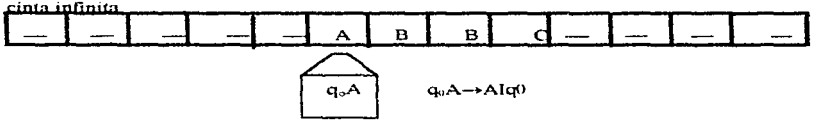


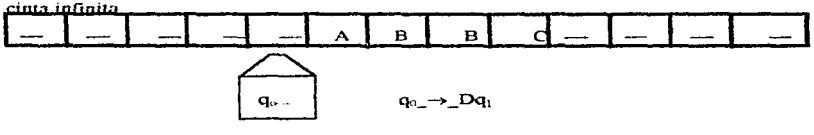
El estado inicial es 'q_0' como se indica en el cuadro III.7, y de acuerdo con el elemento de entrada que la cabeza está leyendo, nos corresponde la producción '$q_0C \rightarrow Cq_0$', el lado derecho nos dice que deje escrito el elemento 'C', luego haga un movimiento hacia la izquierda 'I' y se quede en el estado 'q_0'.



La casilla adjunta tiene el elemento 'B', por lo cual la producción correspondiente es '$q_oB \rightarrow BIq_0$'. Como podemos analizar, la Máquina de Turing está programada para moverse a la izquierda 'I' sin cambiar símbolos hasta que llegue a un blanco '_', esto lo podemos ver en las siguientes representaciones.

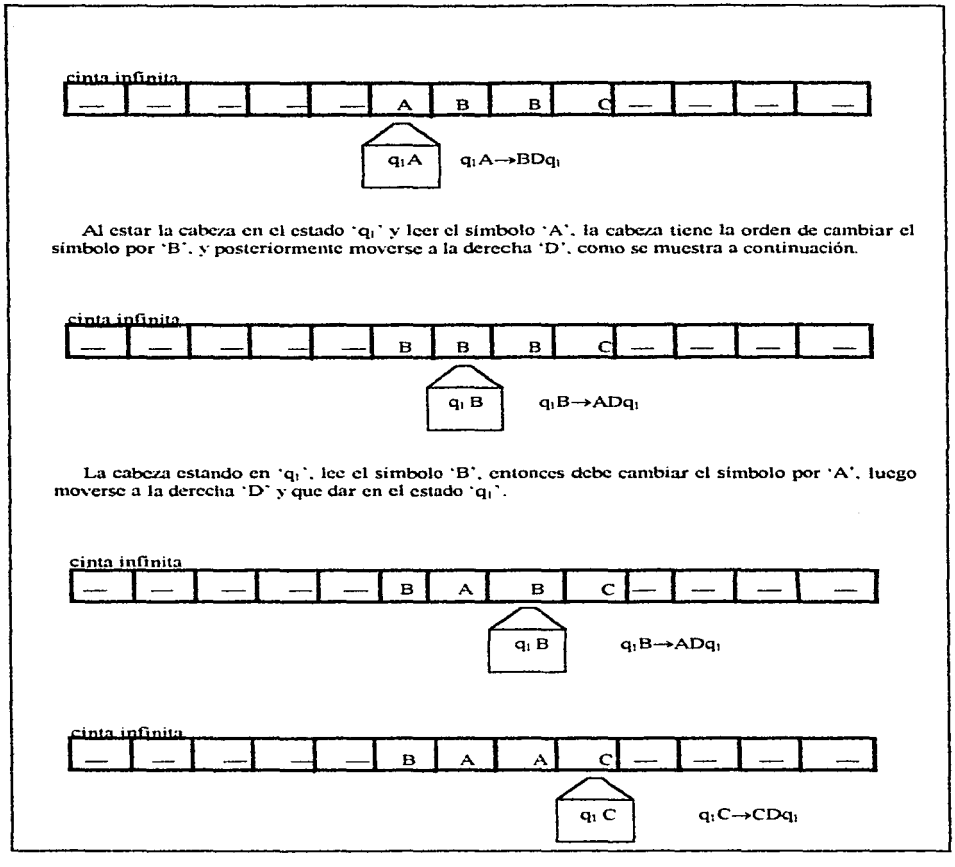






Una vez que llega la cabeza al extremo izquierdo en donde se encuentra el primer blanco '_', entonces cambia el estado de la cabeza a q_1, dejando el blanco y moviéndose a la derecha 'D'.

figura 3.3. Continuación del ejemplo gráfico de un proceso de la Máquina de Turing.

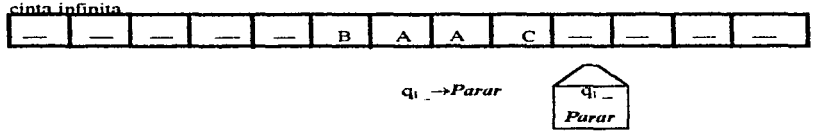


Al estar la cabeza en el estado 'q_1' y leer el símbolo 'A', la cabeza tiene la orden de cambiar el símbolo por 'B', y posteriormente moverse a la derecha 'D', como se muestra a continuación.

La cabeza estando en 'q_1', lee el símbolo 'B', entonces debe cambiar el símbolo por 'A', luego moverse a la derecha 'D' y que dar en el estado 'q_1'.

figura 3.3. Continuación del ejemplo gráfico de un proceso de la Máquina de Turing.

La Máquina de Turing tiene la orden de cambiar las A's por B's y las B's por A's dejando las C's igual.



Cuando llega la cabeza al extremo derecho encontrando el primer blanco '*Para*' terminando el proceso y dejando la salida impresa en la cinta.

figura 3.3. Continuación del ejemplo gráfico de un proceso de la Máquina de Turing.

La esencia de la idea es codificar una descripción de la máquina que se va a imitar y usar eso como entrada del imitador. Este leerá la descripción, hará mucha computación y de vez en cuando hará un movimiento exactamente equivalente al movimiento siguiente que hubiera hecho la primera.

El ejemplo muestra como realmente el propósito de Turing se cumple, ya que su máquina puede hacer cualquier operación (dentro de sus limitaciones), con sólo proporcionar las funciones adecuadas de instrucciones (tabla δ). Desde ese momento presentó la idea de que una máquina tuviese instrucciones e información almacenada.

III.5 LOS SISTEMAS DE PRODUCCIÓN DE NEWELL

Allen Newell (1927-) en 1955 inventó los manipuladores generales de símbolos, la búsqueda heurística y la inteligencia artificial. Sin embargo hasta 1960, las ideas de Newell se inclinaron por los sistemas de producción .

Como la Máquina de Universal de Turing, todas las producciones actúan en una memoria lineal común llamada espacio de trabajo. Pero difieren en el modo de determinar quién trabaja, dónde y cuando. La mayor parte de las acciones modifican los modelos del espacio de trabajo que pueden satisfacer a otros especialistas.

Los espacios de trabajo no están especificados ni de manera absoluta ni relativa; si no que están especificados más bien por sus contenidos, es decir en términos de lo que en ese momento está almacenado .

Para Newell las producciones mismas son el programa. La función del programador es juntar una serie de producciones cada una capaz de ejecutar una determinada acción en determinadas circunstancias, cuyo comportamiento colectivo será el *sistema* deseado.

Cualquier producción tiene dos partes básicas: una condición y una acción: La definición de una producción es esencialmente una regla que dice:

TAN PRONTO COMO < *la condición*> ESTÁ SATISFECHA, EJECUTA <*la acción* >

más esquemáticamente:

<***condición*** > → <***acción***>

La condición es un modelo que puede aparecer en el espacio de trabajo, y la acción es generalmente una modificación de ese modelo. Por ejemplo un sistema que simplifique las ecuaciones algebraicas podría contener una regla como ésta:

" A+ B = C + B " → se reemplaza con " A= C "

Las comillas indican lo que hay en el espacio de trabajo mismo, en tanto que las mayúsculas (A, B, C) son consideradas como las que reservan el lugar para cadenas arbitrarias de elementos del espacio de trabajo reales. Por ejemplo la ecuación:

$$2x+4y(z-14)= w-3+4y(z-14)$$

Tiene considerados los lugares de almacenamiento para sus elementos como sigue:

$$A=2x,\ B=4y(z-14),\ C=w-3$$

Como podemos ver, para simplificar la ecuación utilizamos la regla de producción anterior, reemplazando la ***condición*** "2x+4y(z-14)= w-3+4y(z-14) " por la acción que es " A= C ", quedando la ecuación reducida a:

$$2x = w-3$$

Este ejemplo presupone aptitudes realmente excelentes de reconocimiento, comparación, y adaptación para adecuar la expresión compleja 4y(z-14) con B en ambos lados de la ecuación.

Los sistemas de producción fomentan un grado de modularidad; un módulo es un subsistema independiente que lleva a cabo una tarea bien definida e interactúa con el resto del sistema sólo de determinados y bien definidos modos.

Podemos ver la preocupación de Newell por que las condiciones de las producciones siempre sean correctas, en el cuadro III.9, de ésta manera él nos da un consejo que nos puede servir para que cuando nosotros programemos, siempre lo llevemos en mente.

CUADRO III.9

La repetición en los sistemas de producción

La arquitectura de produccción, es universal: para cualquier algoritmo simbólico, existe un sistema de producción de una manera natural; lo difícil es detenerlo o evitarlo. Supongamos, por ejemplo, un sistema con una solo producción:

cualquier entero, N -> incrementa : N

Si este sistema encuentra siempre un entero por el cual empezar, se sentará y seguirá incrementando. Pero no hay nada que ordene al sistema repetir; este simplemente lo hace. Por otra parte, si lo queremos detener entonces se lo tenemos que decir, añadiendo otra producción con mayor prioridad:

cualquier entero mayor que 99 -> parar

Una producción se activa así misma cuando y donde las condiciones estén correctas.

III.6 LAS GRAMÁTICAS LIBRES DE CONTEXTO ORIENTADOS A LOS LENGUAJES DE PROGRAMACIÓN.

Los lenguajes libres de contexto, son de gran importancia práctica sobre todo en la definición de lenguajes de programación, en la formalización del concepto de análisis de la gramática, simplificación en los lenguajes de programación y en otras aplicaciones de procesamiento de cadenas. Por ejemplo las gramáticas libres de contexto son útiles para describir expresiones aritméticas que tengan una anidación de paréntesis balanceados y estructuras de bloque en los lenguajes de programación (begin y end).

CADENA, ALFABETO Y LENGUAJE

Comenzaremos definiendo los conceptos que se manejarán durante este tema como es el caso de *cadena* la cual es una secuencia finita de símbolos, por lo tanto *abcb* es una cadena. La longitud de la cadena *w*, se denota como */w/*, que es el número se símbolos que componen a la cadena. Así la cadena *abcb* tiene longitud 4. La cadena vacía denotada por $\in$ es la cadena que consiste en cero símbolos y por lo tanto su longitud es de cero.

Un *alfabeto* es un conjunto finito de símbolos permitidos en una gramática (símbolos terminales). En tanto que un lenguaje es un conjunto de cadenas de símbolos tomados del alfabeto. El conjunto vacio $\varnothing$ y el conjunto formado por la cadena vacía $\{\in\}$ son lenguajes.

INICIOS DE LAS GRAMÁTICAS LIBRES DE CONTEXTO

El desarrollo de las gramáticas libres de contexto se debió a la necesidad de escribir los lenguajes naturales. Podemos escribir reglas como las siguientes:

<oración> → <frase sustantiva> <frase verbal>
<frase sustantiva> → <adjetivo> <frase sustantiva>
<frase sustantiva> → <sustantivo>
<sustantivo> → niño , <sustantivo> → roca
<adjetivo> → pequeño , <adjetivo> → corre

En donde los símbolos no terminales se escriben entre paréntesis en ángulo y los terminales se denotan con palabras sin paréntesis.

Por ciertas razones, las gramáticas libres de contexto en general no se consideran adecuadas para la descripción de los lenguajes naturales como el español. Por ejemplo si extendemos las producciones anteriores a todo el idioma podríamos ser capaces de tomar "*la roca*" como frase sustantiva y "*corre*" como frase verbal. Por consiguiente la oración sería " *la roca corre*" lo que carece de significado, es claro que se necesita de alguna información semántica para reglamentar cadenas sin significado que son sintácticamente

correctas. Sin embargo las gramáticas libres de contexto juegan un papel importante en la lingüística computacional.

DEFINICIÓN DE LENGUAJES DE PROGRAMACIÓN

Mientras que los lingüistas estudiaban las gramáticas libres de contexto los científicos de la computación comenzaron los lenguajes de programación mediante una notación conocida como Forma de Backus-Naur (BNF), que es la notación para las gramáticas libres de contexto.
Estas gramáticas ofrecen ventajas significativas para los diseñadores de lenguajes y los escritores de compiladores:
*Una gramática da una especificación sintáctica precisa y fácil de entender en un lenguaje de programación.
*Una gramática diseñada adecuadamente imparte una estructura a un lenguaje de programación útil para la traducción de programas fuente a código objeto correcto y la detección de errores.

Una gramática describe de forma natural la estructura jerárquica en muchas construcciones de los lenguajes de programación. Por ejemplo : Una proposición if else tiene la forma en C :

if (expresión) proposición else proposición

Esto es, la proposición es la concatenación de la palabra clave if un paréntesis que abre, una expresión, un paréntesis que cierra, una proposición, la palabra else y otra proposición. Empleando la variable **expr** para denotar la expresión, la variable ***prop***, para una proposición, esta regla de estructuración se expresa:

<Prop> → if <expr> <prop> else <prop>

donde es posible leer la flecha como "puede tener la forma". Dicha regla se denomina producción como hemos visto anteriormente. En esta producción los símbolos no terminales son la palabra clave **if** y los paréntesis y los símbolos terminales a las variables **expr** y **prop** ya que pueden ser sustituidos por otras producciones. Una gramática independiente del contexto tiene cuatro componentes :

1. Conjunto de símbolos *terminales*, los cuales ya no se pueden sustituir por otras producciones.
2. Conjuntos de *no terminales*, los cuales pueden sustituirse por otras producciones.
3. Un conjunto de *producciones*, en el que cada producción consta de un no terminal, llamado lado izquierdo de la producción, una flecha y una secuencia de símbolos terminales, no terminales, o ambos, llamado lado derecho de la producción.
4. La denominación de uno de los no terminales como *símbolo inicial*.

De una *gramática* se derivan cadenas empezando con el símbolo inicial y reemplazando repetidamente un no terminal por el lado derecho de una producción para ese no terminal.

Las cadenas de símbolos terminales derivadas del símbolo inicial forman el lenguaje que define la gramática. La cadena que contiene cero componentes terminales es cada cadena vacía.

Se especifica la gramática dando:
a) El nombre de la gramática, pudiendo ser (G_i),
b) el alfabeto (A), teniendo en cuenta que sólo pertenecen a este conjunto los símbolos terminales,
c) el símbolo inicial (I),
d) los símbolos no terminales (NT), incluyendo al símbolo inicial, y
e) el conjunto de las producciones (P), que pertenecen a la gramática (G_i).

Para identificar los símbolos terminales se escriben entre paréntesis angulares '< >' y los símbolos no terminales con letras normales y sin los paréntesis en ángulo.

Por ejemplo: Especificamos la siguiente gramática G_1

G_1

A={ +, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
I={ lista }
NT={ lista, dígito }
P={ <lista> → <lista> + <dígito>
<lista> → <lista> - <dígito>
<lista> → <dígito>
<dígito> →0
<dígito> →1
<dígito> →2
<dígito> →3
<dígito> →4
<dígito> →5
<dígito> →6
<dígito> →7
<dígito> →8
<dígito>→9 }

Podemos tener varias producciones con el mismo no terminal del lado izquierdo, por lo cual recurrimos a una cómoda notación, en donde las producciones con el mismo no terminal del lado izquierdo pueden tener sus lados derechos agrupados, separados por el símbolo |, que se leerá "ó", quedando las producciones anteriores de la siguiente forma:

G_1

A={ +, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }, I={ lista }, NT={ lista, dígito },

P={ <lista> → <lista> + <dígito> | <lista> - <dígito> | <dígito>

<dígito> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }

Ejemplo: Utilizando la gramática G_1 se puede deducir que 9-5+2 es una cadena derivada del símbolo inicial lista y la cual podemos obtener utilizando las siguientes producciones:

<lista> → <lista> + <digito>
→ <lista> + 2
→ <lista> - <digito> + 2
→ <lista> - 5 + 2
→ <digito> - 5 + 2
→ 9 - 5 + 2

El lenguaje definido por la gramática del ejemplo anterior está formado por listas de dígitos separados por los signos de más y menos. Este razonamiento se ilustra con el árbol de la figura 4.1. Cada nodo en el árbol está etiquetado con un símbolo de la gramática, un nodo interior y sus hijos corresponden a una producción; el nodo interior corresponde al lado izquierdo de la producción, y los hijos al lado derecho.

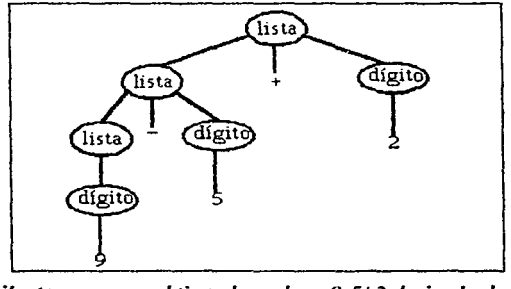


figura 3.4. Árbol que ilustra como se obtiene la cadena 9-5+2 derivada de la gramática G_1.

Estos árboles se conocen con el nombre de árboles de análisis sintáctico.

CAPITULO IV
REDES SEMÁNTICAS EN LA COMPUTACIÓN

IV.1 INTRODUCCIÓN

Ya sea que se planee usar computadora en el trabajo, hogar o escuela, es sorprendente saber cuántos trabajos se pueden automatizar o simplificar utilizando una computadora. Si se escriben textos, si se trabaja con números, si se dibujan ilustraciones, si se toca música o se juega, para casi todo se puede comprar un programa para la computadora que permita ejecutar estas tareas. También podemos crear nuestros programas, pero lo más importante es en qué tipo de lenguaje de programación hacerlo.

Es interesante mostrar en qué parte de la computación podemos emplear a las redes semánticas y es precisamente en los lenguajes de programación orientados a objetos y en el software comprado como Windows, en donde se pueden utilizar para representar su funcionamiento interno. Esto se debe a que las redes semánticas pueden representar relaciones hereditarias.

Para explicar el funcionamiento de los lenguajes orientados a objetos y su representación por medio de las redes semánticas, en este capítulo hablaremos sobre las características o diferencias entre los lenguajes de programación convencional y los orientados a objetos, ya que durante muchos años se utilizaron los lenguajes de programación convencional y esto dificulta el transformar la forma de programar en un lenguaje orientado a objetos.

Por otro lado se explica la forma en que trabaja la técnica orientada a objetos como una introducción a la red semántica de la presentación del menú de Windows'95, para explicar como es su trabajo interno con las herencias.

IV.2 RELACIONES HEREDITARIAS.

Hay que recordar que las redes semánticas son notaciones gráficas de nodos que representan objetos, acciones o eventos y los arcos muestran las relaciones entre nodos, como ya se vio en el Capítulo II.

En las redes semánticas algunas relaciones son hereditarias, las cuales son especialmente importantes en la Inteligencia Artificial dado el enorme tamaño y complejidad de casi todos los programas. En una relación hereditaria, algunos objetos heredan información o atributos de otros. Por ejemplo, en la red semántica de la figura 4.1, Mesa 1 y Mesa 2 son ejemplos de Mesa, cualquier cosa cierta para una Mesa resulta también verdadera para Mesa 1 y Mesa 2. De esta manera no se necesita reescribir los atributos, ya sea su aplicación a la escritura, tanto para Mesa 1 como para Mesa 2. Sin embargo, se puede añadir nueva información a los datos que describen a la Mesa 1 y a la Mesa 2 ya que naturalmente cada una de ellas, tiene propiedades características. Por ejemplo, la Mesa 1 está hecha de madera, mientras que la Mesa 2 está hecha de acero. Por lo tanto las jerarquías indican las diferencias entre objetos a niveles diferentes.

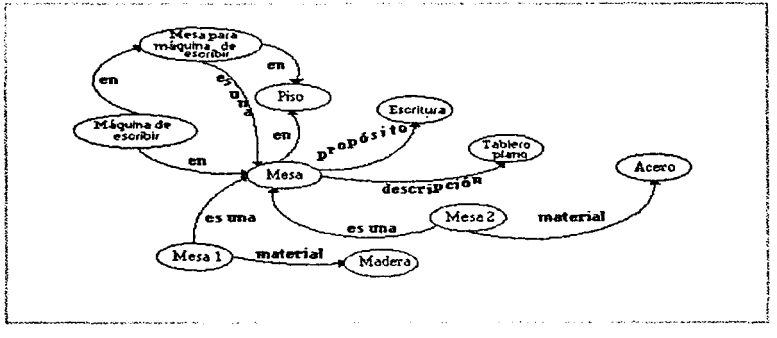


Fig.4.1.Relaciones hederitarias en una red semántica.

Una ventaja de las jerarquías de herencia en los sistemas de Inteligencia Artificial es la capacidad que la jerarquía da a los nuevos nodos u objetos del sistema para intuitivamente obtener información y significado de sus atributos, capacidades y limitaciones tan pronto como se crean. Tales mecanismos están ligados a las jerarquías de las redes semánticas.

Los mecanismos de herencia se encuentran en un grupo de lenguajes de programación llamados lenguajes orientados a objetos. Los dos principales lenguajes de programación orientados a objetos son: Smalltalk y Flavors, entre otros. Smalltalk fue desarrollado principalmente para explorar y facilitar, más que aplicaciones de la Inteligencia Artificial, gráficos de alta calidad desarrollados interactivamente (hombre-máquina). Flavors ha sido implantado como un modelo de la Inteligencia Artificial por encima de LISP, ya que aún cuando LISP es el principal lenguaje de la Inteligencia Artificial, no tiene incorporado ningún mecanismo de herencia, pero los programadores de la Inteligencia Artificial suelen ampliar LISP incorporándole tal característica.

IV.3 CARACTERÍSTICAS O DIFERENCIAS ENTRE LENGUAJES DE PROGRAMACIÓN CONVENCIONALES Y LOS ORIENTADOS A OBJETOS.

Los lenguajes de programación orientados al objeto indican un método diferente de manejar programas y datos en comparación con otros lenguajes de programación.

En cualquier lenguaje convencional, los programas tienen dos componentes: una parte de procedimiento que opera sobre los datos, y los datos sobre los que los componentes operan. El componente de procedimiento que se conoce como programa contiene las instrucciones necesarias para efectuar una operación. Tales componentes incluyen procedimientos para <<sumar>>, <<encontrar>>,<<leer>>, <<escribir>>, <<imprimir>> y <<declaraciones condicionales de programación>> (así como <<si estas condiciones son verdad, entonces hacer eso>>), entre otras. Los datos sobre los que opera el programa usualmente son números, cadenas de caracteres, formas gráficas, documentos o archivos.

En la programación convencional, casi todos consideran los programas como elementos primarios y los datos (u objetos) como elementos secundarios. En la programación orientada a objetos es justamente al revés. Los datos, tales como una forma gráfica o un documento, sobre los que hay que operar, son de importancia primaria.

En la programación orientada a objetos, en vez de enviar en el programa el nombre del objeto que se va a manipular (que corresponde en la programación convencional a proporcionar datos al programa), el programador envía al objeto (o datos) un mensaje.

Un mensaje es en realidad el nombre de una operación o procedimiento (lo que se denomina un método en la programación orientada a objetos) a ejecutar. Las operaciones típicas sobre el objeto <<documento>> incluyen abrir y cerrar el documento, encontrar, imprimir y mostrar. Enviar al objeto <<documento >> uno de estos mensajes es similar a invocar una subrutina en el programa convencional.

El objeto es capaz de responder apropiadamente al mensaje porque un objeto se define junto con el grupo de procedimientos asociados que puede efectuar. Como consecuencia, cuando el objeto, por ejemplo <<documento>> , recibe un mensaje tal como mostrar, sabe como efectuar ese procedimiento porque junto con el objeto están las instrucciones de procedimiento. Para efectuar la misma tarea en programación convencional, el programa llamaría un procedimiento llamado <<mostrar>> que trataría al documento como datos, ejecutaría las instrucciones de representación visual y mostraría el documento.

El estilo orientado al objeto adquiere importancia cuando una operación se puede efectuar de diferente forma, dependiendo del objeto al que se aplica. Bajo estas condiciones, es muy práctico para cada objeto saber cómo efectuar sus propias operaciones. Como veremos, este conocimiento evita errores de programación y permite que las decisiones programáticas se tomen durante la ejecución del programa. También ahorra tiempo porque en la programación orientada al objeto este conocimiento procedural es heredado.

IV. 4 HERENCIA ORIENTADA AL OBJETO.

En lenguajes de programación orientados a objetos, como Smalltalk, Flavors, C++ entre otros la herencia es posible porque estos lenguajes organizan los programas en estructuras jerárquicas denominadas clases y subclases (ver la figura 4.2). Las clases y subclases son conjuntos de una misma clase de objetos genéricos o componentes. Normalmente, los programas orientados al objeto pueden contener clases como números, cadenas de caracteres, formas gráficas, documentos o archivos. En algunos programas como Windows las clases son ventanas, menús e iconos.

En programación orientada a objetos, los objetos más genéricos en una red de clases están más altos en la jerarquía. Los objetos más específicos que se describen en una clase son los inferiores de una jerarquía. Las relaciones entre los objetos de una jerarquía y los situados inmediatamente debajo de ellos se describen mediante una relación heredada padre-hijo, porque algunos objetos de la jerarquía se denominan padres y los objetos con ellos relacionados en el nivel inferior siguiente de la jerarquía son hijos.

Asociados a cada clase hay un grupo de procedimientos (métodos). Cada subclase de una clase hereda automáticamente estos procedimientos de sus padres, a menos que tenga sus propios procedimientos que cubran uno o más de los procedimientos heredados.

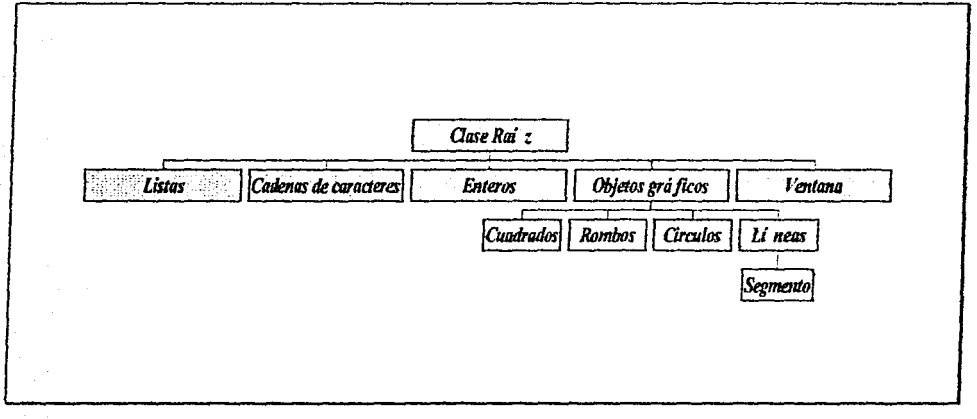


Fig. 4.2. Jerarquías de clases y subclases en programación orientada a objetos

La clase más elevada de una jerarquía es una sola clase raiz (ver figura 4.3, pag. 40). Una determinada clase raíz -por ejemplo, de un programa gráfico- puede, en el siguiente nivel inferior tener clases de listas, cadenas, enteros, objetos gráficos y ventanas. La clase de objetos gráficos puede tener subclases de cuadrados, rombos, círculos y líneas.

Los métodos asociados con la subclase <<línea>> puede incluir *cortar* (que reduce una línea), *nueva* (que crea una línea), *destruir* (que destruye una línea) y *longitud* (que computa la longitud de la línea). Para solicitar a un programa que calcule la *longitud* de una línea, el usuario o programador envía el mensaje de longitud a la línea. Para determinar la longitud, una rutina de mensaje del programa orientado a objetos busca en una tabla de métodos asociada con la subclase <<línea>> y encuentra el nombre <<línea>> del método (o rutina). A partir de aqui, el programa sigue un puntero (la dirección de las instrucciones de la rutina en la memoria de la computadora) de la tabla a la rutina que calcula la longitud de la línea y seguidamente ejecuta la rutina. Se siguen técnicas similares para los mensajes *crear, destruir*, y *cortar*.

Si se añade otra subclase <<segmento>> a <<línea>>, hay que escribir una rutina de *estilo* (esta rutina da como opcion marcar al segmento dirección). Ahora, cuando un usuario envie el mensaje *estilo* a <<segmento>>>, el nuevo mensaje inicia una llamada a la rutina de mensaje que encuentra y ejecuta la rutina de *estilo*.

Por otro lado, si el usuario envía un mensaje a <<segmento>> para ejecutar el procedimiento *nueva* o *destruir*, la rutina de mensaje no encontrará tales procedimientos en la tabla de métodos para la subclase de <<segmento>>, porque estos procedimientos nunca se escribieron allí. Pero no importa, la subclase de línea ejecutará estos procedimientos en cualquier clase porque están contenidos en la tabla de métodos escrita para una clase padre más alta en la jerarquía. No hay que escribir ningún procedimiento porque se trata de un programa orientado al objeto y si la rutina de mensaje no se encuentra los procedimientos requeridos en la tabla de métodos determinada, buscará en la tabla de métodos de la clase en el nivel inmediatamente superior, la clase padre. Si el método se lista allí, se ejecuta inmediatamente.

Lo que ocurre realmente es que la clase <<segmento>> ha heredado de sus padres más altos en la jerarquía, las rutinas prescritas *nueva* y *destruir*. Los procedimientos se pueden heredar también de las clases abuelas o antepasadas. En la herencia ancestral, si un mensaje denomina determinado método que no está listado inmediatamente superior en cuestión, la rutina de mensaje continuará buscando en el árbol de herencia hasta encontrar la clase que contiene ese método, y una vez que lo encuentra, lo ejecuta.

Una ventaja de esta jerarquía de herencia en la programación orientada al objeto es que el programador no tiene que cambiar o añadir demasiados códigos de programación cuando se añade una nueva clase. El programador sólo necesita escribir los códigos de funciones especiales o adaptadas y características que diferenciarán a los hijos. Estos heredan de los padres los atributos restantes. Puesto que sólo hay que escribir el código que diferencia a los hijos como en las redes semánticas, las relaciones jerárquicas indican la diferencia entre objetos en niveles diferentes.

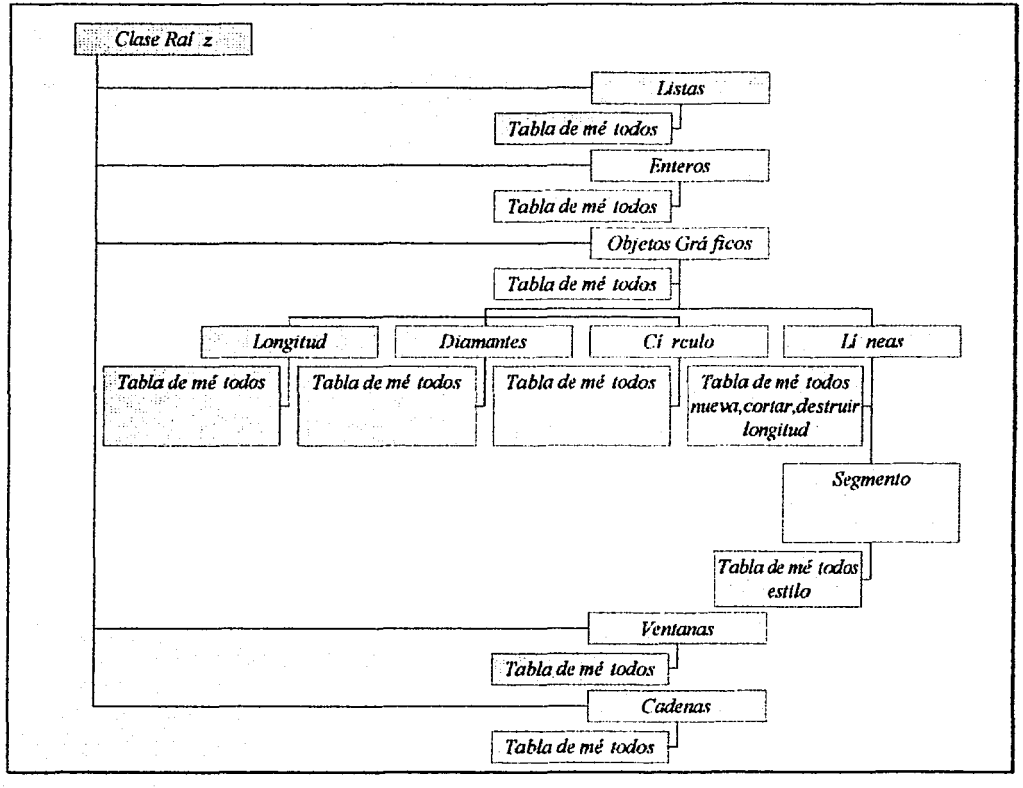


Fig. 4.3. Herencia de métodos en la programación orientada a objetos

IV.5 EJEMPLO DE LA FORMA EN QUE TRABAJA LA TÉCNICA ORIENTADA AL OBJETO.

La técnica orientada al objeto trabaja como describiremos a continuación:

<<Imprimir>> es una operación que se puede aplicar a muchos objetos del sistema. Pero hay muchos modos de implementar <<Imprimir>>, dependiendo del dispositivo asociado con el procedimiento <<Imprimir>>. Por ejemplo, el procedimiento para imprimir un archivo en una pantalla gráfica de una terminal ASCII es diferente a la de una impresora de matriz de puntos.

Para el usuario que desea un dispositivo concreto para imprimir algo, no le importa a qué objeto (o dispositivo) se le aplicará <<Imprimir>>, porque en la programación orientada a objetos, <<Imprimir>> es un procedimiento genérico. Cómo se aplica <<Imprimir>> a un objeto concreto queda determinado por el objeto al que se va a aplicar <<Imprimir>>. Los objetos determinan el significado de <<Imprimir>> porque encapsulado dentro de cada objeto al que se le puede aplicar (tal como el archivo, la pantalla gráfica, la terminal ASCII y la impresora de matriz de punto) <<Imprimir >> existe un procedimiento especial para cada uno de los objetos.

Consecuentemente, durante el desarrollo de un programa, el programador que desea imprimir algo en una terminal ASCII, sólo necesita enviar un mensaje de procedimiento genérico <<Imprimir>> a un objeto terminal ASCII. Cómo imprimir, y qué procedimiento invocar para imprimir en la terminal ASCII, no es asunto del programador. Es un asunto del terminal ASCII, como otros asuntos del sistema, sabe como imprimir y conoce su propia representación interna. Por tanto, tiene toda la información necesaria para aplicarse a sí mismo la <<Impresión>>.

Si después de alguna experimentación, el programador decide que sería preferible imprimir en una pantalla gráfica en vez de la terminal ASCII. No hay que ir al programa y experimentar con procedimientos y equivocaciones como ocurre en la programación convencional. Esta facilidad de cambio permite a los programadores al máximo, dejando que los objetos efectúen sus propias operaciones.

Ahora supongamos que el programador deba añadir un nuevo dispositivo de salida, tal como un sintetizador vocal. Sin lugar a duda, <<Imprimir>> (print) aplicado a un sintetizador vocal difiere de <<Imprimir>> aplicado a los dispositivos que hemos denominado anteriormente. En la programación convencional, el programador acude al procedimiento print del programa cuando el puede encontrar una declaración de caso. Una declaración caso tiene la forma: " Si el caso es imprimir y se aplica a un archivo, entonces imprimir así, pero si el caso es imprimir y se aplica a una terminal ASCII, entonces imprimir así", para una amplia lista de casos similares. Para añadir instrucciones de impresión en el caso del sintetizador vocal, el programador puede ampliar

aún más la declaración para el caso del sintetizador vocal. Pero si al hacerlo se comete una equivocación, el sistema puede fallar y no imprimir nada porque hay un error en <<Imprimir>>.

En contraposición, en la programación orientada a objetos, el programador define tanto el nuevo objeto que representa el sintetizador vocal como la rutina asociada que instruye al sintetizador vocal sobre el procedimiento <<Imprimir>> . Como en el caso de la pantalla gráfica, para aplicar <<Imprimir>> al sintetizador vocal, el programador envía el mensaje <<Imprimir>> el cual sabe como ejecutar el procedimiento. A diferencia de los programas convencionales, si el procedimiento <<Imprimir>> tiene un error, los restantes objetos del sistema siguen imprimiendo por que la rutina <<Imprimir>> está aislada de las restantes y, además, encapsulada con los objetos a ella asociados.

La programación orientada al objeto es ventajosa por que su organización y estructura permite asociar los dispositivos con relevantes procedimientos en vez de que cada operación tenga que saber los procedimientos de cada dispositivo.

IV.6 RED SEMÁNTICA DE ALGUNOS ASPECTOS DEL MENÚ DE WINDOWS.

Como podemos ver Windows es un software que nos proporciona grandes ventajas:

* Toda la información se guarda en un disco, en donde se puede encontrar, cambiar y almacenar los datos. No es necesario reingresar información, ni teclear la misma información dos veces.
* La computadora ejecuta miles de cálculos en forma automática a la velocidad de la luz, evitando al usuario el trabajo de hacerlo y reduciendo el riesgo de error. Todo lo que se necesita es seleccionarse el programa correcto dentro de Windows para obtener los resultados requeridos, ingresar los datos, y seleccionar los comandos para poner al programa a trabajar.
* Se reduce en forma significativa el margen de error. Por ejemplo las hojas de cálculo siguiendo las pistas de las ecuaciones, las funciones y las fórmulas; algunos programas de procesamiento de palabras verifican la ortografía y la gramática.
* Se puede aligerar las cargas de trabajo liberando a los empleados de tareas repetitivas e incrementando su productividad. Por ejemplo no se necesita reteclear un nombre y una

dirección en cartas, rótulos, formas y sobres; la información se ingresa sólo una vez y la computadora la imprime tantas veces sea necesario.

*Windows proporcionan la capacidad de aumentar la calidad del trabajo de la siguiente manera:

- La tipografía computarizada permite producir documentos profesionales. Los nuevos programas de distribución de la página permiten mezclar texto y gráficas, organizar el texto en formatos de varias columnas y agregar efectos especiales como la rotación del texto y las sombras tridimensionales, además imprimen páginas que rivalizan en cantidad con el texto tipográfico, y pueden recortar muchos pasos y muchos de los costos del proceso editorial.
- Con la multimedia logramos la mejor presentación, mezclando la más alta calidad de sonido, gráficas, texto y video disponible para crear presentaciones brillantes, herramientas de aprendizaje o medios interactivos.

A continuación se presenta el software diverso que puede correr bajo la plataforma Windows el cual se usa ya sea en casa, empresa o escuela.

Escritura de textos : Con un programa de procesamiento de palabras, el escribir consume menos esfuerzo que el que se hace utilizando algunos de los otros medios. Un procesador de palabras es un programa que procesa palabras, permitiendo teclear, editar, formatear, corregir e imprimir documentos, algunos programas pueden importar arte directamente a un archivo de texto. Con el procesamiento de palabras tenemos la ventaja de que el texto se ve en la pantalla y se parece mucho a la hoja impresa. Si se encuentran errores en la pantalla, simplemente se hacen las correcciones antes de imprimir el texto.

Mantenimiento de registros financieros: Un programa de hoja de cálculo reemplaza a la calculadora, de manera similar el programa organiza los datos en columnas e hileras. Los cálculos son automatizados, lo cual reduce el margen de error y nos permite jugar con situaciones financieras teóricas.

Escritura musical: El teclado de una computadora no se parece al teclado de un instrumento musical, pero con el equipo y el programa adecuado, se puede escribir y tocar música. Cada computadora tiene una bocina, la cual emite el sonido que se escucha cada vez que se oprime una tecla indebida. Esta capacidad de producir sonido es limitada, pero existe un tipo de conexión, llamado MIDI (Musical instrument Digital Interface- Interfaz digital de instrumento musical), que puede conectar un instrumento musical electrónico a la computadora, y ayudándonos de un programa apropiado, los sonidos se convierten en impulsos eléctricos que se almacenan en la computadora como un archivo. Después se le

puede editar, rearreglar y modificar la música. Mucha de la música que se escucha en la radio y la televisión es generada o editada en forma electrónica.

Generación de formas empresariales: Si una empresa o un empleo está sobrecargado en formas de factura, pedido, información a clientes, inventario, es preferible comprar un programa diseñado especificamente para ayudar a automatizar el trabajo de generación de formas, en el cual se puede encontrar logotipos o arte parecido al que se encuentra en los programas gráficos, texto como en un procesador de palabras, funciones de base de datos como el ingreso automático de datos, ordenación e indexado.

Materiales Publicitarios: La tipografía computarizada es hoy en día una fuerza nueva en el mundo de la publicidad, controlando todo el proceso, desde la escritura, hasta el diseño y la impresión, dando como ventajas:

a)Importación de arte. Los programas de tipografía pueden importar arte que haya sido generado en programas gráficos, e incluso

b)Arte prefabricado y avance rápido. Algunos programas de tipografía incluyen una biblioteca de arte prefabricado que se puede utilizar en las publicaciones ahorrando tiempo en el diseño.

Dibujo y artes gráficas: En las áreas de dibujo, ingeniería y las artes gráficas requieren la capacidad de trasladar visiones creativas al papel o la pantalla. En el caso de los planos que se crean con métodos de dibujo convencionales requieren de horas de trabajo duro y cambios. Independientemente del objeto ilustrado, los resultados deben mostrar al objeto desde varios ángulos con la mayor exactitud posible. Con los programas gráficos, el artista utiliza la pantalla como lienzo y selecciona de entre una variedad de herramientas, colores y paletas de patrones para ellos controlar el modo de representar los elementos en la pantalla.

Enseñanza por computadora: Las computadoras personales han tenido gran demanda en las escuelas. Hay programas educativos disponibles para todas las edades, todos los niveles de destreza, y casi para todos los temas. Los dibujos en color y los caracteres de caricatura dan vida al aspecto de estos programas, y la música y rutinas de generación de voz conservan el interés vivo. La mayor parte de los programas educativos se diseñan de manera que eviten la sensación de estar en la escuela. Los niños encuentran divertido el trabajo con lo computadora.

Juegos: Los juegos computarizados existieron a partir de la introducción de la primer computadora personal. Jugando el papel de máquina para agudizar los sentidos, poner a prueba la habilidad mental y de simples tácticas de diversión, la computadora con sus

juegos puede proporcionar un descanso cuando más se necesita un momento de relajamiento.

Estas ventajas y su gran difusión nos llevan a pensar en la utilidad que puede tener una red semántica mostrando a través de objetos, relaciones y herencias una pequeña parte del menú de Windows, demostrando de esta manera cómo las redes semánticas pueden estar presentes para representar el funcionamiento del software.

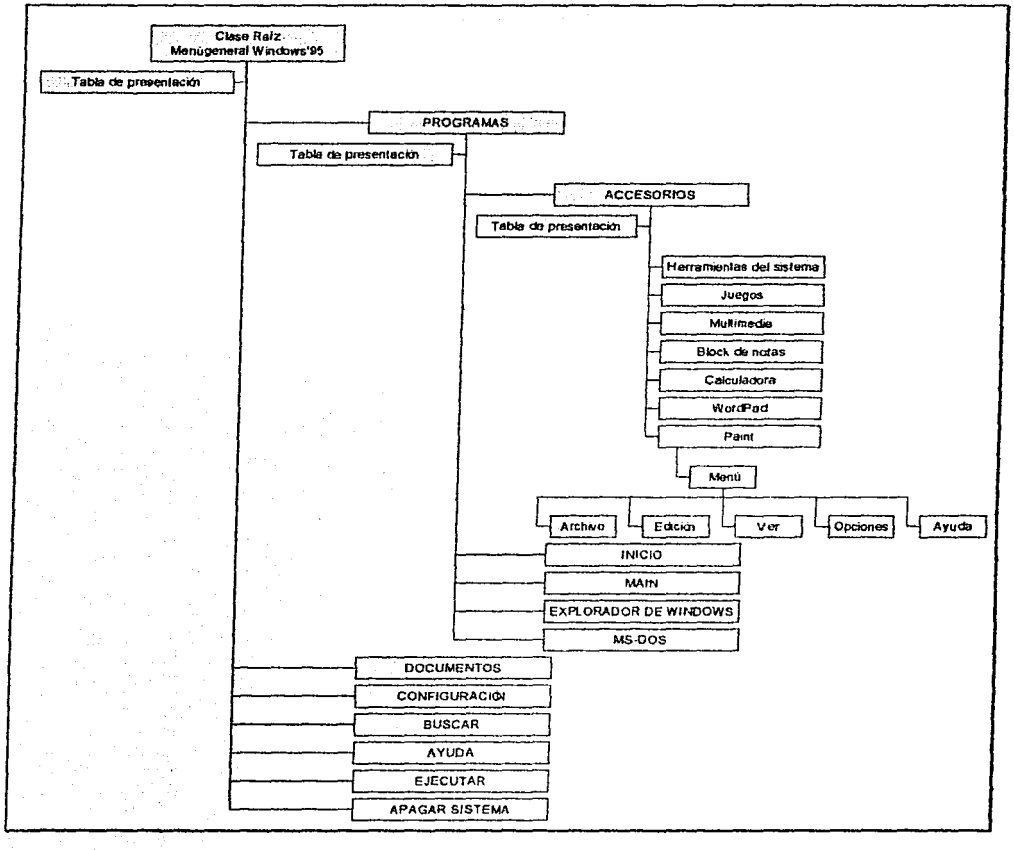
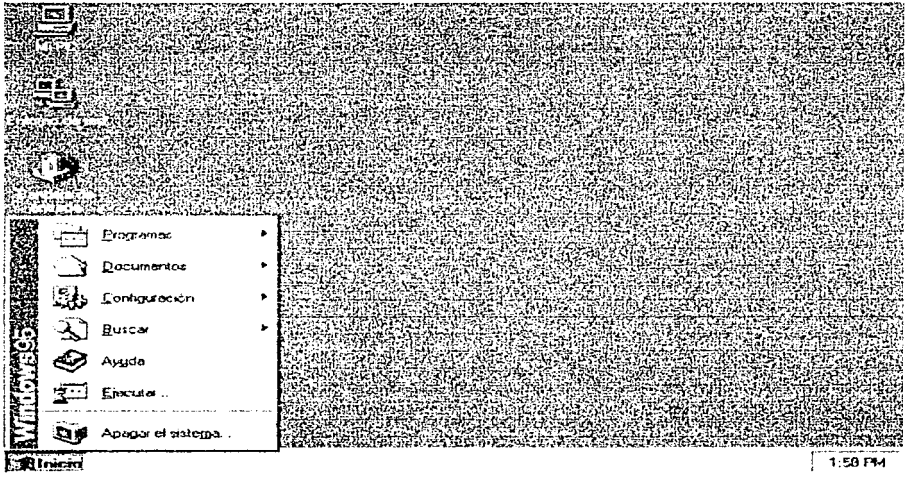


Fig.4.4. Red Semántica del menú de Windows'95

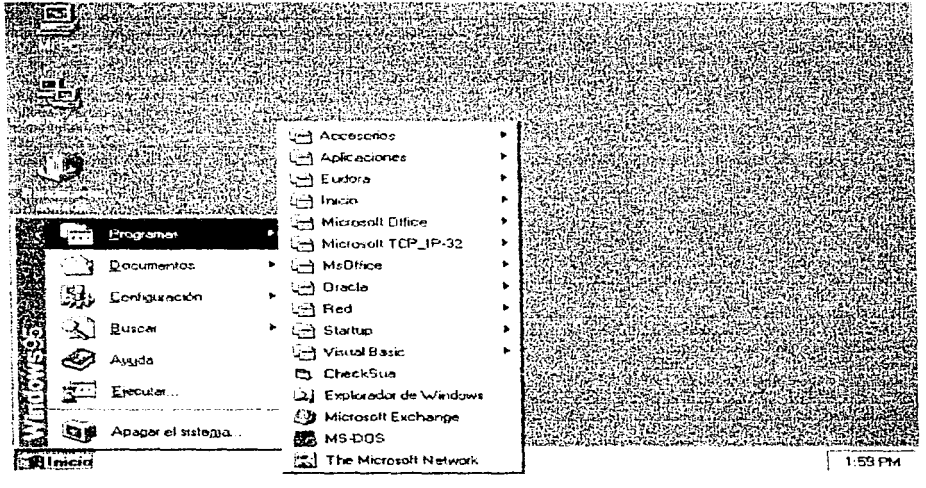
En la figura 4.4, la clase raíz "Menú principal o global Windows'95" tiene una Tabla de presentación que es llamada así, ya que en ella se tienen especificadas las características que deben llevar las pantallas para el menú principal y que serán heredadas a los hijos de la clase raíz o mejor conocidos como submenus.

A continuación vemos una pantalla del menú de Windows'95 para una mejor visión de lo mencionado (pantalla 1).



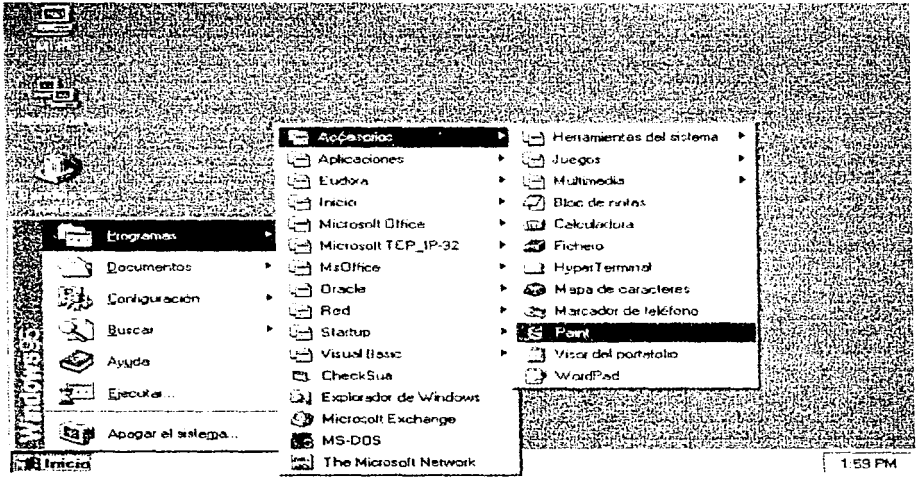
Pantalla 1

Si la opción a utilizar del menú principal fue PROGRAMAS, se nos presenta una lista de los programas que podremos iniciar, como se muestra en la pantalla 2.



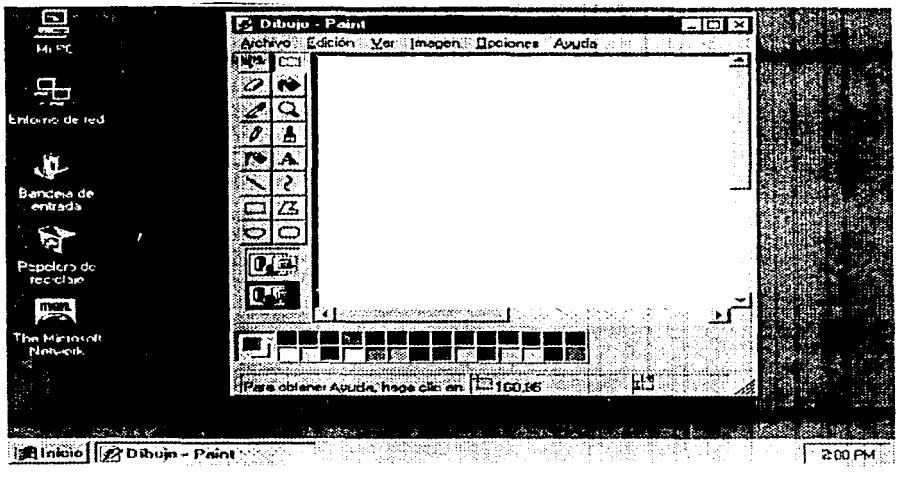
Pantalla 2

En la pantalla 2 vemos como el submenú de PROGRAMAS heredó las características de la Tabla de presentación del menú de Windows. De todas las opciones de éste submenú, escogemos ACCESORIOS, el cual tiene una Tabla de presentación como muestra la figura 4.4, que será heredada a sus hijos.



Pantalla 3

Una las opciones a seleccionar en ACCESORIOS vemos que es Paint (pantalla 3).



Pantalla 4

Paint heredó las características de la tabla de presentación de ACCESORIOS como lo vemos tanto en la pantalla 4 como en la figura 4.4, ya que se presenta al usuario los botones encontrados al extremo superior derecho de la pantalla, que indican minimizar, maximizar y cerrar, así como la barra de tareas localizada en la parte inferior de la pantalla.

Así mismo Paint también tiene su propio menú presentado en la parte superior de la pantalla 4, como en la figura 4.4. Si en el submenú Archivo al presentar un archivo que ya haya sido creado, la pantalla siempre heredará el menú de Paint.

Con todo ésto podemos entender que la utilidad de las redes semánticas en una parte de la computación es para representar herencia.

CONCLUSIONES

A lo largo de este trabajo, me di cuenta de que se ha investigado al ser humano para descubrir qué procesos sigue la información en su mente para que él pueda aprender, reconocer, razonar, entre otras actividades que lleva a cabo, y así llevar este modelo a las computadoras actuales y a los sistemas artificiales.

Pienso que un punto importante es que para que una máquina forme parte de un sistema artificial, tiene que tener en la memoria un conjunto de conceptos que van asociados unos con otros a través de una lista de características. Por lo que existe la necesidad de codificar reglas para manipular conceptos que lleven a inferir conocimientos. Lo anterior da paso a las redes semánticas, las cuales son utilizadas como herramientas visuales para representar y manipular conocimientos en los sistemas artificiales.

Las redes semánticas además de representar conocimientos, pueden ser usadas para interpretar de otro modo ideas y teorías que dieron paso al desarrollo de algunos procesos que forman parte de las computadoras actuales. Por otro lado, las redes semánticas sirven para representar información jerárquica que ayuda a la explicación del funcionamiento de la técnica orientada al objeto.

Sugiero que esta Tesina se tome como ayuda didáctica para estudiantes que lleven asignaturas relacionadas con producciones, ya que se plasma la utilidad de este concepto en diferentes formas de uso.

GLOSARIO

Arco: representación de relaciones entre nodos.

Artificial: sintético, imitar o duplicar fenómenos.

Base de conocimientos: es la representación simbólica de conocimientos relacionados con hechos e información general, así como heurísticos tales como juicios, intuición y experiencias sobre determinada área específica. Almacena conocimientos causa-efecto, reglas e información imprecisa y probabilística.

Gramática: reglas que enseñan a hablar y escribir correctamente.

Inteligencia: es la habilidad para resolver problemas, aprender, salir airoso ante lo desconocido. Así hablamos de juicio sabiduría, perspicacia, percepción y aprendizaje.

Inteligencia sintética: es la capacidad de una máquina para imitar las cosas que, por ahora hacen los seres humanos mejor como el percibir, razonar y actuar.

Lenguaje: cualquier medio que se utiliza para expresar ideas.

Máquina: instrumento que facilita la actividad del ser humano.

Memoria: lugar en donde se almacena la información obtenida a través del tiempo.

Nodo: objetos, conceptos o situaciones en un campo determinado.

Programa: Conjunto de instrucciones y reglas para el funcionamiento de un sistema.

Red. Conjunto de nodos relacionados entre si por medio de arcos.

Semántica: Es la significación de un contexto.

Sistema: Conjunto de componentes que interactuan entre si como un todo con un objetivo determinado.

Sistemas Expertos: Son los programas informáticos que incorporan representaciones cognitivas y técnicas estructurales para que programas de computadora muestren comportamientos que normalmente asociamos con la inteligencia.

BIBLIOGRAFÍA.

ALFRED, BINET
LAS IDEAS MODERNAS SOBRE LOS NIÑOS
ED. FONDO DE CULTURA ECONÓMICA, S.A.DE C.V., MÉXICO. 1985, P. 358

ALFRED, V. AHO
COMPILADORES PRINCIPIOS, TÉCNICAS Y HERRAMIENTAS
ED. ADDISON WESLEY IBEROAMERICANA ESTADOS UNIDOS, 1990, P. 820

DANIEL I. A. COHEN
INTRODUCTION TO COMPUTER THEORY
BY JOHN WILEY & SONS INC, UNITED STATES OF AMERICA, 1986, P 823

ENRIQUE, CASTILLO
SISTEMAS EXPERTOS , APRENDIZAJE E INCERTIDUMBRE
ED. PARANINFO, S.A., MADRID, 1989, P. 335

HERBERT, SCHILDT
TURBO C/C++ MANUAL DE REFERENCIA.
ED MC GRAW HILL, MÉXICO, 1994, P 874

JOHN, HAUGELAND.
LA INTELIGENCIA ARTIFICIAL
ED. SIGLO VEINTIUNO EDITORES, MÉXICO, 1988, P. 255

JOHN, E. HOPCROFT
INTRODUCCION A LA TEORÍA DE AUTOMATAS , LENGUAJES Y COMPUTACION
COMPAÑÍA EDITORIAL CONTINENTAL, S.A. DE C.V., MÉXICO 1993

KATHERINE, MURRAY
MANUAL DE INTRODUCCIÓN A LAS COMPUTADORAS PERSONALES
PRENTICE HALL HISPANOAMERICANA, S.A. , TERCERA EDICIÓN, TOMO I, MÉXICO 1992, P. 245

LAWRENCE ,S. ORILIA
LAS COMPUTADORAS Y LA INFORMACION
ED MC GRAWN HILL TERCERA EDICIÓN MÉXICO 1988, P. 744

MARY, P. DOLCIANI
ALGEBRA MODERNA ESTRUCTURA Y METODO
ED.PUBLICACIONES CULTURAL S.A , VOLUMEN I, MÉXICO 1989, P 591

RAFAEL, ARECHIGA G
FUNDAMENTOS DE LA COMPUTACIÓN
ED LIMUSA SEGUNDA EDICIÓN MEXICO, P. 391

RICH, E.
INTELIGENCIA ARTIFICIAL
ED MAC GRAW HILL MÉXICO 1988, P. 311

SIMONS, G L.
INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL
ED DIAZ DE SANTOS MÉXICO 1990, P 400

WENDY , B. RAUCH
APLICACIONES DE LA INTELIGENCIA ARTIFICIAL EN LA ACTIVIDAD EMPRESARIAL, LA CIENCIA Y LA INDUSTRIA
EDICIONES DÍAS DE SANTOS, S A , MADRID 1989, P.683

MANUAL DEL USUARIO WINDOWS & MS-DOS PARA TRABAJO EN GRUPO
ED.MICROSOFT CORPORATION, MÉXICO 1993, P 305

INTRODUCCIÓN A MICROSOFT WINDOWS'95
ED MICROSOFT CORPORATION MÉXICO 1994, P 87

DICCIONARIO ENCICLOPÉDICO SALVAT UNIVERSAL
ED SALVAT EDITORES, S A , TOMO 18, ESPAÑA 1976

DICCIONARIO ENCICLOPÉDICO SALVAT UNIVERSAL
ED. SALVAT EDITORES, S.A , TOMO 17, ESPAÑA 1976