11
2ej.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

CAMPUS ARAGON

" PROLOG APLICADO A LA INTELIGENCIA ARTIFICIAL "

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO EN COMPUTACION

PRESENTAN:

CORCHADO SALINAS CLAUDIA DOLORES
LOPEZ ZAMUDIO NORA ANGELICA

ASESOR : ING. AMILCAR A. MONTERROSA ESCOBAR

AGOSTO 1998

TESIS CON
FALLA DE ORIGEN

26426

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Les doy las gracias a:

DIOS:

Por darme todo lo que tengo y he logrado durante mi vida. Gracias Señor por mis padres, mi hermano, mi familia, mis amigos y por la fé que tengo en tí.

MIS PADRES:

Josefina y **José Inés** por todo su apoyo, comprensión, dedicación, esfuerzo, amor y cuidado que simpre me han dado. Y sobre todo por ser el motivo pricipal de mi superación y por darme la más grande de las herencias, una profesión.

MI HERMANO:

Juan José por su apoyo y comprensión que siempre me ha brindado.

LA FAMILIA LOPEZ ZAMUDIO:

Por todo el apoyo brindado a mi y a mi familia.

MIS AMIGOS:

Por su amistad y apoyo que me brindaron durante mi preparación profesional, en especial a **Claudia**, mi compañera de tesis.

MIS PROFESORES:

Por brindarme sus conocimientos y experiencias para lograr mi formación profesional, en espcaial a mi asesor de tesis **Ing. Amilcar A. Monterrosa Escobar**.

LA UNAM Y A LA ENEP ARAGON:

Por la oportunidad que me brindaron para mi formación profesional.

Y EN MEMORIA:

A mi abuelito **José** por haberme dado simpre la alegría de vivir y ser un ejemplo de lucha.

De todo corazón:

NORA ANGELICA

La máxima obra propuesta al ser humano es la de forjarse un futuro.

Alejo Carpentier.

Haz lo necesario para logra tu más ardiente deseo y acabarás lograndolo.

Beethoven.

Al llegar a este momento de mi vida hago un recuento de todo lo que he vivido y de todas las personas que de una o de otra manera han intervenido durante mi formación profesional Quisiera agradecerles a cada una de ellas todo lo que hicieron por mi pero estoy segura de que no mencionaría a todas; sin embargo mencionare a dos seres que durante toda mi vida han sido parte importante en todo momento de mi vida

Dios:

Gracias por darme la oportunidad de haber nacido, de haber crecido en una familia que siempre me ha apoyado, de darme todo lo necesario para poder vivir cada día, de darme la oportunidad de llegar a este momento

Cande:

Sabes muy bien lo que significas para mí y lo importante que has sido para lograr mis metas, eres la persona a la que yo más quiero Nunca podré de dejarte de dar las gracias por todo, acepta este pequeño homenaje de tu hija

> " Hay una mujer que tiene algo de Dios por la inmensidad de su amor Y mucho del ángel por la incansable solicitud de sus cuidados. Una mujer que siendo joven tiene la reflexión de una anciana y en la vejez, trabaja con el vigor de la juventud, una mujer que si es ignorante descubre los secretos de la vida con más acierto que un sabio y si es instruida se acomoda a la simplicidad de los niños Una mujer que siendo pobre se satisface con la felicidad de los que ama y siendo rica, daría con gusto su tesoro por no sufrir en su corazón la herida de la ingratitud, una mujer que siendo vigorosa se estremece con el vagido de un niño y siendo débil se reviste a veces con la bravura del león, una mujer que mientras vive no la sabemos estimar porque a su lado todos los dolores se olvidan pero despues de muerta, daríamos todo lo que somos y todo lo que tenemos por mirarla de nuevo un solo instante, por recibir de ella un solo abrazo, por escuchar un solo acento de sus labios "
>
> Ramón Angel Jara

Indice

Introducción

No existe una definición para la Inteligencia Artificial que sea totalmente aceptada, ya que ni siquiera existe una definición de lo que es la inteligencia humana, sin embargo, es considerada como una rama de la informática que intenta hacer que una computadora pueda pensar y razonar como una persona

El presente trabajo de tesis muestra como a través del lenguaje de programación PROLOG, se pueden desarrollar aplicaciones de la Inteligencia Artificial Para lograr este objetivo el trabajo esta dividido en tres partes

La primera parte presenta un panorama general de lo que es la Inteligencia Artificial, así como la investigación y desarrollo de la misma También da una introducción a las áreas de estudio de la Inteligencia Artificial y los lenguajes que se utilizan para desarrollar aplicaciones en dichas áreas Además de la representación del conocimiento que permite resolver problemas complejos en la Inteligencia Artificial

La segunda parte, describe uno de los lenguajes de programación más utilizados en la Inteligencia Artificial, PROLOG, así como la sintaxis y semántica de este lenguaje

Las áreas de la Inteligencia Artificial se explican con más detalle en la tercera parte Dichas areas son

- Busqueda de Soluciones
- Sistemas Expertos
- Procesamiento de Lenguaje Natural
- Percepción Visual
- Robótica

En cada uno de los capítulos de este apartado se describen los fundamentos principales de las áreas listadas anteriormente Se incluye además el desarrollo de una aplicación por cada área El listado de los programas se adjunta en el apartado de anexos

Parte 1 Inteligencia Artificial

Capítulo

1

Introducción

Desde que se inventó la computadora, se pensó que algún día ésta podría llegar a poseer "facultades intelectuales" iguales o superiores a las del hombre

Al estar considerada la Inteligencia Artificial como algo poco complejo, se pensaba que una máquina inteligente podría llegar a contar con facultades similares a las de los seres humanos. John von Newman, arquitecto de la primera computadora, quedó fascinado a tal punto por la similitud entre máquina y cerebro, que hasta llegó a llamar neuronas a los módulos de cálculo de las primeras computadoras Esto es lo que dio lugar, en un principio, a que a las computadoras se les diera el nombre de cerebros electrónicos

> " Imaginemos que nos encontramos en el futuro, un estudiante está en la biblioteca recogiendo información para un proyecto de investigación. Con no mucha seguridad, teclea: "¿Tienes alguna información sobre la industrialización del siglo XIX?", preguntándose si la computadora entenderá la pregunta. Pero en la pantalla aparece: "Ciertamente, ¿qué es lo que necesita saber?"
>
> "Lo que necesito saber - reflexiona en voz alta - es cómo la invención del telar Jacquard influenció las posteriores tendencias a la automatización". Para su sorpresa, la computadora le contesta: "¿Quiere usted esta información estructurada en la forma de un informe formal o le gustaría mantener una conversación interactiva?"
>
> Desgraciadamente es hora de volver al presente. En la actualidad no hay ninguna computadora que ofrezca las posibilidades de la máquina maravillosa. " (1)

La parte de la informática referida al diseño de las computadoras inteligentes es la llamada **Inteligencia Artificial (IA)**, que está saliendo del entorno de los laboratorios de investigación para ocupar un lugar en distintas actividades de la vida diaria

La Inteligencia Artificial se ha caracterizado, desde sus inicios, por la gran actividad en la investigación, y en la actualidad ésta es mayor que nunca El sector privado está incrementando su apoyo a la investigación en este campo y está creciendo el interés en el uso y comercialización de los programas de IA. De hecho, hay ya grupos de investigación en IA sólidamente fundados en los Estados Unidos y otros países industrializados En México existe el Centro de Inteligencia Artificial (CIA) enfocado a resolver problemas de la industria, y la Sociedad Mexicana de Inteligencia Artificial (SMIA) dedicada a la difusión de la IA en el país

La IA goza de una gran difusión, e incluye aplicaciones a los negocios, tales como sistemas expertos y los más inteligentes procesadores de textos, así como ciertas aplicaciones sobre visión de robots y reconocimiento del habla Hay que señalar que el reconocimiento de modelos es una parte fundamental de varias áreas de la IA El reconocimiento de modelos en general no se limita a la visión de las máquinas, sino también se da en el habla, en el sentido del tacto, en los juegos y en muchas otras areas que se usarán en la robótica y en las computadoras del futuro.

Una de las aplicaciones más interesantes es el reconocimiento y el procesamiento de las formas que se obtienen al filmar mediante cámaras de video Este procesamiento puede aplicarse al recuento de células sanguíneas en un microscopio, también ayudar a la mano de un robot a localizar su objetivo, o en general a detectar cuántos cuadrados, círculos u otras figuras geométricas hay en determinada imagen Otro uso importante es la detección de errores en los procesos de manufacturación

Gracias a estos trabajos, se ha logrado conocer más a fondo las similitudes existentes entre el cerebro humano y la computadora Esta mayor comprensión de la inteligencia humana ha mejorado considerablemente el rendimiento de las computadoras.

Los avances de la Inteligencia Artificial se han visto siempre limitados por la tecnología de las computadoras existentes Los programas de IA normalmente necesitan más recursos de programación que otros programas, de hecho algunas teorías de IA no han sido implementadas como software, debido a que las computadoras existentes no han sido lo suficientemente potentes

Historia

Antes de que existiera un campo de conocimiento llamado Inteligencia Artificial y mucho antes de que existieran las computadoras o la electrónica, ya se pensaba en la posibilidad de crear individuos con inteligencia

A lo largo de la historia podemos encontrar intentos de creación de seres inteligentes en la mitología griega, en la Europa medieval, en películas de ciencia ficción tales como Frankestein, 2001: Odisea en el espacio y La guerra de las galaxias.

Durante la II Guerra Mundial, los americanos y británicos fueron los primeros en usar las computadoras para tareas tan complejas como cálculos numéricos e interpretación de claves, actividades que anteriormente se suponía que requerían inteligencia humana. El matemático Alan Turing formo parte del Proyecto Ultra, él cual logró la interpretación de la clave alemana "Enigma"

Turing ayudó a diseñar una de las primeras computadoras que se llegaron a construir, además escribió varios artículos sobre distintos aspectos teóricos de las matemáticas y el cálculo En su artículo "Sobre el cálculo numérico", de 1937, describe cómo una máquina hipotética (actualmente llamada máquina de Turing) podría usar un sistema de códigos binarios para realizar cualquier operación algorítmica.

En 1950 Turing escribió un artículo titulado "Computing Machinery and Intelligence" ("Computadoras e Inteligencia"), por el cuál se le reconoce como el "padre" de la Inteligencia Artificial En uno de sus artículos escribió " "Yo propongo la siguiente cuestión: ¿Pueden las máquinas pensar?" la cual continúa siendo debatida aun hoy en día, aunque él mismo llegó a la conclusión de que era una pregunta sin sentido " (2)

Turing predijo que habría muchas objeciones a la proposición de que la máquina pudiera pensar De hecho, en el mismo artículo consideró y respondió a algunas de estas posibles objeciones Turing, dándose cuenta de las dificultades que implicaba el intento de responder a esa pregunta, propuso una prueba en la forma de un juego que podía ayudar a esclarecer el asunto. Le llamó el **"juego de imitación"**, pero en su honor se le conoce con el nombre de **"prueba de Turing"**

En la misma epoca otros matemáticos y lógicos, fundamentalmente en Gran Bretaña y los Estados Unidos, también se hacían el mismo tipo de preguntas acerca del cerebro y las computadoras

En 1943, McCulloch compartió la autoría de un artículo titulado "Cálculo lógico de las ideas relacionadas en la actividad nerviosa", publicado en el Boletín de Biofísica Matemática Para entonces McCulloch era el director del laboratorio de Investigación Básica en el Departamento de Psiquiatría de la Universidad de Illinois El artículo proponía que una red de neuronas en el cerebro trabajaba de una forma análoga a una hipotética máquina de Turing, es decir, podía ser útil el considerar el cerebro como una computadora

Conforme se conocen más cosas del cerebro, la correspondencia sugerida por McCulloch entre cerebros y máquinas ha desaparecido Sin embargo, las descripciones de las semejanzas entre computadora y cerebro contribuyeron enormemente al desarrollo de un campo llamado ciencia de la información y asímismo al de la Inteligencia Artificial

En 1854, el matemático británico George Boole propuso un sistema para describir la lógica en términos matemáticos, la cual se convirtió en la llamada Algebra booleana En 1937, Claude Shannon usó el álgebra booleana para describir el comportamiento de los circuitos eléctricos de conmutación Las ideas de Shannon contribuyeron al desarrollo de la ciencia de la información y al almacenamiento de información en la computadora digital con el sistema binario. Sospechaba que si el álgebra booleana podía ser usada para describir los circuitos eléctricos, quizá los circuitos podían ser usados para describir el pensamiento

Otra aportación de Shannon, es un artículo que apareció en el Scientific American en 1950, donde discutía la posibilidad de usar computadoras para jugar ajedrez Fue uno de los primeros que dio a entender que poseer una computadora que tuviera en cuenta todas las posibles combinaciones de movimientos que se podían hacer no era una estrategia práctica para jugar ajedrez. Hizo una estimación considerando que, aún si la computadora podía evaluar un millón de movimientos por segundo, una computadora tardaría 10^{95} años en seleccionar un movimiento

Norbert Wiener es muy conocido por desarrollar una aproximación al entendimiento de cómo trabaja el universo Wiener sugirió un modelo valioso para entender tanto a las computadoras como a las personas, en el que exponía que la transferencia de información era la mejor manera de modelar muchas formas distintas de fenómenos científicos **Cibernética** fue el nombre dado por Wiener para describir su acercamiento a los fenómenos a través de la información. Al describir los sistemas interrelacionados en función del intercambio de información, la cibernética apunta a las semejanzas funcionales entre los hombres y las máquinas, que demostraron ser enormemente valiosas en las primeras investigaciones llevadas a cabo en Inteligencia Artificial

La revolución de la IA se inició en el verano de **1956** en la pequeña ciudad universitaria de Hanover, New Hampshire (EE UU), donde se encontraba el Dartmouth College En esté lugar se llevó a cabo la famosa **Conferencia de Dartmouth**, donde se reunieron algunos científicos representantes de diferentes disciplinas como matemáticas, neurología, psicología e ingeniería electrica Los cuales utilizaban la computadora para realizar sus investigaciones

Una nueva rama de la informática se definió en esta conferencia, combinando elementos de distintas áreas de investigación en un campo unificado No se llegó a un acuerdo de como se podía llamar a esta nueva ciencia, sin embargo, **Inteligencia Artificial** fue el nombre sugerido por **John McCarthy**

La Conferencia estuvo organizada por cuatro científicos, de los cuales dos provenían de la universidad y los otros dos de la industria, sus nombres son **John McCarthy, Marvin Minsky, Nathaniel Rochester y Claude Shannon** De acuerdo con su propuesta, la Conferencia debía examinar la posibilidad de cada aspecto del aprendizaje o cualquier otra característica de la inteligencia que pudiera ser descrita con tanta precisión que se logrará construir una máquina que la simulara Esta posibilidad sigue siendo el objetivo de la investigación en IA.

John McCarthy ha sido el responsable de varios de los desarrollos más importantes en IA y en informatica. En **1958** creó **LISP (Procesador de Listas)**, que es el lenguaje de programación que se usa normalmente en IA Ha sido ganador del premio Turing por sus contribuciones transcendentales a la informática. La aportacion más significativa es quizá un proceso llamado **tiempo compartido**. Este proceso permite el uso de una computadora por varios usuarios al mismo tiempo, el cual es esencial en la informática moderna Inició también la investigación en robótica en Stanford y trabajo sobre la formalización del conocimiento y del razonamiento en el sentido común

Marvin Minsky y **Jhon McCarthy** fundaron el laboratorio de Inteligencia Artificial en **MIT** *Minsky es conocido por su trabajo en la organización y representación de estructuras de* conocimiento, un área con importantes investigaciones en las distintas aplicaciones de IA.

Allen Newell y **Herbert Simons** llegaron a la Conferencia con un programa de IA llamado **Teórico Lógico**, un programa de procesamiento de la información En colaboración con **J. C. Shaw** desarrollaron este programa para generar demostraciones de teoremas matemáticos En particular eligieron los teoremas de Principia Mathemática, un sistema de cálculo proposicional (inferencia lógica) propuesto a principios de este siglo por Bertrand Russell y Alfred Norton Whitehead Además del Teórico Logico, Newell y Simon también desarrollaron con J C Shaw, el **IPL (Lenguaje de Procesamiento de la Información)**, que fue el precursor de LISP. También crearon un programa para la Resolución General de Problemas, escogiendo un camino adecuado para la obtención de un objetivo específico

Arthur Samuel creó un programa al que se le suele llamar el **Jugador de Damas de Samuel**. La característica más importante de este programa es que fue el primero capaz de aprender de sus propios errores

Alex Bernstein programó un **juego de ajedrez** El programa de ajedrez de Bernstein es importante en la historia de la IA, porque uso técnicas heurísticas para la búsqueda de los mejores movimientos La complejidad del juego de ajedrez impuso la necesidad de conseguir distintos métodos para ir eliminando posibilidades, siendo éste un concepto fundamental en la investigacion en IA.

Investigación y Desarrollo

Durante muchos años la investigación en IA fue llevada a cabo en unas cuantas instituciones, fundamentalmente centros de investigación en las universidades En la década de los setenta, el número de estos centros aumentó considerablemente y en la actualidad la IA tiene una gran difusión.

Algunas instituciones dedicadas a la investigación son las siguientes:

- **AIC SRI International`s Artificial Intelligence Center (AIC)**

Es uno de los mejores centros del mundo de investigación en Inteligencia Artificial. Fundada en 1966, el AIC ha sido un pionero y un colaborador en el desarrollo de herramientas de computación para el comportamiento inteligente en situaciones complejas. Sus objetivos son entender los principios computacionales inteligentes fundamentales en hombres y máquinas, y desarrollar métodos para construir sistemas basados en computadoras para resolver problemas, para comunicarse con la gente, y para percibir e interactuar con el mundo físico.

El Centro esta asociado con universidades y otros grupos de investigación, y ofrece oportunidades a estudiantes a participar en proyectos en el extranjero

Los tres programas de investigación del AIC que se han establecido como proyectos se encuentran en las siguientes areas de Inteligencia Artificial

- Lenguaje Natural
- Percepción
- Representación y Razonamiento

- **Institute Technology Massachussets (MIT)**

El principal objetivo del laboratorio de IA es entender la manera de lograr que las computadoras puedan mostrar inteligencia Otros de sus objetivos son construir sistemas inteligentes y entender aspectos de inteligencia biológica La investigación en el laboratorio incluye trabajos en robótica, visión, lenguaje natural, aprendizaje, razonamiento y resolución de problemas, modelo-basado en sistemas expertos, ingeniería de diseño, supercomputadora y teoría básica

Algunas de las investigaciones son las siguientes

1. Visión

El grupo del profesor Grimson se ha orientado principalmente en métodos de reconocimiento de objetos en obstrucción, ruido, medio ambiente no estructurado. Así se han desarrollado los sistemas mano-ojo, de navegación de vehículos autónomos y control de procesos para partes industriales Recientes esfuerzos estan enfocados a procedimientos formales para la evaluación de métodos de reconocimiento alternativo, a métodos grupales para el preprocesamiento de datos de entrada dentro de la regla de atención visual en reconocimiento de imágenes y métodos de agrupación y reconocimiento para identificar objetos Estos esfuerzos se estan integrando dentro de un sistema móvil ojo-cabeza para encontrar objetos ocultos en una habitación y para atención focal en puntos de interes de la misma.

Un proyecto aplica estos métodos para acrecentar el funcionamiento del cerebro, por medio de imágenes ordinarias surgidas de un paciente, con imágenes producidas por resonancia magnética. Estos métodos de visualización se utilizan en neurocirugía y como parte de un estudio clínico de esclerosis multiple.

El profesor Horn y sus estudiantes trabajan sobre problemas de movimiento de visión La extensión de los métodos existentes en la dirección del tiempo están siendo explorados. Mientras se obtiene información de movimiento sobre dos estructuras de imagen, la distancia de los objetos es determinada arbitrariamente Las mejoras de precisión de los objetos reconstruidos son logrados basándose en la forma, posición y el movimiento estimados en el tiempo presente, después de diez mejoras los errores introducidos por la fase de predicción empiezan a balancear los progresos obtenidos de la situación continua en tiempo

Finalmente, se trabaja en una propuesta especial de visión-temprana con el chip VLSI. Este chip determina el foco de expansión, que apunta alrededor de la imagen Esto produce la necesidad de detectar y analizar las caracteristicas de la imagen. Se espera que el chip pueda operar a mil estructuras por segundo Cuenta con un arreglo de sensores de 32 por 32 y proceso de elementos que son usados para poder recuperar el foco de expansión con resolución de subpixeles. Trabajan también para que el chip pueda distribuirse con combinaciones arbitrarias de movimientos traslacionales y rotacionales, y que la escena sea vista en forma plana.

2. Máquinas Aprendiendo y Redes

Las investigaciones del profesor Poggio se enfocan sobre el problema de aprendizaje de funciones multivariante de datos esparcidos El grupo del profesor desarrollo un marco teórico, basado en la teoría de regularización, que origino la teoría clásica de la función de la aproximación. El esfuerzo en la investigación en el laboratorio de IA en conjunto con el centro de aprendizaje computacional y biologico continua creciendo en algunas direcciones ediciones de teoría y matemáticas, algoritmos de aprendizaje, uso de información anterior para argumentar, serie de ejemplos de exploración virtual, aplicaciones y neurociencias

El doctor Federico Girosi, Mitchel Jones y el profesor Poggio elaboraron un marco teorico de arquitecturas de redes neuronales y técnicas estáticas El doctor Girosi y el profesor Anzellotti demostraron que algunas arquitecturas de redes neuronales y técnicas de aproximación funcional están estrechamente relacionadas también desde el punto de vista del curso de dimensionalidad

Algunas aplicaciones han sido demostradas en reconocimiento de objetos, gráficas de computadoras, conferencias de video, detección de características, detección de blancos, inspección visual, detección de fase, análisis de series de tiempo y modelos en finanzas El doctor David Beymer y el profesor Poggio probaron un sistema de reconocimiento de fase, basado en una postura invariante usando algunas visiones de personas y corriendo sobre la maquina, en forma paralela, una base de datos de casi mil fases de imágenes El rango de reconocimiento es cerca del 98% en una base de datos de 68 personas

3. Entendiendo el Lenguaje Natural y adquisición

El profesor Berwich y sus colegas terminaron la construcción de la siguiente generación del lenguaje natural parses y traductores, basados en las teorías linguísticas modular y universal Durante el año pasado, un parser modular en japonés fue desarrollado para cubrir un largo subconjunto de este lenguaje y el sistema también se amplio en holandés, alemán e italiano Adicionalmente, en colaboración con los investigadores del departamento de linguística y filosofía, el parser universal se extendió para manipular el lenguaje de Arabia e India.

El profesor Berwich también dirigió un grupo de investigación especial en desarrollo de herramientas multilinguísticas para construir diccionarios en diferentes lenguas como el hindú, belga, griego y japonés

En el caso de la Inteligencia Artificial en México, se han formado asociaciones de investigación, formación académica y divulgación Existen trabajos de gran calidad en Solución Cooperativa de Problemas, Sistemas Expertos, Programación Lógica y Redes Neuronales.

- **Sociedad Mexicana de Inteligencia Artificial (SMIA)**

La SMIA es una sociedad científica y profesional dedicada al apoyo y la difusión de la IA en nuestro país.

Desde su formación organiza una reunión nacional, y más recientemente, un simposium internacional anual de aplicaciones de la Inteligencia Artificial, un congreso iberoamericano bianual y numerosos coloquios, seminarios, conferencias y simposiums

En el verano de 1983 en la Ciudad de Xalapa y dentro de la III Semana de Computadoras en la Educación, evento organizado por la Fundación Arturo Rosenblueth, se comentó sobre la organización de una reunión de trabajo sobre el tema de la inteligencia artificial en México A este evento concurrieron diversas instituciones académicas, entre las que figuran la UNAM, la UAM , la UAM-I, la FAR, el IPN y la UDLA Los organizadores de este evento fueron el Dr Enrique Calderón Alzati (FAR), el Dr José Negrete (UNAM-IIB) y el Mtro José Luis Mora (UAM-I).

En esta reunión se seleccionó a la Cd de Guanajuato para llevar a cabo la primera reunión de trabajo de IA, en marzo de 1984 En la Asamblea de fundadores realizada en agosto de 1986, en las instalaciones de la FAR con 25 asistentes, resultó nombrado presidente el Dr. José Negrete.

La Sociedad Mexicana de Inteligencia Artificial, queda constituida como Asociación Civil en 1987, y organiza la cuarta reunión nacional de IA en la Cd de Puebla en marzo de 1987, con una asistencia aproximada de 180 participantes entre estudiantes, ponentes y asistentes, a partir de lo cual se consideró que las reuniones deberían hacerse en la provincia mexicana con el fin de recorrer el país difundiendo esta área del conocimiento en las diferentes universidades e instituciones.

En un viaje de trabajo realizado por el Presidente y el Vicepresidente de la SMIA, Dr José Negrete y Act Pablo Noriega, a España y Portugal, se concretó la realización de la Reunión Iberoamericana de IA, coorganizada por la SMIA, la asociación Española por la IA (AEPIA) y la Asociación Portuguesa de IA (APIA); la reunión fue convocada para enero de 1988, en la ciudad de Barcelona, denominándola IBERAMIA 88

Entre los temas mas destacados en las reuniones son acerca de sistemas expertos, lenguaje natural, programación lógica y redes neuronales Por otro lado la UNAM, la UDLA, el IPN y el ITESM son las instituciones que han participado con mayor frecuencia

La SMIA ha llevado a cabo reuniones mensuales de difusión de los temas de la IA, cuenta con algunas publicaciones de divulgación, así como apariciones en la radio y televisión mexicana, tiene relaciones con diversas asociaciones de IA en España, EUA, Canadá y Sudamérica.

Finalmente, la SMIA se integró al directorio de IA para Norteamérica, que junto con la AAAI (American Association For Artificial Intelligence), la CAIA (Canadian Association of Intelligence Artificial) y el SIGAI-ACM, se publica con el fin de tener un directorio integrado de las instituciones científicas interesadas en el tema en el norte de nuestro continente

- **Centro de Inteligencia Artificial (CIA)**

El Centro de Inteligencia Artificial (CIA), tuvo origen en el ITESM Campus Monterrey, se dedica a la formación de recursos humanos, a la investigación científica de tipo básica y aplicada a la extensión en el campo de la Inteligencia Artificial (IA) y disciplinas asociadas

Las actividades académicas en Inteligencia Artificial (IA) en el ITESM Campus Monterrey inicia durante la década de los años setenta con iniciativas de algunos profesores quienes llevan a cabo actividades docentes y de investigación y desarrollo, en conjunto con investigadores nacionales y extranjeros en esta disciplina

A principios de los años ochenta la IA empieza a aplicarse en la solución de problemas reales con los sistemas expertos Los sistemas expertos se declaran como una línea de investigación prioritaria del Centro de investigación en Informática En 1984 se ofrecen los primeros cursos de posgrado en IA y en 1985 se forma un grupo de investigación de IA que se dedica principalmente a la docencia, la asesoría de tesis, y la asimilación de la tecnología para aplicarla en la solución de problemas médicos e industriales, además atender la especialidad de IA en la Maestría en Ciencias Computacionales.

Entre 1987 se inician los primeros proyectos con el Grupo CYDSA consistente en la aplicación de sistemas expertos para resolver problemas de mantenimiento de maquinaria industrial, entre 1988 y 1989 crece la relación con otras empresas, y en 1989 se consolida la disciplina con el establecimiento del Centro de Inteligencia Artificial para llevar a cabo formación de recursos humanos a nivel posgrado, realizar investigación básica y aplicada, y transferir tecnología de IA al medio empresarial

En 1991 se inicia una especialidad en IA en el Doctorado en Informática y se continua con la especialidad en IA en la Maestría en Ingeniería de Sistemas Computacionales

A partir de 1995 se ofrece el Doctorado en Inteligencia Artificial, la Maestría en Sistemas Inteligentes, especialidades en IA en las maestrías de Tecnología Informática e Ingeniería de Control, y se inicia una estrategia de generalización para utilizar la tecnología de IA como una herramienta de análisis y solución de problemas complejos en las carreras profesionales y programas de graduados del ITESM Campus Monterrey

Los objetivos que persigue el CIA son los siguientes

1 Formar recursos humanos en IA a nivel de doctorado, maestría y profesional con un enfoque de especialización en IA aplicado en áreas de ingeniería, informática y administración a través de programas académicos.

2 Realizar investigaciones científicas en IA del tipo básica y aplicada, con financiamiento de agencias gubernamentales, internacionales y empresariales, para generar nuevos conocimientos ya existentes a través de los grupos de investigación del CIA

3 Realizar extensión en las siguientes áreas

- Consultoría
- Desarrollo de productos genéricos para su posterior comercialización
- Diplomados y seminarios de capacitación profesionales
- Congresos internacionales Simposium Internacional de IA, Congreso Mundial de Sistemas Expertos, comités de programas de diversas conferencias internacionales

Definición

> La *inteligencia* es la capacidad de una persona para razonar o pensar. Es la capacidad de adquirir conocimientos y de aplicarlos. Los humanos tienen inteligencia innata y pueden aumentarla y mejorarla." (3)

Existen algunas características sugeridas por Douglas Hofstadter como ""actitudes esenciales de la inteligencia" :

1. *Responder de una manera muy flexible a las situaciones.* No se responde de la misma forma siempre que uno se enfrenta a un problema idéntico, si se hiciera así se tendría una conducta mecánica en lugar de inteligente.

2. *Encontrar el sentido de mensajes contradictorios o ambiguos.* Se entienden muchas declaraciones que parecen ambiguas o contradictorias debido a nuestro conocimiento y experiencia.

3. *Reconocer la importancia relativa de los diferentes elementos de una situación.* Aunque recibimos cada día una gran cantidad de información le asignamos diferentes niveles de importancia a los diversos acontecimientos.

4. *Encontrar semejanzas entre las situaciones, a pesar de las diferencias que puede haber entre ellas.* Al reconocer semejanzas en lo que se ha aprendido en el pasado se pueden basar las acciones futuras. "(4)

Algunas definiciones que se han dado a la Inteligencia Artificial son las siguientes:

> " La Inteligencia Artificial es la parte de la informática que intenta hacer a las computadoras más inteligentes. Más específicamente la inteligencia artificial se compone de un conjunto de técnicas que permiten a las computadoras imitar a la mente humana." (5)

Las investigaciones en IA intentan descubrir la forma de hacer pensar y razonar a una computadora como persona, sin embargo no es posible duplicar perfectamente la increíble complejidad del cerebro humano La meta de la IA es conseguir que las computadoras sean más potentes y útiles, haciéndolas imitar las funciones del cerebro Las funciones estándar de proceso de datos no sirven para ciertas clases de problemas, que se pueden tratar con la IA

> " La Inteligencia Artificial es el estudio de cómo hacer que las computadoras hagan cosas que por el momento las personas realizan de una forma más perfecta.
>
> Elaine Rich, Artificial Intelligence " (6)

Las tareas que las computadoras pueden realizar mejor que las personas son actividades mecánicas En los últimos cuarenta años se ha demostrado que las computadoras pueden superar al hombre en actividades mentales Sin embargo hay muchas cosas que las personas pueden realizar mucho mejor que las computadoras, tradicionalmente, todas aquellas tareas que requieren inteligencia, como entender la información, dar sentido a lo que se ve y escucha, y extraer nuevas ideas

> " La Inteligencia Artificial es la parte de la informática relacionada con el diseño de sistemas de computadoras inteligentes, es decir, sistemas que exhiben las características que asociamos con la inteligencia en la conducta humana.
>
> Avron Barr y Edward A. Feigenbaum,
> The Handbook of Artificial Intelligence " (7)

El objetivo de la IA, de acuerdo con Barr y Feigenbaum, es desarrollar computadoras inteligentes

Uno de los propósitos de la IA, es mejorar la comprensión y capacidades mentales del ser humano La IA busca continuamente analogías entre la conducta humana y los programas de computadora

Cuando hay alguna actividad que se realiza de una forma natural sin tener que pensar sobre ella, se tendra gran dificultad en describir exactamente cómo se hizo, por lo que las habilidades que son naturales en las personas, son las más difíciles de programar

El campo que investiga estos detalles de los mecanismos de la inteligencia humana se conoce como **Ciencia Cognitiva** La investigación llevada a cabo por los científicos cognitivos explicando cómo trabaja ésta es de gran ayuda a los investigadores de la IA que están tratando de simular esa inteligencia en una computadora. Los científicos cognitivos, al determinar los procesos que producen inteligencia humana en una situación dada, pueden definir los procedimientos que pueden ser programados en una computadora para intentar simular esa conducta Esta técnica de IA se llama **modelación o simulación**

> " La Inteligencia Artificial es la rama de la informática, que tiene como métodos para procesar información la representación del conocimiento usando símbolos en lugar de números y la heurística o reglas basadas en la experiencia
>
> Bruce G. Buchanan, Encyclopedia Britannica."(8)

La **heurística** es una técnica que proporciona ayuda en la resolución de problemas que dependen de técnicas de experiencia para mejorar el desempeño La regla mas común en la resolución de problemas es if-then, a través de la cual un programa es activado por una estructura de control solamente cuando ciertas condiciones se encuentran, tal activación esta basada en el conocimiento acerca de un problema especifico ha resolver La heurística también toma la forma de una estructura sistemática de suposición en la cual la mejor opción es seleccionada por importancia, opciones que no son factibles son eliminadas, o la actual condición es evaluada contra los objetivos finales (criterio)

La heurística podrían también ser descrita como reglas de buen juicio La heurística es frecuentemente usada en problemas en los cuales los resultados no pueden ser garantizados de ser exitosos

Usando la heurística, no se tiene que plantear por completo un problema cada vez que se enfrenta a él Si ya se tiene una experiencia anterior, ésta sugiere la forma en la que se debe proceder La programación heurística incorpora esta aproximación basada en la experiencia al proceso de resolver problemas con computadoras usando Inteligencia Artificial

Los sistemas expertos se diseñan normalmente para ayudar a los expertos, no para reemplazarlos Se ha demostrado su utilidad en áreas distintas como la diagnosis médica, el análisis químico, la exploración geológica y la configuración de sistemas de cálculo

Procesamiento de lenguaje natural

Para muchos investigadores en IA, el procesamiento del lenguaje natural (conocido con frecuencia como **PLN**) es uno de los fines principales que la IA debe alcanzar porque permite a la computadora la entrada del lenguaje humano de forma directa El objetivo del procesamiento del lenguaje natural es permitir la comunicación hombre-máquina en un lenguaje humano "natural", en lugar de un lenguaje de programación El mayor obstáculo para alcanzar esta meta es el tamaño y la complejidad de los lenguajes humanos. Además, el problema de que la computadora sea consciente de la información contextual que pueda aparecer en cualquier situación que no sean las más simples

Percepción Visual

La investigación en visión por computadora tiene como objetivo el dotar a las computadoras con esta herramienta en el entendimiento de su entorno Su uso más común está en el área de la robótica y el procesamiento de imágenes Un sistema óptico forma una imagen de algún objeto conjunto de objetos tridimensionales La imagen, de dos dimensiones, es tomada y pasada a un formato que puede ser leído por la máquina Este es el propósito de un sistema de visión artificial, obtener de esta imagen la información necesaria y útil para la ejecución de una tarea. En el caso más simple, la información se refiere solamente a la posición y orientación de un objeto aislado, en otros casos, se deben reconocer los objetos y determinar sus relaciones espaciales La descripción debe ser adecuada para la aplicación particular. Esto es, las características visuales no relevantes deben ser destacadas, mientras que las relaciones necesarias entre partes de objetos deben ser deducidas de su proyección óptica

Robótica

Un **robot** es un dispositivo electromecánico que puede ser programado para que realice tareas manuales. No toda la robótica es considerada como parte de la IA. Un robot que sólo realice acciones que han sido previamente programadas es considerado como "tonto". Un robot "inteligente" incluye alguna forma de aparato sensorial, como puede ser una cámara, que le permite responder a cambios en su entorno en lugar de seguir instrucciones establecidas previamente

Aplicando a la robótica, la IA ayuda a que una computadora controle el movimiento usando un razonamiento espacial Para los robots industriales como los utilizados en el ensamblaje de automóviles, los problemas para la IA aparecen al tratar de suministrarles movimiento natural o preciso dentro de un conjunto de posiciones concretas Los robots autónomos tienen mayores problemas para desenvolverse en un mundo autónomo, con sus obstáculos, sucesos inesperados y cambios de ambiente.

Capítulo

3

Lenguajes de la Inteligencia Artificial

Es posible escribir cualquier programa en cualquier lenguaje, se ha ido probando que puede facilitarse considerablemente, la construcción de sistemas de IA mediante el uso de un lenguaje de programación, que proporcione un soporte para diversas estructuras usuales, tanto para datos como para control

Las características que debería poseer un buen lenguaje para construir casi cualquier tipo de sistema complejo incluyen

- Diversos tipos de datos para escribir las diversas clases de información
- La capacidad de descomponer el sistema en unidades pequeñas y comprensibles de forma que sea relativamente fácil hacer cambios en una parte del sistema
- Estructuras de control flexibles que faciliten tanto la recursión como la descomposición paralela en el sistema.
- La capacidad de comunicarse con el sistema interactivamente, tanto para el desarrollo del programa como para poder usar el sistema
- La capacidad de producir código eficiente de forma que el rendimiento del sistema sea aceptable

Ademas de estas características generales deseables en un sistema, existen otras importantes en el desarrollo de los sistemas de IA. Estas características incluyen

- Buenas facilidades para manipular listas, puesto que las listas son una estructura ampliamente usada en casi todos los programas de IA
- El establecimiento del tamaño de la estructura de una base de datos, o el tipo de un objeto sobre el cual se va a operar. En muchos programas de IA no es posible determinar tales cosas excepto cuando el programa puede esperar
- Facilidades de emparejamiento de modelos, tanto para identificar datos como para determinar el control
- Facilidades para realizar algún tipo de deducción automática y para almacenar una base de datos de asertos que proporcione la base para la deducción
- Facilidades para construir estructuras complejas de conocimiento, tales como armazones, de forma que puedan agruparse juntas las piezas de información relacionadas, y pueda accederse a ellas como a una unidad

- Mecanismos mediante los cuales el programador pueda proporcionar un conocimiento adicional que pueda usarse para centrar la atención del sistema en donde es probable que sea más provechoso
- Estructuras de control que faciliten la conducta dirigida por la meta (procesado descendente) además del procesado dirigido por los datos (ascendente)
- La capacidad de entremezclar procedimientos y estructuras declarativas de datos de la forma que mejor se adapte a una tarea concreta

Ningún lenguaje existente proporciona todas estas características

Clasificación de los Lenguajes

En la actualidad se usa como criterio de clasificación el modo de razonamiento sobre problemas, construir algoritmos y como están estructurados los datos y las instrucciones. De acuerdo con este criterio se suelen distinguir 4 tipos de lenguajes

• Lenguajes Procedimentales

Se escriben programas que indican a la computadora lo que debe hacer. La computadora recibe la secuencia de instrucciones paso-a-paso denominada **"algoritmo"**, que define claramente las acciones que se deben tomar para resolver un problema dado

Para resolver un problema informático convencional, los programadores analizan en primer lugar el problema, para determinar exactamente qué datos de salida deben obtenerse. Los programadores desarrollan entonces un algoritmo, este es un procedimiento incremental claramente definido que realiza las operaciones deseadas El programador toma entonces este algoritmo y lo convierte en una lista secuencial de instrucciones, sentencias o comandos, según estén definidos en el lenguaje de programación apropiado. Después, usando las herramientas adecuadas, se genera el código binario correspondiente a las instrucciones anteriores y se almacena en la memoria de la computadora Cuando se ejecute, este código resolverá el problema según lo especificado. El algoritmo se almacena en la memoria de la computadora junto con los datos que se van a tratar El algoritmo manipula los datos de entrada para producir los resultados deseados.

A continuación se describen algunos lenguajes procedimentales.

1. BASIC

El lenguaje **BASIC** cuyo nombre corresponde a las iniciales Beginners All-Purpose Symbolic Instruction Code (Código Simbólico de Instrucciones de tipo General para Principiantes), fue desarrollado en la Universidad de Dartmouth, Estados Unidos por Thomas E Kurtz y John Kemeny en la década de los sesentas Con la finalidad de iniciar a los estudiantes en la programación dada la facilidad de comunicación con la máquina y con la rapidez con la que se obtenían los resultados.

2. PASCAL

El lenguaje **PASCAL** es el resultado del trabajo iniciado en Working Group 2 1 of IFIT, con el objetivo de que fuera el sucesor del **ALGOL 60** Una primera versión del lenguaje PASCAL fue hecha en 1968 En 1970, apareció el primer compilador y fue introducido finalmente por Niklaus Wirth

PASCAL es un lenguaje que tiene las siguientes características

- Secuencialidad
- Estructuración. Las tres principales estructuras básicas son Secuenciales, Alternativas (IF-THEN-ELSE) e Interactivas o Repetitivas (el bucle WHILE).
- Fragmentación. Es dividir el programa en partes que realicen su objetivo de una manera más independiente del resto del programa En PASCAL esto se consigue por medio de las funciones y procedimientos

3. C

Dennis Ritchie inventó e implemento el primer lenguaje **C** en un DECPDP-11 que usaba el sistema operativo UNIX. El lenguaje C es el resultado de un proceso de desarrollo que comenzó con un viejo lenguaje llamado **BCPL** (desarrollado por Martin Richards). El BCPL influyó en un lenguaje llamado **B**, que invento Ken Thompson y que permitió el desarrollo del C en los setenta.

Algunas de las características de C son las siguientes

- C es conocido como un lenguaje de nivel medio, ya que combina elementos de lenguaje de alto nivel con la funcionalidad del ensamblador
- El código C es transportable. La portabilidad significa que se puede adaptar el software escrito para un tipo de computadora a otro tipo
- Permite la manipulación de bits, bytes y direcciones (que son los elementos básicos con los que trabaja una computadora) Esta posibilidad hace de C un lenguaje adecuado para la programación a nivel de sistema donde son comunes estas operaciones

• Lenguajes Funcionales

Church con el análisis funcional desarrolló un tipo de Lógica, situado entre la Lógica ordinaria y la Lógica combinatoria, que tiene como núcleo la profundización de funciones. Se fundamenta en el operador de conversión λ, y de ahí su denominación de "**cálculo-lambda**". Partiendo del supuesto de que cualquier función parcialmente recursiva se puede representar y además definir en el cálculo-λ y que por lo tanto se puede reducir a funciones λ-definibles y supone, además, que todas las funciones recursivas son computables ya que están construidas desde funciones computables específicas, Church propuso en el año 1935 la proposición reciproca a la hipótesis de Helbert " Toda función computable lo es por recurrencia, esto es, toda función computable es recursiva, es decir, toda función parcialmente recursiva es λ-definible y recíprocamente. " (9)

McCarty, uno de los pioneros de IA, demostro que la teoría y la técnica de funciones basada en el "**cálculo-λ**" puede servir para crear expresiones simbólicas que pueden ser implementadas y calculadas por una computadora como estructuras de datos Se dio cuenta que el enfoque funcional podía ser la base para representar nuestros conocimientos, y esta representación podía ser manipulada por las computadoras restringidas hasta entonces al procesamiento de números En estos principios se baso para la elaboración de un lenguaje simbólico de programación que sirvió para automatizar la solución de problemas usando una gran cantidad de conocimientos, se trata del lenguaje **LISP**

A continuación se describen brevemente algunos lenguajes funcionales

1. IPL

Uno de los primeros lenguajes diseñados específicamente para aplicaciones de IA fue el **IPL (Information Processing Language: Lenguaje de Procesado de Información)**. IPL se enfocaba principalmente en el procesado de listas El lenguaje mismo tenía más características de lenguaje máquina que de lenguaje de alto nivel. El IPL ya no se usa, ha sido completamente desplazado por los lenguajes de nivel más alto como el LISP Pero es interesante, puesto que fue uno de los primeros puntos de arranque para el desarrollo de lenguajes de IA

2. LISP

El **LISP** nació en los años 60 En esta época se contaba con pocos lenguajes de programación El LISP es el lenguaje que más se utiliza en Inteligencia Artificial, y aunque se han desarrollado ya otros lenguajes adecuados para ésta, debido a su estructura, LISP hace más fácil la programacion de ciertos conceptos de Inteligencia Artificial, por lo tanto, permite comprobar eficazmente ciertas ideas de búsqueda

Del LISP puede decirse que es eminentemente recurrente y que soporta todo tipo de procedimiento recurrente No es necesario definir tipos de variables (locales, globales), como en PASCAL Se dispone de toda flexibilidad de la noción de listas. Además, en vez de utilizar un inmenso procedimiento recurrente puede distribuirse mediante numerosas llamadas a funciones recurrentes, cada una de las cuales efectúa una parte del trabajo

3. LOGO

El **LOGO**, que fue originalmente creado para su utilización con grandes sistemas, es esencialmente un dialecto del LISP Pero, además, posee ciertas características que hacen de él un medio para aprender y reflexionar, desde el punto de vista informático, LOGO se diferencia de LISP en que aporta el uso de la tortuga gráfica y por la recurrencia infinita

LOGO permite desplazar un objeto por la pantalla, de tal manera que dicho objeto deje una estela visible, permitiendo así realizar dibujos. La novedad de LOGO no es que permita realizar dibujos, sino la forma de ejecutarlos El lenguaje de comandos es sumamente simple consiste en describir de forma natural los movimientos del lapicero. Sin embargo, la tortuga LOGO no es sólo una herramienta pedagógica, es una herramienta gráfica potente

En todo lenguaje recurrente, cuando se llama a los programas recurrentes, la computadora salvaguarda el entorno de la función En LOGO, el interpretador estima oportuno, en ciertos casos, no salvaguardar el entorno antes de ejecutar la llamada recurrente, permitiendo así la recurrencia infinita Esta recurrencia es declarada cuando la llamada recurrente es la última instrucción del procedimiento (TAIL RECURSION)

4. INTERLISP

Tiene todas las capacidades del LISP básico Además, tiene una colección de otras características que facilitan la construcción de programas grandes. Estas características incluyen:

- Diversos tipo de datos, además de listas, tales como matrices y cadenas de bits.
- Una pila espaghetti, donde se almacenan los contextos del programa
- Diversas herramientas para facilitar la programación

5. PLANNER

Es un lenguaje basado en LISP y diseñado para representar tanto el razonamiento tradicional hacia adelante como el razonamiento atras dirigido hacia la meta. Los programas en PLANNER consisten en 2 tipos de sentencias

- **Aserciones**, que verifican simplemente los hechos conocidos.
- **Teoremas**, que describen cómo pueden inferirse nuevos hechos a partir de los antiguos.

Existen tres tipos de teoremas que pueden aparecer en los programas PLANNER.

- **Teoremas consecuentes**, que describen razonamiento hacia atrás o dirigido por la meta
- **Teoremas antecedentes**, que describen razonamiento hacia adelante o dirigido por los datos.
- **Teoremas de borrado**, que eliminan asertos de la base de datos.

6. KRL

Es un lenguaje, basado en el INTERLISP, que facilita la representación del conocimiento en estructuras de marco (estructuras ranura-y-llenado). Su diseño fue motivado por las siguientes suposiciones sobre las representaciones del conocimiento y los programas que las usan:

- El conocimiento debería organizarse alrededor de entidades conceptuales con descripciones y procedimientos asociados
- Una descripción debe poder representar conocimiento parcial sobre una entidad y acomodar múltiples descriptores que puedan describir la entidad asociada desde puntos de vista diferentes.
- Un importante método de descripción es la comparación de una entidad prototipo con una entidad conocida, especificando las diferencias entre ellas
- El razonamiento esta dominado por un proceso de reconocimiento, en él se comparan los nuevos objetos y acontecimientos a conjuntos almacenados de prototipos esperados, y en donde las estrategias de razonamiento especializadas se indexan mediante dichos prototipos

- Un lenguaje de la representación del conocimiento debe proporcionar un conjunto flexible de herramientas tanto sobre estrategias de procesado como sobre representación de áreas específicas del conocimiento

• Lenguajes Declarativos

Turing había demostrado que la definición informal de algoritmo como computable por una máquina ideal (máquina de Turing) es idéntica a la función lambda-definible, es decir, que toda función recursiva es computable por una máquina de Turing

Turing era partidario de que las instrucciones de las computadoras estuvieran basadas en los operadores de la Lógica desde los cuales se pudieran construir operaciones numéricas más especializadas Es así como se crean los lenguajes declarativos de programación, que se basan en la investigación sobre la demostración automática de teoremas

Robinson, en los años 60, propuso el método de resolución que permitía hacer deducciones automáticas partiendo de fórmulas que previamente han sido convertidas a la forma clausal, es decir a un conjunto de sentencias en las que las condiciones son conjuntas (unidas por el operador de la alternativa) Creó las técnicas de demostración de teoremas basadas en el principio de resolución, en el que dadas dos proposiciones en forma de cláusulas se deduce una nueva como consecuencia de las dos anteriores " Una importante propiedad de la resolución es que ante un conjunto de cláusulas inconsistentes solo derivaría la cláusula vacía entendiendo por cláusulas inconsistentes aquellas que no representan simultáneamente expresiones verdaderas y por cláusula vacía la expresión lógica de falsedad. Otra propiedad es la integridad por la cual un teorema deducido de unas hipótesis, la Resolución demostraría la inconsistencia entre la negación del teorema deducido y las hipótesis originales, dando como resultado la cláusula vacía. " (10)

Algunos lenguejes declarativos son

1. PROLOG

Fue desarrollado en **Francia** en **1973,** es un lenguaje de programación ampliamente usado por los investigadores de IA en Europa Su uso también está muy extendido en Japón, ya que ha sido adoptado como el lenguaje oficial de programación de IA para el Proyecto de la Quinta Generación.

PROLOG se basa en la utilización de la lógica para resolver problemas, se le conoce como un "**sistema que demuestra teoremas**" y que usa una técnica de la **lógica formal** llamada "**Cálculo de predicados**" para demostrar la verdad de las proposiciones a partir de un conjunto de axiomas. PROLOG es un lenguaje de programación mucho más sencillo que LISP. En los Estados Unidos se usa a menudo PROLOG como un complemento al LISP, añadiendo características que puedan ayudar a los programas LISP a resolver ciertos tipos de problemas. En el siguiente capítulo se describirá ampliamente al lenguaje PROLOG.

2. OPS

El OPS ("Official Production System") fue creado por C L Forgy de la Universidad Carnegie Mellon en la mitad de la década de los setenta

El OPS es un lenguaje con un sistema de representación del conocimiento en forma de reglas de producción, con un motor de inferencia de orden 1, empleando la comparación de patrones, con encadenamiento hacia delante

La existencia de una memoria de producción facilita los procesos de mantenimiento de la base de condiciones en un Sistema Experto y de aprendizaje de los programas escritos en OPS

Por otra parte, la memoria de trabajo es similar a la base de datos y facilita la separación de datos y programas por una parte y unifica ficheros, memoria y datos existentes en otros lenguajes

La versión más conocida es la OPS5 de digital que esta disponible en la gama de VAX.

• Lenguajes Orientados a Objetos

En la actualidad se considera un tipo de programación muy acorde con las necesidades de la IA La razón es que permite una simulación del comportamiento más real Se trata de un tipo de programación que se acerca más a nuestra forma natural de pensar al ser más estructurado, modular y racional que los lenguajes imperativos y declarativos. Con la programación orientada a objetos los programas tienen menos líneas de código, menos sentencias de bifurcación, y módulos que son más comprensibles porque refleja de una forma clara la relación existente entre cada concepto a desarrollar y cada objeto que interviene en dicho desarrollo

Las propiedades fundamentales de un lenguaje orientado a objetos son **encapsulamiento**, **jerarquía-herencia**, **polimorfismo** y **enlace dinámico** Se realizan estas características en programas cuyos módulos encapsulan datos con sus correspondientes métodos que definen un objeto A través de ellos se construyen clases en la que los objetos están jerárquicamente organizados y enlazados por metodos virtuales Supone un grado de abstracción superior en el desarrollo de los lenguajes de programación que históricamente comienza con la codificación de los datos y algoritmos en el lenguaje de máquina y culmina en la actualidad en objetos que contienen datos y procedimientos en una sola unidad A pesar de su grado de abstracción funcionan de una manera secuencial.

Algunos lenguajes se describen a continuación.

1. SMALLTALK

El más conocido de todos los lenguajes orientados a los objetos es el SMALLTALK.

El desarrollo del lenguaje SMALLTALK comenzó en Palo Alto Research Center de Xerox, por Alan Kay a partir del lenguaje LISP, dando origen a la versión Smalltalk 72 Poco después influenciado por el lenguaje Simula 67 se desarrollo la versión Smalltalk 76. Hasta la versión Smalltalk 80, Xerox decide su difusión y comercialización.

SMALLTALK además de las características de los lenguajes orientados a los objetos variables, objetos, clases, métodos y mensajes, es muy modular y flexible

El concepto del SMALLTALK lo hace especialmente útil en las tareas de prototipo.

2. EIFFEL

Es un lenguaje de programación avanzado orientado a objetos creado en **1985** por **Bertrand Meyer** Después le dio el nombre de Gustave Eiffel (ingeniero que diseñó la Torre Eiffel), Eiffel fue desarrollado por Interactive Software Engineering (ISE Ingeniería de Software Interactiva) de Goleta, California Este es un lenguaje de programación orientado a objetos que hace énfasis en el diseño y desarrollo de software de calidad y reusable Eiffel no es un super conjunto o extensión de algún otro lenguaje Hace interfase con otros lenguajes tales como C y C++. Eiffel fomenta el concepto de "Design by Contract" para mejorar las correcciones de software Eiffel es un lenguaje orientado a objetos basado en los siguientes conceptos

- Herencia múltiple incluyendo redefinición, no definición, renombrar, selección.
- Aserciones para escribir, depurar y documentar software automáticamente.
- Desarrollo uniforme adecuar todo el ciclo de vida desde el análisis y diseño de alto nivel hasta la implantación y mantenimiento.
- Clases, sirviendo como las bases tanto para la estructura del módulo y para el tipo de sistema
- Herencia para clasificación, tipos, subtipos y reuso de software.
- Manejo de excepción para recuperar estados no planeados.
- Tipos estáticos y enlaces dinámicos
- Eiffel tiene un elegante diseño y estilo de programación, y es fácil de aprender

3. DYLAN

Es un nuevo lenguaje dinámico orientado a objetos (OODL) que esta siendo desarrollado por Apple Este esfuerzo en el desarrollo del lenguaje tiene como objetivo el desarrollar una herramienta práctica para realizar aplicaciones comerciales El propósito es combinar las mejores cualidades de lenguajes estáticos (pequeños, programas rápidos) con las mejores cualidades de lenguajes dinámicos (rápido desarrollo, código fácil de leer, escribir y mantener). Dylan tiene las siguientes caracteristicas que lo distinguen de C++ que lo hacen mas poderoso y flexible

- Manejo de memoria automática
- Claro, sintaxis consistente
- Modelo completo y consistentemente orientado a objetos
- Dinámico así como chequeo de tipo estático.
- Soporte para el incremento de la compilación
- Funciones first-class y clases

Capítulo

4

Representación del Conocimiento

La representación de la realidad es una de las tareas más complejas con las que se ha enfrentado el hombre desde sus orígenes, y muchos problemas no han tenido solución hasta que no se han desarrollado los sistemas de representación adecuados para los mismos.

La necesidad de profundizar en el "funcionamiento" de la realidad, ha hecho que el hombre desarrolle un lenguaje matemático con el fin de construir modelos (modelos matemáticos), que es completamente "artificial" La lógica también es una forma de representar la realidad, que permite determinar la veracidad o falsedad de las conclusiones o resultados en base de una serie de hechos o axiomas de partida

Existen dos características de los sistemas de representación del conocimiento, con las que hay que llegar a una solución

- Poder expresivo facilidad de descripción Cuando más simbólico es un sistema, más potente es
- Eficacia del cálculo cuánto más simbólico es un sistema de representación menor es su eficacia de cálculo

Durante mucho tiempo la psicología cognitiva ha investigado las formas de representar el conocimiento que más se aproximen al del cerebro humano Basándose es estos estudios, la ingeniería del conocimiento, ha creado diversos sistemas de representación del conocimiento, implantados sobre una computadora, que actúan según los modelos teóricos elaborados por la psicología cognitiva.

Existen múltiples formas de representar el conocimiento en una computadora, y cada una de ellas representa alguna ventaja sobre el resto El deseo de alcanzar soluciones permitió la aparición de las llamadas representaciones híbridas, que intentan incorporar las ventajas de varios sistemas.

Para resolver los problemas complejos en la Inteligencia Artificial, se necesita tanto una gran cantidad de conocimiento como algunos mecanismos para manipular dicho conocimiento a fin de crear soluciones a problemas

Los programas de IA han explotado diversas formas de representación del conocimiento (hechos). Pero antes de describirlos se debe considerar lo siguiente:

- **Hechos** verdades en algún mundo relevante Estas son las cosas que queremos representar
- Representaciones de los hechos en algún formalismo elegido. Estas son las cosas que realmente podremos manipular

Representación Lógica

La **Representación Lógica** es una forma de representar el conocimiento por medio de predicados y relaciones lógicas entre predicados enlazados por operadores lógicos.

La **lógica matemática** es aquella orientación de la Lógica que estudia el razonamiento humano con métodos algebraicos Métodos algebraicos que se concretan en el uso de símbolos inequívocos, definiciones precisas y axiomas que facilitan el razonamiento y las demostraciones rigurosas de modo puramente mecánico

La **lógica formal** se divide en tres grandes apartados **Formal**, **Metalógica** y **Metodología** La lógica formal, también denominada de primer orden porque solamente cuantifica variables individuales, en la que se incluye la lógica proposicional y de predicados o términos La **lógica proposicional** estudia aquellos razonamientos cuya validez depende de las relaciones que se dan entre las proposiciones tomadas en bloque, es decir, sin analizar su estructura interna Al estudiar el razonamiento se tienen en cuenta los constituyentes de una proposición, es decir, su estructura interna, pasamos al **cálculo de predicados**, también denominado **cuantificacional** porque se introducen los cauntificadores 'todos' y 'algunos'

• Lógica Proposicional

La lógica más simple es la de las proposiciones. La proposición se enuncia a partir de un hecho que puede ser cierto o falso Cuandola proposición es cierta se le llama **aserción.**

Una proposición es cierta, si está de acuerdo con los hechos que se desean describir En caso contrario, es falsa

Para obtener proposiciones se dispone de operaciones sobre las proposiciones. Con el fin de describirlas, se utilizan lo que se denomina una tabla de verdad. Los principales operadores son:

- NO
- Y
- O
- IMPLICA
- SI ENTONCES SINO

Las reglas que permiten deducir la veracidad de una conclusión de la veracidad de ciertas proposiciones se denomina **regla de inferencia** Por ejemplo, tal regla es

(p cierto) y (q cierto) implica que (p y q cierto)

los dos primeros miembros son denominados **premisas** y el último **conclusión**.

La lógica de las proposiciones esta limitada en la medida en que trata proposiciones hechas que permiten pocas combinaciones.

- **Lógica de Predicados**

Se considera la siguiente proposición

"La caja está sobre la mesa"

Se trata de una proposición en la que aparecen dos objetos o variables: "mesa" y "caja" Esta proposición aparece como relación entre estas dos variables.

En la siguiente proposición

"Todo hombre es mortal"

la variable "hombre" está asociada a la calidad de mortal El atributo de esta proposición en este caso "mortal", se denomina **predicado**

La potencia de la lógica de predicados se debe a la utilización de variables en las proposiciones

La proposición lógica:

SOBRE(caja,x) Y SOBRE(x,mesa) implica SOBRE(caja,mesa)

La letra x designa cualquier variable, reemplazando x por un objeto cualquiera, se obtiene una proposición lógica

La utilización de las variables permite representar de forma extremadamente concisa una gran cantidad de conocimientos.

La proposición

"Todos los hombres son mortales"

puede traducirse con la proposición lógica

HOMBRE(x) implica MORTAL(x)

Si se sustituye la variable x por un nombre, se efectua lo que se denomina **una instanciación**

• Método de Resolución

Los programas de IA utilizan un método denominado Método de Resolución. En primer lugar se definen los siguientes términos

a) Un **átomo** es una proposición que no se puede descomponer en otras proposiciones.
Ejemplo:
(p y q) no es un átomo
p es un átomo

b) Un **literal** es tanto un átomo como la negación de un átomo
Ejemplo.
no p es un literal

c) **Cláusula** es una serie de literales unidas por la conjunción "o"
Ejemplo
p o (no q) o (no r) es una cláusula

El procedimiento de resolución es un proceso iterativo simple en el cual, en cada paso, se comparan (resuelven) dos cláusulas llamadas cláusulas padres, produciendo una nueva cláusula que se ha inferido de ellas La nueva cláusula representa la forma en que las dos cláusulas padres interaccionan entre si. Suponiendo que hay dos cláusulas en el sistema

invierno o verano
no invierno o frío

Siempre deberá ser cierta una de las aserciones invierno o no invierno. Si invierno es cierto, entonces frío debe ser cierto para garantizar la verdad de la segunda cláusula. Si no invierno es verdad, entonces verano debe ser cierto para garantizar la verdad de la primera cláusula. Así pues, a partir de esas dos cláusulas se puede deducir

verano o frío

Esta es la deducción que hará el procedimiento de resolución La resolución opera tomando dos cláusulas tales que cada una contenga el mismo literal, en este ejemplo, invierno El literal debe estar en forma positiva en una cláusula y en forma negativa en la otra El resolvente se obtiene combinando todos los literales de las dos cláusulas padres excepto aquellos que se cancelan

Si la cláusula producida es la cláusula vacía, es que se ha encontrado una contradicción

Partiendo del principio de resolución para deducir automáticamente los resultados de premisas y reglas de inferencia, bastará con poner las premisas y las reglas de inferencia en forma de cláusulas para lograr las conclusiones deseadas

Representación Procedural

Las técnicas de lógica de predicados son útiles para resolver problemas en una amplia variedad de dominios Desafortunadamente, en muchos otros dominios interesantes, la lógica de predicados no proporciona una buena forma de representar y manipular la información importante.

Gran parte del razonamiento de las personas supone la manipulación de un conjunto de creencias Cada una de estas creencias está apoyada por alguna evidencia y puede reforzarse mediante alguna motivación personal que mantenga dicha creencia. No es extraño que un conjunto de creencias, llamado sistema de creencias, sea tanto incompleto como inconsistente

Un sistema raramente tiene a su disposición toda la información que le sería útil Hasta el momento, se ha supuesto que un hecho es cierto o que no es cierto No se ha considerado la posibilidad de que se puede saber que algo es "probablemente cierto" Sin embargo, hay situaciones en las cuales tal procedimiento es importante Con frecuencia, cuando falta información, pueden hacerse algunas suposiciones que se conocen como **razonamiento por defecto**.

• Teoría de la Probabilidad

Es una rama de las matemáticas que proporciona modelos sobre los sucesos aleatorios Se denomina **aleatorio** aquel acontecimiento o hecho del que no se puede predecir de una manera rigurosa su resultado antes de llevarlo a cabo Se puede repetir cuantas veces se desee el experimento para ver el resultado y sin embargo no se puede pronosticar con certeza

La formulación de la teoría de la probabilidad parte del supuesto de que todos los sucesos tienen la misma probabilidad, es decir, de la previa asignación de la probabilidad que debe tener cualquier suceso y que por ello se llama **probabilidad a priori** Con la **probabilidad a posteriori** se ha comprobado empíricamente que al repetir un experimento un número suficiente de veces la frecuencia absoluta con la que ocurre un suceso se estabiliza

Las dos teorías han servido de fundamento para la construcción de una teoría axiomática de la probabilidad que ha tenido un amplio desarrollo. La **teoría de la probabilidad condicional** nos dice: la probabilidad de que suceda A una vez que haya ocurrido B Se calcula de acuerdo a la siguiente fórmula

$$P(A/B) = P(B \cap A)/P(B)$$

Una consecuencia de la teoría anterior es la fórmula del **teorema de Bayes** por medio del cual se calcula la probabilidad de un suceso en un espacio muestral (conjunto de todos los casos posibles de un experimento aleatorio) en el que la probabilidad de un suceso condiciona el resultado de otro suceso siendo ambos incompatibles

- **Teoría de la Evidencia**

Se basa en el modelo propuesto por **Dempster** y **Shafer** Tiene como punto de partida la definición de un marco de referencia en el que se representan todas las informaciones o hipótesis que poseen acerca de un **dominio del universo** Matemáticamente lo podemos definir como un **conjunto U** en el que se especifican todos los subconjuntos posibles incluidos el conjunto vacio y que podemos representar por P(U).

> " La información que se posee sobre cada uno de los subconjuntos permite definir la unidad de evidencia como aquella asignación básica de probabilidad sobre el conjunto P(U) y expresa la evidencia que se posee sobre la ocurrencia de cada uno de los subconjuntos de U. Y si le damos el valor numérico 1, a la probabilidad del conjunto vacio le daremos 0. " (11)

Lo anterior permite definir los puntos focales como aquellos elementos del conjunto A cuya asignación básica de probabilidad no es nula, es decir, como aquellos elementos de los que se dispone de información ya sea a través de datos estadísticos, información de los expertos, etc Desde ellos se puede calcular la evidencia de los puntos no focales por medio de una cota inferior denominada **plausibilidad** y una cota inferior denominada **creencia**.

La creencia y la plausibilidad están relacionadas según la siguiente fórmula.

$$Pl(A) = 1 - Cr(\neg A)$$

La interpretación que dio Dempster del significado de las funciones de creencia y plausibilidad es que el intervalo [Cr(A), Pl(A)] representa la cota de probabilidad de A y que por medio de él se puede medir la incertidumbre y la ignorancia Este intervalo se distribuye en 4 valores: ignorancia, información máxima, certidumbre e incertidumbre

- **Teoría de la Posibilidad**

Se puede definir como aquella rama de la lógica matemática que intenta formalizar los conceptos y razonamientos imprecisos e inciertos partiendo de la teoría de los conjuntos vagos o borrosos. Muy recientemente se ha empezado a reconocer la relevancia e importancia de la teoría de las posibilidades que hasta ahora se encontraba en el campo de la investigación pura. Los factores que han hecho posible su reconocimiento son:

- Las limitaciones de la teoría de la probabilidad y la teoría de la evidencia en la que se procesan conocimientos y reglas imprecisas como si fueran precisas

- La aspiración de la IA para construir máquinas tan inteligentes como el hombre.

- La reciente inundación en el mercado por parte de los japoneses de aparatos fiables, baratos y efectivos basados en los principios de la lógica de lo vago Hay que mencionar la aspiración a construir una computadora cuyo hardware trabaje según los principios de la lógica difusa

Tiene su comienzo en las investigaciones de **Zadeh** sobre la manera de controlar sistemas complejos para los que no servían los modelos clásicos basados en ecuaciones diferenciales Ello dio origen a la teoría de subconjuntos difusos o borrosos y más tarde a la lógica borrosa o difusa Su punto de referencia fue la teoría ordinaria de conjuntos que pretendió ampliar Por ello, primeramente se consideran las nociones básicas y elementales sobre esta rama de la Matemática como punto de unión con los subconjuntos difusos para terminar finalmente en la lógica de lo vago

1) Teoría Clásica de Conjuntos

Un **conjunto** se puede definir como una colección de cosas o sujetos diferenciados considerados como una sola entidad Cada uno de los objetos o miembros del conjunto de denominan **elementos** Los conjuntos se simbolizan con letras mayúsculas, los elementos con letras minúsculas. Se utiliza la notación $x \in A$ y $x \notin A$ para designar que x es o no es un elemento que pertenece al conjunto A

El conjunto que agrupa todos los subconjuntos, el vacio y a sí mismo, se denomina **conjunto universal**, **universo de discurso** o **referencial** y se simboliza por **U** Al conjunto que no posee ningún elemento se le llama **conjunto vacio** y se simboliza por ∅, el que tiene un sólo miembro se llama **conjunto de un solo elemento** o **conjunto unitario**

Para determinar un conjunto se puede usar la notación en lista en la que cada uno de los elementos del conjunto se encierra en dos llaves Se llama **cardinalidad de un conjunto** N al número de elementos que contiene.

Resulta útil representar los conjuntos a través de diagramas denominados **diagramas de Venn** y que son una ayuda gráfica para comprobar la validez de ciertos teoremas de la teoría de conjuntos

2) Subconjuntos Difusos

De acuerdo a la teoría clásica de conjuntos, si x es un elemento de un conjunto universal U, y A un subconjunto $A \subset U$, solamente existen dos posibilidades: un elemento pertenece o no pertenece al subconjunto A

$$x \in A : \mu_A(x) = 1$$
$$x \notin A : \mu_A(x) = 0$$

La idea es que la función de pertenencia o función característica $\mu_A(x)$ solamente asigna valores discretos en el conjunto de evaluación {1,0} Y aquí es donde aparece una diferencia fundamental ya que la función de pertenencia en los conjuntos borrosos o difusos asigna valores continuos en el intervalo cerrado, es decir, varía gradualmente de 0 a 1 [0,1] Este intervalo es denominado **conjunto de evaluación** en el que **1** significaría **pertenencia entera** o **total**, y **0 no pertenencia absoluta**

Sea U el universo de discurso o dominio, y A un subconjunto borroso, la función de pertenecía asigna diferentes grados de distribución en el subconjunto A, que por definición se supone borroso, de los elementos de U Por ello en la teoría de los subconjuntos borrosos sus elementos no se simbolizan con variables discretas sino continuas.

3) Lógica Difusa

Si la lógica clásica se define como la teoría del razonamiento formalmente correcto, la **lógica difusa** o **vaga** se puede definir como la teoría del razonamiento incierto, impreciso, o aproximado Cuando se representa el conocimiento o se razona con información imprecisa y/o incierta es necesario usar la lógica difusa o borrosa.

Está proporciona el sistema formal que permite la representación y procesamiento de los conceptos y razonamientos que las personas en la vida cotidiana expresan por medio del lenguaje natural En la lógica difusa

- Todo es cuestión de grado
- El conocimiento se interpreta como un conjunto de acotaciones borrosas elásticas y el razonamiento como una propagación de la mismas
- La lógica difusa supone que pueden existir muchos valores de verdad y además el valor verdadero de cada uno de ellos es vago en si mismo, esto es, matices del predicado metalingüístico verdad
- Los predicados son imprecisos, hay una amplia gama de modificadores y una enorme variedad de cuantificadores borrosos.

Representaciones Estructuradas del Conocimiento

Un sistema para representar el conocimiento estructurado y complejo debería poseer las cuatro propiedades siguientes

- **Adecuación representacional** La capacidad de representar todas las clases de conocimiento que se necesitan
- **Adecuación inferencial**· La capacidad de manipular las estructuras representativas de forma que puedan derivarse nuevas estructuras correspondientes a conocimiento nuevo a partir de las anteriores.
- **Eficiencia inferencial**. La capacidad de incorporar a la estructura de conocimiento información adicional que pueda usarse para centrar la atención de los mecanismos de inferencia

- **Eficiencia adquisicional** La capacidad de adquirir nueva información con facilidad

Las estructuras que a continuación se definen pueden usarse para representar el conocimiento en cualquier dominio de problemas, y pueden hacerlo independientemente de la forma en que funcione el conocimiento dentro del sistema. Por otra parte, históricamente la mayoría de ellos se desarrollaron para sistemas que intentaban comprender las entradas en lenguaje natural

Cada una de ellas es una estructura de datos donde puede almacenarse el conocimiento sobre dominios concretos de problemas Muchas de estas estructuras se componen de estructuras más pequeñas Por lo tanto, el término estructura de conocimiento significa unas veces una base de datos completa de información sobre un dominio concreto, y otras, subestructuras de una estructura mayor Estas subestructuras corresponderán usualmente a cosas tales como objetos o sucesos que pertenezcan al dominio

Una de las razones por las que las estructuras de conocimiento son importantes, es que proporcionan una forma de representar la información sobre modelos que ocurren usualmente.

- **Redes Semánticas**

Fueron creadas de manera independiente por **Quilian** para simular la memoria humana y por **Raphael** en el año de **1968** Una **red semántica** se puede definir como un grafo dirigido cuyos nodos y arcos representan y describen a través de relaciones binarias la conexión que existe entre entidades de un dominio del universo

A través de los nodos de la red semántica se representan ideas y entidades discretas Son típicos los siguientes

- **Conceptos**: sirven para especificar las entidades o ideas sobre objetos.
- **Sucesos** sirven para describir los eventos o acciones que se producen entre las entidades del dominio
- **Propiedades**. sirven para representar las características y estados de diferentes entidades, ya sean vistas desde una perspectiva estática o dinámica.

A través de los arcos de una red semántica se pueden representar los siguientes tipos de conocimiento estructurado generalmente usados para describir la relación entre los nodos.

- **Pertenencia** a un conjunto que podemos expresar con el término "**miembro_de**". Relaciona una entidad con su clase y sirve para realizar clasificaciones.
- **Generalización** por medio de la cual se conecta una entidad con otra más general, es decir, la relación de una entidad con una clase Se puede representar por "**es_un**".
- **Agregación** en la que se relaciona una entidad con sus componentes. Se puede representar con "**parte_de**"

Las redes semánticas se diseñaron originalmente como una forma de representar los significados de las palabras inglesas Una de las primeras maneras en que se usaron las redes semánticas era para encontrar relaciones entre los objetos activando cada par de nodos y viendo donde esta activación producia un encuentro, este proceso se llamó búsqueda de intersección

Las aplicaciones más recientes de las redes semánticas usan procedimientos de búsqueda más dirigidos a responder cuestiones específicas Aunque en principio no hay restricciones en la forma que puede representarse información en las redes, estos procedimientos de búsqueda sólo pueden ser efectivos si hay consistencia en lo que significa cada nodo y cada enlace.

- **Dependencia Conceptual**

Llamada a veces **DC**, es una teoría de cómo representar el significado de frases del lenguaje natural de forma que

- Facilite sacar conclusiones de las frases
- Sea independiente del lenguaje en que estaban las frases originalmente

La representación DC de una frase no se construye a partir de primitivas que correspondan a las palabras usadas en la frase, sino de primitivas conceptuales que pueden combinarse para formar significados de palabras en cualquier lenguaje concreto Se ha usado en diversos programas que leen y comprenden textos en lenguaje natural. A diferencia de las redes semánticas, que sólo proporcionan una estructura en la cual pueden colocarse los nodos que representan la información en cada nivel, la dependencia conceptual proporciona, no solamente una estructura dónde representar el conocimiento, sino también un conjunto específico de elementos de construcción a partir de los cuales se edifican las representaciones Un conjunto así contiene un grupo de acciones primitivas, a partir de las cuales pueden construirse acciones de un nivel más elevado correspondiendo a palabras castellanas.

Las ventajas de usar la dependencia conceptual son

- Se necesitan menos reglas de inferencia de las que se requerirían si no se hubiese descompuesto el conocimiento en primitivas
- Muchas inferencias ya están contenidas en la representación misma
- La estructura inicial que se ha construido para representar la información contenida en una frase tendrá huecos que deberán llenarse Estos agujeros pueden servir como focalizador de atención para el programa que debe entender las frases
- Usando las primitivas facilitamos describir las reglas de inferencia mediante las cuales puede manipularse el conocimiento

- **Marcos (Frames)**

Las personas no analizan nuevas situaciones empezando desde cero, en vez de ello tienen disponible en la memoria una colección de estructuras que representan sus experiencias anteriores con objetos, lugares, situaciones y gente Para analizar una nueva experiencia, evocan las estructuras almacenadas apropiadas y, a continuación, las llenan con los detalles del acontecimiento actual. Un mecanismo general diseñado para la representación por computadora de tal conocimiento común es el **marco**

Típicamente, un marco describe una clase de objetos, tales como SILLA o HABITACION Consiste en una colección de ranuras que describen el aspecto de los objetos Estas ranuras se llenan mediante otros marcos que describen otros objetos Asociado con cada ranura puede haber un conjunto de condiciones que deben cumplir las entidades que vayan a llenarla Cada ranura puede también llenarse con un valor por defecto, de forma que, en ausencia de información específica sobre lo contrario, pueda suponer las cosas

También puede asociarse información procedural con cada ranura concreta. Por ejemplo, suele ser útil describir lo que debería hacerse siempre que se llena una ranura (llamado a menudo un **procedimiento si-añadido**) o como puede calcularse un valor para una nueva ranura si se requiere (llamado generalmente **procedimiento si-necesitado** o **para-establecer**) A tales procedimientos se les conoce como demonios, que conceptualmente es un proceso que vigila si alguna condición se hace cierta y, entonces, activa un proceso asociado Por supuesto, normalmente los demonios no se implementan como procesos a la espera de eventos, sino que los eventos que esta esperando son los que los activan cuando es apropiado

La mayoría de los sistemas útiles deben explotar más que un marco Los marcos relacionados pueden agruparse juntos para formar un **sistema de marcos**.

- **Guiones (Scripts)**

Lo define **Schrank** como una estructura de datos que representan una situación típica. " Los guiones son conjuntos prefabricados de expectativas, inferencias y conocimientos que se aplican en las situaciones comunes, como un plan de acción con los detalles en blanco." (12). Un guión esta formado por un conjunto de ranuras Asociado a cada ranura puede estar alguna información sobre que tipo de valores puede contener, así como un valor por defecto que puede usarse si no se dispone de ninguna otra información

Aunque poseen la misma estructura que los marcos en ellos resaltan no tanto las propiedades de los objetos sino la secuencia de sucesos que ocurren en situaciones típicas.

Los componentes importantes de un guión son.

- Condiciones de entrada Condiciones que, en general, deben satisfacer antes de que puedan ocurrir los sucesos escritos en el guión.
- Resultados. Condiciones que en general, deberán ser ciertas después de que ocurran los eventos que se describen en el guión
- Material Ranuras que representan los objetos que están involucrados en los eventos descritos en el guión La presencia de esos objetos puede inferirse incluso sino se mencionan explícitamente.
- Papeles Las ranuras que representan a las personas involucradas en los eventos descritos en el guión La presencia de estas personas puede inferirse aunque no se les mencione explícitamente Si se mencionan individuos específicos, pueden insertarse en las ranuras apropiadas
- Lugar Este guión representa una variación específica de un modelo más general. Diversas ubicaciones del mismo guión pueden compartir muchos componentes, aunque no todos.

- Escenas Las secuencias de eventos que ocurren. Los eventos se representan mediante el formalismo de la dependencia conceptual.

Los guiones son útiles porque en el mundo real hay modelos de los eventos Estos modelos surgen a causa de las relaciones causales entre dichos eventos Los eventos escritos en un guión forman una cadena causal gigante El principio de la cadena es el conjunto de condiciones de entrada, que permite que ocurran los primeros eventos del guión. El final de la cadena es el conjunto de resultados, que puede permitir eventos posteriores o secuencias de eventos (descritos posiblemente en otros guiones)

Pies de página

(1) A Fondo Inteligencia Artificial; MISHKOFF Henry , pág. 9
(2) A Fondo Inteligencia Artificial, MISHKOFF Henry , pág 36
(3) A Fondo Sistemas Expertos, FRENZEL, Louis E Jr , pág 14
(4) A Fondo Inteligencia Artificial, MISHKOFF Henry , pág. 15-16
(5) Fundamentos de Inteligencia Artificial, ALVAREZ, Munárriz Luis, pág. 19
(6) A Fondo Inteligencia Artificial, MISHKOFF Henry ; pág. 12
(7) A Fondo Inteligencia Artificial, MISHKOFF Henry , pág. 14
(8) A Fondo Sistemas Expertos, FRENZEL, Louis E Jr ; pag. 14
(9) Fundamentos de Inteligencia Artificial, ALVAREZ, Munárriz Luis, pág 31
(10) Fundamentos de Inteligencia Artificial, ALVAREZ, Munárriz Luis, pág. 33
(11) Fundamentos de Inteligencia Artificial, ALVAREZ, Munárriz Luis, pág 139
(12) Fundamentos de Inteligencia Artificial; ALVAREZ, Munárriz Luis, pág 156.

Bibliografía

Fundamentos de Inteligencia Artificial
ALVAREZ, Munárriz Luis
Universidad de Murcia
Madrid, 1994
366 pp.

Inteligencia Artificial
AUBERT, Jean Pascal y SCHOMBERG, Richard.
Ed Paraninfo
Madrid, 1986
132 pp.

Biblioteca de Informática.
Vol 7 México, 1992.
Ed Limusa Grupo Noriega.

A fondo Sistemas Expertos
FRENZEL, Louis E Jr
Ed Anaya Multimedia
México, 1987
213 pp

A fondo Inteligencia Artificial
MISHKOFF, Henry
Ed Anaya Multimedia.
Madrid, 1988.
270 pp

Inteligencia Artificial
RICH, Elaine
Ed Mc Graw-Hill
España, 1997
2da. edición
703 pp

Sistemas Expertos· Una metodología de programación
SANCHEZ y Beltrán, Juan Pablo
Ed. Macrobit.
Estados Unidos, 1990
261 pp

Utilización de C en Inteligencia Artificial
SCHILDT, Herbert
Ed Mc Graw-Hill.
México, 1990
340 pp

Inteligencia Artificial en México
U T M Universidad Tecnológica de la Mixteca
Ed Amanuense.
México, 1992
178 pp

INTERNET

http://www-cia.mty itesm.ms/historia.html
http //www.lania mx/spanish/eventos/taller html
http //wwwcsai.diepa unipa it/research/projects/projects.html
http //www yahoo com/science/computer/science/artificial

Capítulo

1

Introducción

El nombre **PROLOG** proviene de **PROgramación LOgica** El lenguaje fue concebido por el profesor francés Alain Colmerauer, el cual investigaba la obtención de un sistema que proporcionara una detección eficiente de errores sintácticos en un programa de computadora Durante esta búsqueda en 1963 logró un análisis más complejo del lenguaje natural, y la primera versión definitiva de **PROLOG** se produjo en la Universidad de Marsella en 1973.

PROLOG estuvo en principio limitado a círculos académicos franceses y a la arquitectura de las computadoras de este país, pero el interés mostrado por la empresa Digital Equipment Company llevó a la implementación del lenguaje en computadoras de dicha marca, de este modo se *abrió camino en el mundo comercial El trabajo llevado a cabo en Gran Bretaña por el* profesor Robert Kowalski y un grupo de investigadores del Imperial College de Londres llamó poderosamente la atención del mundo entero **PROLOG** también se desarrollo en otros puntos. en Marsella en 1982 se obtuvieron nuevos resultados, los trabajos de Edimburgo con el **DEC-10 PROLOG**, la versión de Clocksin y Mellish llamada "**C & M syntax**" o "**Edimburgo syntax**" es aceptada como el estándar, la versión húngara llamada **M-PROLOG** y, por supuesto, el proyecto japonés.

La diferencia entre **PROLOG** y otros lenguajes es que un programa en **PROLOG** dice a la computadora que hacer (una técnica llamada programación declarativa) mientras que los programas en otros lenguajes dicen a la computadora como hacerlo (programación procedural). **PROLOG** hace esto haciendo deducciones y derivaciones de los hechos y reglas almacenadas en una base de datos, llevadas a cabo por los queries (preguntas) del usuario, como resultado del mecanismo de inferencia incorporado de **PROLOG** llamado backtracking (búsqueda hacia atrás).

La potencia y la flexibilidad de la base de datos de **PROLOG** junto con la facilidad con que puede ser extendida y modificada, lo convierte en un lenguaje atractivo para el usuario comercial, además de ser un lenguaje fácil de aprender

PROLOG fue originalmente diseñado para procesos de información no numérica, pero actualmente tiene extensiones matemáticas Existen versiones de **PROLOG** incluyendo aquellas que trabajan sobre sistema operativos **CPM**, **MSDOS** y **UNIX**.

Para la programación en computadora, necesitamos simplificar y regularizar la sintaxis sin perder mucho de la semántica Para adaptar el lenguaje natural a la programación, se usan generalizaciones y patrones que simplifican nuestra expresión de conocimiento. La **semántica** es el significado o las ideas a ser expresadas, y la **sintaxis** es la estructura organizacional del lenguaje.

Capítulo

2

Semántica

Estructura de un programa en Prolog

Un programa en **PROLOG** es un archivo que contiene cláusulas y/o queries en el formato apropiado de **PROLOG**.

Las cláusulas son de dos tipos **hechos** y **reglas** Los **hechos** son **átomos** o **estructuras**. Por ejemplo

verde(árbol).
hombre(pedro).

Las **reglas** están formadas por un término seguido del símbolo "**:-**" y uno o más términos unidos por comas

verde(X) :- húmedo(X).
mari (gusta, X) :- guapo(X).

La **cabeza** de un regla es el término que está a la izquierda de "**:-**", el **cuerpo** es el término que está a la derecha

Un **predicado** puede ser definido por una mezcla de reglas y hechos. Estos son llamados cláusulas de un predicado Un **predicado** o una de sus cláusulas es con frecuencia referido como un **procedimiento** A diferencia de otros lenguajes, **PROLOG** permute al mismo procedimiento tener mas de una definición, cada cláusula es una de sus definiciones.

Un **query** es una orden para que la computadora haga algo.

Hechos

Una parte fundamental de la descripción de un universo en **PROLOG**, son los **hechos** que el universo contiene Cada **hecho** es como una oración, que describe una parte de la información acerca de uno o mas objetos del universo. Una colección de **hechos** para **PROLOG** es llamada una **base** o **base de datos** Por ejemplo

capital(yucatan,merida).

Este **hecho** de PROLOG muestra que hay una relación entre dos objetos, yucatan y merida

Una **relación** es una regla que une a entes, por ejemplo "Tomás es-hermano-de Jaime" define una relación entre Tomás y Jaime (se une mediante guiones las palabras "es-hermano-de" ya que **PROLOG** solo acepta una única palabra como denominación de una relación, y los guiones sirven para hacer que las tres palabras aparezcan como una sola). Este tipo de relación que une a dos entes se conoce como **relación binaria**

No todas las relaciones son binarias, es posible tener relaciones que sólo implican a un ente, por ejemplo "Jaime es-un-hombre" Esta es una **relación unaria** ya que sólo implica a Jaime. También es posible definir **relaciones ternarias**, o sea, relaciones en las que aparezcan implicados tres entes, *tal es el caso de* "Alejandro da dinero a Jorge" A los entes que intervienen en las relaciones se les llama **argumentos** de la relación De este modo, una relación binaria poseería dos argumentos y una relación unaria tendría tan sólo un argumento Las relaciones binarias pueden ser expresadas, de forma general, como "**X R Y**", donde R puede ser cualquier relación, y X e Y son los argumentos. La forma general de una relación unaria es "**X R**" donde X es el argumento de la relación R

Para escribir hechos se utilizan reglas, las cuales son llamadas **reglas de sintaxis** El nombre que define la relación aparece primero, el cual es llamado **predicado**, los nombres de los objetos aparecen dentro del paréntesis, separados por una coma Como en una oración el hecho finaliza con un punto (.) Por ejemplo:

alimenta(carlos,perro).

Tanto el nombre de la relación como el del objeto empieza con letras minúsculas. Esto es para indicar a **PROLOG** que estos son nombres de objetos específicos y que los nombres no cambiarán, es decir **constantes** Hay que notar que no hay espacios dentro del hecho

En el ejemplo anterior, se puede interpretar el hecho como "Carlos alimenta al perro". El orden de los nombres de los objetos y la interpretación de los hechos son elegidos libremente

Algunas veces solamente un nombre de objeto aparece dentro del paréntesis del hecho El nombre fuera del paréntesis normalmente representa un **atributo** o **característica** del objeto Por ejemplo

mujer(maria).

Los hechos en **PROLOG** simplemente permiten expresar relaciones arbitrarias entre objetos

Queries (Preguntas)

Cuando se tiene una base de hechos, se puede usar **PROLOG** para hacer preguntas acerca del universo que se ha descrito en la base de datos. Un **query** sigue las mismas reglas de sintaxis que los hechos, teniendo al inicio un signo " **?-**" Por ejemplo

?-capital(yucatan,merida).

PROLOG busca si un hecho existe en la base de datos usando un **patrón de coincidencia** Cuando se hace una pregunta, **PROLOG** compara el primer hecho en la base de datos con la expresión dada en la pregunta Si esta no coincide, **PROLOG** va al siguiente hecho para tratar de relacionar, y continua el proceso hasta que una de dos cosas sucede Ya sea que este tenga éxito en encontrar el hecho, con lo cual **PROLOG** responde "**yes**" o este checa la base hasta el final y no logra encontrar el hecho , y responde "**no**"

PROLOG marca el lugar en la base de datos cuando una coincidencia es encontrada

Conjunción

Para queries más complejos en **PROLOG** se usa la conjunción "," la cual significa "**y**"

Suponiendo que se tiene la siguiente base de datos

gustar(maria,comida).
gustar(maria,vino).
gustar(juan,vino).
gustar(juan,maria).

y se pregunta

?-gustar(juan,maria) , gustar(maria,juan)

Cuando una secuencia de objetos (separados por comas) es dada a **PROLOG**, éste intenta satisfacer cada objetivo por medio de una búsqueda en la base de datos para satisfacer el objetivo en turno Todos los objetivos tienen que ser satisfechos para que la secuencia pueda ser satisfecha

La respuesta a la pregunta es no Esto es porque el hecho de que a Juan le gusta María, es el primer objetivo verdadero Sin embargo el segundo no lo es, ya que la ocurrencia de que a María le gusta Juan es falsa

Cuando **PROLOG** intenta repetidamente satisfacer y resatisfacer objetivos en una conjunción es llamada "**backtracking**"

Variables

Algunas veces se desea que **PROLOG** compruebe información contra una base de datos e informe un "**yes**" o "**no**" Frecuentemente, se desea que **PROLOG** llene el nombre de un objeto que pertenece a una relación Para hacer esto, se debe hacer una pregunta que contenga el patrón que se esta buscando, pero en lugar del nombre que deseamos, se pone un nombre de **variable** En **PROLOG** se especifica que un nombre es una **variable**, si el nombre empieza con una letra mayúscula. Por ejemplo

?-capital(oaxaca,Ciudad).

Ciudad es una variable ya que esta empieza con una letra mayúscula, C Si se hace esta pregunta a la base, **PROLOG** responder con **Ciudad = oaxaca**.

Se pueden usar variables para cualquiera de los objetos que están dentro de el paréntesis. Por ejemplo

?-capital(Estado,guanajuato).

se obtendrá la respuesta

Estado=guanajuato

Además no se limita a una variable en cada pregunta

?-capital(Estado,Ciudad).
Estado=aguascaliates Ciudad=aguascalientes

En este caso, cualquiera de los hechos en la base de datos podría corresponder **PROLOG** responde con la primera pareja de objetos que éste encuentra

Mientras las variables pueden ponerse en el lugar de cualquiera de los nombres de objetos en un hecho, ellos no pueden ser usados en lugar del nombre de la relación.

Cuando **PROLOG** utiliza una variable, la variable puede ser **instanciada** o **no instanciada** Una variable es **instanciada** cuando hay un objeto asociado a ella. Una variable es **no instanciada** cuando no tiene asociado un objeto a ella

Disyunción

Cuando **PROLOG** regresa un nombre como una constante en respuesta con una variable, éste se detiene esperando que la respuesta sea aceptada En vez de dar un <<**enter**>>, se puede responder con un punto y coma ";" seguido por un <<**enter**>> En **PROLOG** el ";" significa "**o**" Interpreta esto para seguir buscando a través de la base de datos y encontrar otra coincidencia a la pregunta Por ejemplo:

?-capital(Estado,Ciudad)

PROLOG responde:

Estado=aguascalientes Ciudad=aguascalientes

y se detiene Si se responde con un ";" **PROLOG** continua

Estado=baja_california Ciudad=tijuana

y así sucesivamente hasta llegar al final

Reglas

Los hechos forman la base para las respuestas que **PROLOG** da a las preguntas realizadas. Los hechos son oraciones que son verdaderas en la base de datos, así mismo son los bloques fundamentales de la base que describe un universo Se puede ampliar la descripción del universo que se ha definido agregando **reglas** a la base de datos Estas **reglas** están basadas en hechos o en otras reglas y hechos Las **reglas** permiten dar más información a las respuestas de **PROLOG**

En **PROLOG**, las reglas son usadas cuando se quiere decir que un hecho depende de un grupo de otros hechos y también para expresar definiciones

Una **regla** consiste de una **cabeza (conclusión)** y un **cuerpo (requerimientos)**. La **cabeza** y el **cuerpo** están conectado por la palabra ":-" y finalizan con un **punto**.

La **regla** especifica que la **conclusión** puede ser considerada verdadera si los **requerimientos** son verdaderos Por ejemplo

gustar(juan,X) :- gustar(X,vino) , gustar(X,comida).

en otras palabras, a Juan le gusta alguien a quién le guste el vino y la comida

En la regla de arriba la variable X es usada tres veces Cuando X llega a ser instanciada en el mismo objeto, todas las X's son instanciadas. El alcance de X es toda la regla, incluyendo desde la cabeza hasta el punto.

Listas

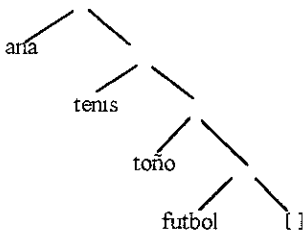
La **lista** es una estructura de datos ampliamente usada Una **lista** es una secuencia ordenada de elementos que pueden tener cualquier longitud

Existen dos tipos de listas, la **lista vacía** que se escribe en **PROLOG** como [ ] y la **lista no vacía** que contiene el primer término llamado **cabeza de la lista** y la **cola** que es el resto de la lista. Por ejemplo

[ana, tenis, toño, futbol]

la **cabeza** es ana y la **cola** de la lista.

[tenis, toño, futbol]



La notación de la barra vertical es un hecho más general, se puede listar un número de elementos seguidos por "|" y el resto de la **lista** Por ejemplo

[a, b, c] = [a | [b,c]] = [a, b | [c]] = [a, b, c | [ ]]

La **lista** ayuda a proporcionar una estructura más compacta a los programas, y una mayor eficiencia y manejabilidad a los programadores y usuarios La **lista** es una de las estructuras de datos que tiene una mayor aplicación en programas de **IA**, asimismo, es la estructura fundamental en el lenguaje **LISP PROLOG** también tiene gran capacidad para el procedimiento de **listas** y usa métodos similares a los empleados en **LISP**.

Recursión

La **recursión** es usada de dos maneras Puede ser usada para describir estructuras que tienen otras estructuras como componentes, y también puede ser usada para describir programas que necesitan satisfacer una copia de ellos mismos antes de que ellos mismos puedan ocurrir. Por ejemplo, suponiendo que se tiene un automóvil llamado neón que mucha gente ha tenido

propietario(neon,Persona) :- compro(neon,Persona,Vendedor), propietario(neon,vendedor).

Esta regla puede ser leída como "una persona es propietaria del automóvil neón, si aquella persona compró el neón de alguien y aquel alguien fue propietario del neón". Esta clase de **regla** es llamada **recursiva** ya que la relación en la conclusión aparece otra vez (recurrente) en el cuerpo de la regla, donde los requerimientos son especificados Las reglas recursivas son útiles cuando una relación va de un objeto a otro y de aquel objeto al otro En este caso, el propietario del neón va de una persona a otra

El patrón de la relación, mover un objeto a otro, es definido por la forma en que las variables en la regla están relacionadas Las personas representadas por las variables toman diferentes papeles en diferentes partes de la regla

Una **regla recursiva** es una manera de generar una cadena de relaciones Para que una regla recursiva sea efectiva debe haber en algún lugar donde la recursión se detenga. Para el ejemplo anterior, alguien compró el neon en la agencia

propietario(neón, Persona) :- compro(neon,Persona,agencia).

Algunas veces las reglas recursivas no se comportan como se ha planeado Se pueden evitar problemas de recursividad tomando en cuenta lo siguiente

- La condición para detener la recursividad debe de ser escrita antes de la regla recursiva
- Poner el predicado recurrente al final del cuerpo de la regla recursiva, esto es llamado cola recursiva y el proceso, recursion de cola
- Una regla no debe de tener en sus requerimientos la conclusión de una segunda regla, si la segunda regla tiene la conclusión de la primera regla en sus requerimientos Por ejemplo:

hermano(jose,ana).
hermano(El,Ella) :- hermana(Ella,El).
hermana(Ella,El) :- hermano(El, Ella).

El Cut

El **cut** permite decirle previamente a **PROLOG** cual objetivo no necesita considerar cuando regresa hacia atrás El efecto es hacer inaccesible el lugar marcado por los objetivos para que no puedan ser resatisfechos " Hay dos razones por la que es importante hacer esto:

- El programa puede operar rápidamente porque este no pierde tiempo intentando satisfacer los objetivos que previamente se especificaron que no contribuirán a una solución.
- El programa puede ocupar menos espacio de memoria si el backtracking apunta a registros que no tienen que ser examinados posteriormente.

Se puede dividir el uso común del cut en tres áreas principales:

- El primero se refiere a los lugares donde se quiere decir al sistema de PROLOG que este ha encontrado la regla correcta para un objetivo particular.
- El segundo se refiere a los lugares donde se quiere decir al sistema PROLOG que fallo un objetivo particular sin probar las soluciones alternativas.
- El tercero se refiere a los lugares donde queremos terminar la generación de soluciones alternativas a través del backtracking. " (1)

Capítulo

3

Sintaxis

Tipos básicos de datos

Términos

Los programas de **PROLOG** están construidos por **términos** Un **término** es una **constante**, una **variable** o una **estructura**. Cada **término** esta escrito como una secuencia de caracteres Los caracteres están divididos dentro de cuatro categorías que son

- A B C D E F G H Y J K L M N O P Q R S T U V W X Y Z
- a b c d e f g h y j k l m n o p q r s t u v w x y z
- 1 2 3 4 5 6 7 8 9
- + - * / \ ~ ^ < > ? @ # $ &

Cada tipo de término ya sea una constante, variable o estructura tiene diferentes reglas de como construir sus nombres

Constantes

Las **constantes** son nombres específicos de objetos o relaciones específicas. Hay dos tipos de **constantes**. **átomos** y **enteros**.

Los **átomos** pueden ser construidos de tres formas

1) Cadena de caracteres, dígitos y subrayado "_", empezando con letras minúsculas

ana
x25
x_25AB
alfa_beta_procedimiento

2) Cadena de caracteres especiales

<---->
=====>
...
.:.
::=

Cuando usamos **átomos** de esta forma es necesario tomar en cuenta que algunas cadenas de caracteres especiales ya han están predefinidas, un ejemplo es "- -"

3) Cadena de caracteres encerrados entre comillas sencillas. Pueden empezar con letras mayúsculas Son encerradas entre comillas sencillas para distinguirlas de las variables

' Tom'
'Sur_América'
'Sara Jones'

Los **números** son otro tipo de constantes, que pueden ser **números reales** o **números enteros** La sintaxis de los enteros es simple

1
1313
-97

Normalmente el rango en que pueden ser implementados es entre 16383 y -16383

Variables

El segundo tipo de término usado en **PROLOG** es la **variable**. Las **variables** pueden ser iguales a un átomo, excepto que estos pueden empezar con letra mayúscula o subrayado, ejemplo

Resultado
Objeto2
Participante_ lista
_ x23

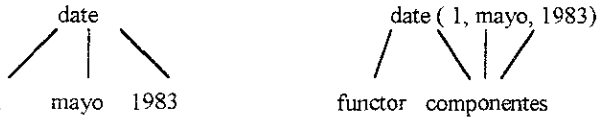
Cuando una variable aparece en una cláusula una sola vez no se tiene que inventar un nombre para esta ya que se puede utilizar la variable anónima, la cual esta escrita por un simbolo de subrayado "_"

Estructuras

Una **estructura** es un objeto simple (functor) el cual consiste de una colección de otros objetos llamados componentes Los **componentes** son agrupados dentro de una estructura simple Las estructuras son a veces llamadas "**términos complejos**" o "**términos compuestos**"

Un adecuado functor es **date** Por ejemplo 1ero de Mayo de 1983 puede ser escrito como.
date(1,mayo,1983)

Todos los componentes en este ejemplo son constantes Los **componentes** pueden ser **variables** u otras **estructuras** Todas las estructuras de objetos pueden ser dibujadas como **árboles**



Las **estructuras** ayudan a organizar los datos en un programa porque ellos permiten tener un grupo de información relacionada para ser tratada como un solo objeto en vez de entidades separadas. La forma en que se pueden descomponer los datos dentro de los componentes depende de que tipo de problema se quiera resolver

El **functor** es definido por dos términos.

- El **nombre**, su sintaxis es como la del átomo
- La **aridad**, es el número de componentes que el functor tiene

Para referirse al functor f(a,b,c), en vez de decir "f, con 3 argumentos" o "f, con aridad de 3" se dice "f/3"

Cuando una estructura contiene otras estructuras, tales como f(a,b(c,d),e), el functor más alejado (f/3, en este caso) es llamado el **Functor Principal** de la estructura

Cadenas

Los nombres de constantes y variables son construidas por medio de cadenas de caracteres . **PROLOG** reconoce dos tipos de caracteres. **caracteres imprimibles** y **caracteres no imprimibles**. Los **caracteres imprimibles** aparecen en pantalla, mientras que los **no imprimibles** llevan solo una acción, tales como espacio en blanco, empezar una nueva línea de texto y quizá un sonido.

Operadores

Las operaciones aritméticas se escriben comúnmente como **operadores** En la expresión aritmética "x + y * z", el signo + y el signo * son llamados **operadores**

Los **operadores** no realizan ningún tipo de operación aritmética. Así, en **PROLOG**, 3+4 no significa que sea igual a 7 El término 3+4 es otra forma de escribir +(3,4), la cual es una estructura de datos

La sintaxis de un término con **operadores** depende en parte de la posición del **operador** Los **operadores** más (+), guión (-), asterisco (*) y barra inclinada (/) se escriben entre sus respectivos argumentos, llamados **infijos** También es posible poner operadores delante de sus argumentos, como en -x+y, donde el guión delante de la *x* se utiliza para indicar negación A estos **operadores** se les llama **prefijos** Algunos operadores pueden ir detrás de su argumento, por ejemplo, el operador factorial de x se escribe x!, donde el signo de exclamación se utiliza para indicar factorial A los operadores que se escriben detrás de su argumento se les llama **sufijos** De esta manera la posición de un **operador** indica dónde se escribe con relación a sus argumentos

> " La precedencia de un operador se utiliza para indicar qué operación se efectúa primero. Cada operador utilizado en PROLOG tiene una clase de precedencia asociada a él. En PROLOG, los operadores de multiplicación y división se encuentran en una clase de precedencia mayor que los de suma y resta, de forma que el término a-b/c es lo mismo que el término -(a,/(b,c)). Si nos encontramos con la expresión "8/2/2", ¿significa esto "(8/2)/2" u "8/(2/2)"?. Para poder distinguir entre estos dos casos, debemos saber si el operador es asociativo por la izquierda o asociativo por la derecha. Un operador asociativo por la izquierda debe tener a la izquierda operaciones con una precedencia igual o menor, y operaciones con una precedencia menor a la derecha. Por ejemplo, todas las operaciones aritméticas (suma, resta, multiplicación y división) son asociativas por la izquierda. Esto significa que expresiones como "8/4/4" se tratan como "(8/4)/4". De igual forma "5 + 8/2/2" significa "5 + ((8/2)/2)". " (2)

Igualdad y Coincidencia

Un operador importante es el de la **igualdad**, el cual es un operador infijo escrito como "="

" Dado un objetivo de la forma *X* = *Y*, donde *X* e *Y* son dos términos a los que se permite contener variables no instanciadas, las reglas para decidir si *X* e *Y* son iguales son las siguientes:

- Si *X* es una variable no instanciada, y si *Y* esta instanciada a cualquier término, entonces *X* e *Y* son iguales, además, *X* quedará instanciada a lo que valga *Y*.
- Los enteros y los átomos son iguales a si mismos.

- Dos estructuras son iguales si tienen el mismo functor y el mismo número de componentes y todos y cada uno de los correspondientes componentes son iguales. " (3)

PROLOG también proporciona un predicado "\=", que se interpreta como no igual. El objetivo $X \backslash= Y$ sucede si $X = Y$ falla, y falla si X y Y suceden

Aritmética

Los operadores aritméticos son usados para comparar números y para hacer cálculos Los argumentos pueden ser variables instanciadas como enteros o pueden ser enteros escritos como constantes Hay algunos otros operadores para comparar números. La siguiente tabla muestra los operadores de comparación, los cuales son de tipo infijo

X = Y	X e Y tienen el mismo número
X \= Y	X e Y tienen diferentes números
X < Y	X es menor que Y
X > Y	X es mayor que Y
X =< Y	X es menor o igual que Y
X >= Y	X es mayor o igual que Y

El operador "**is**" es nuevo El "**is**" es un operador infijo, este argumento va a la derecha del término, el cual es interpretado como una expresión aritmética Para satisfacer un "**is**", **PROLOG** primero evalúa que este a la derecha del argumento de acuerdo a las reglas de aritmética

Unificación

Dos términos pueden ser **unificados (relacionados)** si ellos son semejantes o pueden ser hechos semejantes por instanciación (dando valores a) de variables. (Una instanciación puede hacer una variable igual a otra) Por ejemplo:

1. " f (X,Y) unifica con f (a,b) instanciando X=a, Y=b.
2. f (X,Y) unifica con f (Z,Z) haciendo X, Y y Z igual a la variable.
3. f (X,Y) no unifica con f (a,b) ya que X no puede tener dos valores diferentes al mismo tiempo en el mismo término. " (4)

La **unificación** es independiente del orden, si se **unifica** un conjunto de términos se puede obtener el mismo resultado sin tener en cuenta el orden en el cual los términos son encontrados. Mucho de el poder de **PROLOG**, particularmente para el proceso del lenguaje natural , viene de el orden independiente de unificación

Pies de página

(1) Programming in PROLOG. Clocksin W F y Mellish C S , pag. 75
(2) Programación en PROLOG, Clocksin W.F y Mellish C S , pag 37-38
(3) Programming in PROLOG,. Clocksin W.F y Mellish C S , pag 29
(4) Natural Lenguage Processing for Prolog Programmers, Covington, Michael A , pag 293

Bibliografía

Prolog. Programming for Artificial Intelligence
BRATKO, Ivan
Ed. Addison-Wesley
Gran Bretaña, 1990.
2a ed
597 pp.

Programación en PROLOG
CLOCKSIN, W F y MELLISH, C.S.
Colección Ciencia Informática
Ed. Gustavo Gili
Barcelona, 1987
270 pp

Programming in Prolog.
CLOCKSIN, W F y MELLISH, C.S.
Ed Springer-Verlag.
Estados Unidos, 1994
4a ed
263 pp

Natural Lenguage Processing for Prolog Programmers.
COVINGTON, Michael A
Ed Prentice Hall
Estados Unidos, 1994
348 pp

Inteligencia Artificial y PROLOG en Microordenadores
MCALLISTER J.
Ed. Marcombo.
España, 1991
221 pp

A Prolog Primer
ROGERS, Jean B
Ed Addison-Wesley
Estados Unidos, 1986.
219 pp.

Parte 3 Aplicando Prolog en las Áreas de Inteligencia Artificial

Capítulo

1

Búsqueda de Soluciones

Introducción

En los primeros días de la investigación de la IA, el desarrollo de buenos métodos de búsqueda fue el objetivo principal Fue necesario que los programadores idearan buenas técnicas de búsqueda para resolver problemas debido a las limitaciones de las computadoras usadas en esa época Además, los programadores deseaban buenas técnicas de búsqueda porque pensaban que la búsqueda era fundamental para resolver problemas.

La **búsqueda** es el proceso de examinar un conjunto grande de soluciones a un problema a fin de encontrar la mejor La búsqueda es un proceso de gran importancia en la resolución de problemas difíciles para los que no se dispone de técnicas más directas

Para construir un sistema que resuelva un problema específico, es necesario realizar estas cuatro acciones

1 Definir el problema con precisión La definición debe incluir especificaciones precisas tanto sobre la o las situaciones iniciales como sobre las situaciones finales que se aceptarán como soluciones al problema

2 Analizar el problema Algunas características de gran importancia pueden tener un gran efecto sobre la conveniencia o no de utilizar las diversas técnicas que resuelven el problema.

3. Aislar y representar el conocimiento necesario para resolver el problema

4 Elegir la mejor técnica(s) que resuelva el problema y aplicarla(s) al problema

Definición del Problema mediante una Búsqueda en un Espacio de Estados

La representación como **espacio de estados** forma la base de la mayoría de los métodos de IA Su estructura se corresponde con la estructura de la resolución de problemas por dos importantes razones

- Permite definir formalmente el problema, mediante la necesidad de convertir alguna situación dada en la situación deseada usando un conjunto de operaciones permitidas
- Permite definir el proceso de resolución de un problema como una combinación de técnicas conocidas (representadas por una regla que define un movimiento en el espacio) y **búsqueda**, la técnica general de exploración en el espacio intenta encontrar alguna ruta desde el estado actual hasta un estado objetivo.

Con frecuencia, los problemas contienen de forma explícita o implícita la exigencia de encontrar la secuencia más corta (más rentable) Si es así, este requisito provoca un efecto significativo en la elección del mecanismo que guía la búsqueda de una solución

El primer paso que se debe dar para diseñar un programa que resuelva un problema, es crear una descripción formal y manejable del propio problema Es adecuado poder escribir programas tales que ellos mismos produzcan descripciones formales a partir de descripciones informales. Este proceso recibe el nombre de **operacionalización**

Para poder producir una descripción formal de un problema, debe hacerse lo siguiente:

1 Definir un espacio de estados que contenga todas las configuraciones posibles de los objetos más relevantes (y quizá algunos imposibles). Es, por supuesto, posible definir este espacio sin tener que hacer una enumeración de todos y cada uno de los estados que contiene.
2 Identificar uno o más estados que describan situaciones en las que comience el proceso de resolución del problema. Estos se denominan **estados iniciales**
3 Especificar uno o más estados que pudieran ser soluciones aceptables del problema. Estos estados se denominan **estados objetivo**
4 Especificar un conjunto de reglas que describan las acciones (operadores) disponibles

De esta forma, el problema puede resolverse con el uso de las reglas en combinación con una estrategia apropiada de control para trasladarse a través del espacio problema hasta encontrar una ruta desde un **estado inicial** hasta un **estado objetivo**. De esta forma, el **proceso de búsqueda** es fundamental en el proceso de resolución de problemas. El hecho de que el proceso de búsqueda forma la base de la resolución de problemas no significa, sin embargo, que no puedan utilizarse otras aproximaciones más directas al problema Cuando sea posible, deben incluirse como partes de la búsqueda, codificándolas como reglas.

Los Sistemas de Producción

Debido a que la búsqueda es el núcleo de muchos procesos inteligentes, es adecuado estructurar los programas de IA de forma que se facilite describir y desarrollar el proceso de búsqueda Los sistemas de producción proporcionan tales estructuras

Un **sistema de producción** consiste en

- Un conjunto de reglas compuestas por una parte izquierda (un patrón) que determina la aplicabilidad de la regla, y una parte derecha, que describe la operación que se lleva a cabo si se aplica la regla
- Una o más bases de datos/conocimientos que contengan cualquier tipo de información apropiada para la tarea en particular Partes de la base de datos pueden ser permanentes, mientras que otras pueden hacer referencia sólo a la solución del problema actual La información almacenada en estas bases de datos debe estructurarse de forma adecuada.
- Una estrategia de control que especifique el orden en el que las reglas se comparan con la base de datos, y la forma de resolver los conflictos que surjan cuando varias reglas puedan ser aplicadas a la vez.
- Un aplicador de reglas

Búsqueda a Ciegas

Las técnicas de **búsqueda a ciegas** examinan todo el espacio de búsqueda de forma ordenada para encontrar una conclusión Las búsquedas a ciegas comienzan en el nodo inicial y exploran sistemáticamente el espacio de izquierda a derecha

Aunque las técnicas de búsqueda a ciegas garantizan un solución, tiene la desventaja de ser un proceso lento y tedioso Cuando se trata de un espacio de búsqueda de gran tamaño, la búsqueda de una solución puede llevar demasiado tiempo

Se puede calcular matemáticamente el número total de soluciones posibles de un espacio de busqueda Si cada nodo en el espacio de búsqueda presenta *n* derivaciones hacia otros nodos y el espacio tiene d niveles (el nodo inicial no se considera un nivel), el número total de nodos del espacio de búsqueda se calcula elevando *n* a la *d-ésima* potencia

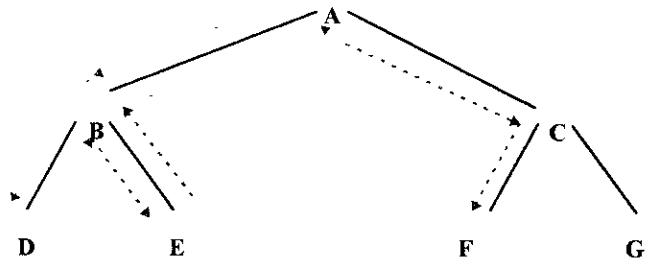
$$\text{NUMERO TOTAL DE NODOS} = n^{d}$$

cuando mayor sea el número de nodos y el número de derivaciones por nodo, más rápidamente se expandirá el espacio de búsqueda Esta expansión matemática del espacio de búsqueda se denomina **explosión combinatoria**

Existen cuatro formas fundamentales de examinar un espacio de búsqueda:, **búsqueda en profundidad**, **búsqueda en anchura**, **búsqueda progresiva** y **búsqueda regresiva**

Búsqueda en Profundidad

La **búsqueda en profundidad** comienza en el nodo inicial y va recorriendo los distintos niveles del espacio de búsqueda hasta alcanzar el nodo más profundo de la derivación más a la izquierda Si el nodo final de dicha derivación no es uno de los objetivos, se retrocede hasta un nivel en que se puede continuar la búsqueda El proceso de búsqueda continua de izquierda a derecha hasta que se encuentra una solución El sistema de búsqueda en profundidad se ilustra en la siguiente figura

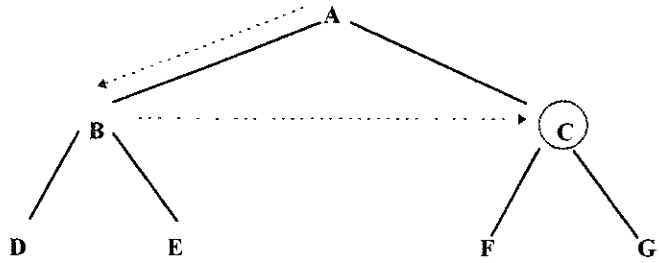


Ventajas de la Búsqueda Primero en Profundidad

- La búsqueda primero en profundidad necesita menos memoria ya que sólo se almacenan los nodos del camino que se sigue en ese instante Esto contrasta con la búsqueda primero en anchura en la que debe almacenarse todo el árbol que haya sido generado hasta ese momento.
- Si se tiene cuidado en ordenar los estados alternativos sucesores, la búsqueda primero en profundidad puede encontrar una solución sin tener que examinar gran parte del espacio de estados. En el caso de la búsqueda primero en anchura debe examinarse todas las partes del árbol de nivel n antes de comenzar con los nodos del nivel $n + 1$ Esto es particularmente relevante en el caso de que existan varias soluciones aceptables. La búsqueda primero en profundidad acaba al encontrar una de ellas.

Búsqueda en Anchura

Una **búsqueda en anchura** comienza en el nodo inicial y continua hacia abajo examinando todos los nodos de un nivel antes de pasar al siguiente Como se ilustra en la figura, la búsqueda en anchura examina de izquierda a derecha todos los nodos de un nivel antes de pasar a la siguiente



Ventajas de la búsqueda Primero en Anchura

- La búsqueda primero en anchura no queda explorando callejones sin salida. Esto se contrapone con la búsqueda primero en profundidad en la que se puede seguir una ruta infructuosa durante mucho tiempo, antes de acabar en un estado sin sucesores Esto es particularmente un problema en la búsqueda primero en profundidad si hay ciclos
- Si existe una solución, la búsqueda primero en anchura garantiza que se logre encontrarla. Además, si existen múltiples soluciones, se encuentra la solución mínima (es decir, la que requiere el mínimo número de pasos Esto está garantizado por el hecho de que se explora una ruta larga hasta que se hayan examinado todas las rutas más cortas que ella. En cambio, en la búsqueda primero en profundidad es posible encontrar una solución larga en alguna parte del árbol, cuando puede existir otra mucho más corta en alguna parte inexplorada del mismo.

Búsqueda Progresiva

Las técnicas de **búsqueda progresiva**, también denominadas "**razonamiento progresivo**" o "**encadenamiento progresivo**", comienzan en el nodo inicial y prosiguen hacia abajo hasta encontrar un objetivo o nodo solución Las técnicas de búsqueda progresiva son una mezcla de las técnicas de búsqueda en anchura y profundidad

Búsqueda Regresiva

En las técnicas de **búsqueda regresiva** , denominadas también de "**razonamiento regresivo**" o de "**encadenamiento regresivo**", la búsqueda comienza en el nodo objetivo Una forma de resolver un problema consiste en asumir que la solución tiene una forma determinada y entonces buscar pruebas que soporten dicha suposición Si la búsqueda basada en un objetivo falla, puede iniciarse otra partiendo de otro objetivo e intentando encontrar hechos que soporten dicho objetivo. Este tipo de búsquedas se denominan **controladas por los objetivos** y simulan un tipo de razonamiento inductivo Para demostrar la validez del objetivo seleccionado, el camino de búsqueda se puede recorrer en sentido inverso

Búsqueda Heurística

La **heurística** es una técnica que aumenta la eficiencia de un proceso de búsqueda Algunas heurísticas ayudan a guiar el proceso de búsqueda sin eliminar alguna demanda de completitud que el proceso haya podido tener previamente. Otras pueden ocasionalmente causar que una buena ruta sea pasada por alto Pero, en promedio, mejoran la calidad de las rutas que exploran

El propósito de una función heurística es el de guiar el proceso de búsqueda en la dirección más provechosa sugiriendo qué camino se debe seguir primero cuando hay más de uno disponible. Cuando más exactamente estime la función heurística los méritos de cada nodo del espacio de búsqueda (**árbol**), más directo sera el proceso de solución.

Existen algunas **heurísticas de propósito general** que son adecuadas para una amplia variedad de dominios de problemas Además, es posible construir **heurísticas de propósito especial** que exploten el conocimiento específico del dominio para resolver problemas particulares

La **heurística del vecino más próximo** es un ejemplo de una buena heurística de propósito general valida para varios problemas combinatorios Consiste en seleccionar en cada paso la alternativa localmente superior

En la heurística del vecino mas próximo, es con frecuencia posible probar límites de error, lo cual proporciona la certeza de que no se esta pagando un precio demasiado alto con la precisión en beneficio de la velocidad En muchos problemas de IA, sin embargo, no es posible dar estos límites por dos motivos

- Para los problemas del mundo real, normalmente es difícil medir con precisión el valor de una solución particular
- Para los problemas del mundo real, normalmente es adecuado introducir heurísticas basadas en un conocimiento relativamente desestructurado Con frecuencia es imposible definir este conocimiento de forma que pueda llevarse a cabo un análisis matemático de su efecto sobre el proceso de búsqueda

Sin las heurísticas, se entraría en una explosión combinatoria Solo este argumento podría resultar suficiente para demostrar la necesidad de su uso No obstante, existen otros argumentos:

- Normalmente no se necesita una solución óptima, con frecuencia una buena aproximación es adecuada Buscan una solución que satisfaga algún conjunto de requisitos, y tan pronto como la encuentran, terminan
- Si bien las aproximaciones que se logran con una heurística pueden no ser muy buenas en los peores casos, estos peores casos raramente ocurren en el mundo real.
- Intentar comprender porque funciona una heurística o porque no lo hace, normalmente sirve para profundizar en la comprensión del problema

Existen dos formas fundamentales de incorporación de conocimiento heurístico específico del dominio de un proceso de búsqueda basado en reglas·

- En las mismas reglas
- Como una función heurística que evalúa los estados individuales del problema y determina su grado de "deseabilidad"

Una función heurística es una correspondencia entre las descripciones de estado del problema hacia alguna medida de deseabilidad, normalmente representada por números. Los aspectos del problema que se consideran, cómo se evalúan estos aspectos, y los pesos que se dan a los aspectos individuales, se eligen de forma que el valor que la función heurística da a un nodo en el proceso de búsqueda, sea una estimación tan buena como sea posible para ver si ese nodo pertenece a la ruta que conduce a la solución

Todos los procesos de búsqueda pueden tratarse como un recorrido sobre una estructura en forma de árbol en el que cada nodo representa un estado del problema y cada enlace representa una relación entre los estados que representan los nodos que están conectados

Cabe mencionar algunos aspectos importantes de las técnicas de búsqueda de propósito general

- La dirección en la que se va a conducir la búsqueda (razonamiento **hacia delante** o **hacia atrás**) Se puede buscar hacia delante partiendo del estado inicial hacia el estado objetivo, o se puede buscar hacia atrás partiendo del objetivo
- La forma de seleccionar las reglas a aplicar (**emparejamiento o matching**) Los sistemas de producción emplean por lo general la mayor parte del tiempo en buscar reglas para aplicarlas, de forma que hacer eficientes procedimientos para emparejar reglas con estados es un factor crítico
- La forma de representar los nodos en el proceso de búsqueda (**el problema de la representación del conocimiento** y **el problema del marco (frame problem)**).

Técnicas de Búsqueda Heurística

Menor Coste

La **búsqueda de menor coste**, toma el camino del mínimo esfuerzo Comienza generando rutas completas, manteniéndose la ruta más corta encontrada hasta el momento. Deja de explorar una ruta tan pronto como su distancia total, hasta ese momento, sea mayor que la que se ha marcado como la más corta Usar esta técnica garantiza hallar la ruta más corta La cantidad exacta de tiempo utilizado en un problema en particular, depende del orden en que se exploren las rutas

Generación y Prueba

El algoritmo de **generación y prueba** es un procedimiento de búsqueda primero en profundidad ya que las soluciones completas deben generarse antes de que se comprueban. De una forma más sistemática, es simplemente una búsqueda exhaustiva por el espacio problema El método de generación y prueba puede funcionar de forma que genere las soluciones de forma aleatoria, pero esto no garantiza que se pueda encontrar alguna vez la solución Esta forma de trabajar se conoce también como **algoritmo del Museo Británico**, en referencia a un método empleado para encontrar objetos en el museo, haciendo que éste se recorriera aleatoriamente.

La forma más sencilla de implementar una generación y prueba sistemática es mediante un árbol de búsqueda primero en profundidad con vuelta-atrás Sin embargo, si algunos estados intermedios aparecen con frecuencia en el árbol, puede resultar mejor modificar el procedimiento descrito antes, para que recorra un grafo en lugar de un árbol. Un **grafo** es un diagrama en que, mediante puntos unidos por líneas, se representa un sistema de correlaciones Normalmente estas líneas, rectas o curvas, tienen un sentido indicado por flechas.

Escalada

El método de la **escalada** es una variante del de generación y prueba; en él existe realimentación a partir del procedimiento de prueba que se usa para ayudar al generador por cuál dirección debe moverse en el espacio de búsqueda En un procedimiento de generación y prueba puro, la función de prueba responde sólo un sí o un no Pero si la función de prueba se amplía mediante una función heurística que proporcione una estimación de lo cercano que se encuentra el estado objetivo

Una variación útil del método de escalada simple consiste en considerar todos los posibles movimientos a partir del estado actual y elegir el mejor de ellos como nuevo estado Este método se denomina método de **escalada por la máxima pendiente (steep-est-ascent hill climbing)** o **búsqueda del gradiente (gradient search)**.

Tanto la escalada básica como la de máxima pendiente pueden no encontrar una solución. Cualquiera de los dos algoritmos puede acabar sin encontrar un estado objetivo, y en cambio encontrar un estado del que no sea posible generar nuevos estados mejores que él. Esto ocurre si el programa se topa con un máximo local, una meseta o una cresta

Un **máximo local** es un estado que es mejor que todos sus vecinos, pero que no es mejor que otros estados de otros lugares En un máxima local, todos los movimientos producen estados peores. Los máximos locales son particularmente frustrantes porque frecuentemente aparecen en las cercanías de una solución En este caso se denomina **estribaciones (foot-hills)**.

Una ***meseta*** **(pateau)** es una tarea plana del espacio de búsqueda en la que un conjunto de estados vecinos posee el mismo valor En una meseta no es posible determinar la mejor dirección a la cual moverse haciendo comparaciones locales.

Una *cresta* (**ridge**) es un tipo especial del máximo local Es un área del espacio de búsqueda más alta que las áreas circundantes y que además posee en ella misma una inclinación (la cual se podría escalar) Pero la orientación de esta región alta, comparada con el conjunto de movimientos disponibles y direcciones en la cual moverse, hace que sea imposible atravesar la cresta mediante movimientos simples

La escalada es un método local, lo que significa que decide cuál va a ser el siguiente movimiento atendiendo únicamente a las consecuencias "inmediatas" que va a tener esa elección, en lugar de explorar exhaustivamente todas las consecuencias

Enfriamiento simulado

El **enfriamiento simulado** es una variante de la escalada en la que, al comienzo del proceso, pueden realizarse algunos movimientos descendentes La idea consiste en realizar una exploración lo suficientemente amplia al principio, ya que la solución final es relativamente insensible con el estado inicial Así, se disminuye la probabilidad de caer en un máximo local, una meseta o una cresta

El objetivo de este proceso es alcanzar un estado final de mínima energía De esta forma el proceso consiste en el descenso de un valle en que la función objetivo es el nivel de energía Sin embargo, existe cierta probabilidad de que se produzcan transiciones hacia estados con energía más alta

Búsqueda el Primero Mejor

La búsqueda de **el primero mejor (best-first search)**, que representa una forma de combinar las ventajas que presentan tanto la búsqueda primero en anchura como la primero en profundidad en un solo método.

- **Grafos O**

En cada paso del proceso de búsqueda el primero mejor, se selecciona el nodo más prometedor que se haya generado hasta ese momento Esto se puede conseguir con una función heurística apropiada. Después se expande el nodo elegido aplicando las reglas para generar a sus sucesores Si alguno de ellos es una solución, el proceso termina Si no es así, estos nuevos nodos se añaden a la lista de nodos que se han generado hasta ese momento De nuevo, se selecciona el más prometedor de ellos y el proceso continua de la misma forma Lo normal es que la forma de funcionar se parezca un poco a la búsqueda primero en profundidad al explorar las ramas Sin embargo, si no se encuentra una solución, la rama empezará a parecer menos prometedora que otras por encima de ella y que se habían ignorado En este caso la rama que previamente se había ignorado aparece ahora como la más prometedora y, por lo tanto, comienza su explotación. Sin embargo, la vieja rama no se olvida. Su último nodo se almacena en el conjunto de nodos generados pero aún sin expandir. La búsqueda puede volver a él en el momento en que los otros sean lo suficientemente malos como para que éste sea de nuevo el camino más prometedor de todos

Este procedimiento es muy similar al de escalada por la máxima pendiente, excepto en dos aspectos El método de la escalada, al seleccionar un movimiento todos los demás se abandonan y nunca pueden volver a ser considerados Esta es la causa del característico comportamiento del método de escalada En el método de búsqueda de el primero mejor, se sigue seleccionando un movimiento, pero todos los demás se mantienen de forma que pueden visitarse si el camino que se ha seleccionado llega a ser menos prometedor Además de esto, en la búsqueda de el primero mejor se selecciona el mejor estado posible, aun si este estado tiene un valor menor que el del que se estaba explorando Esto contrasta con el método de la escalada en donde el proceso se detiene si no se encuentra un estado sucesor mejor que el estado actual

En ocasiones es importante realizar la búsqueda sobre un grafo de forma que los caminos duplicados no se exploren Tal algoritmo debe realizar una búsqueda en un grafo dirigido donde cada nodo representa un punto en el espacio de estados Cada nodo contiene además de una descripción de lo que representa en el espacio de estados, una indicación de lo prometedor que es, un enlace paterno que apunta al mejor nodo desde el que se ha generado el actual, y una lista de los nodos que se generan a partir de él El enlace paterno nos posibilita restablecer el camino hacia el objetivo una vez que se ha encontrado dicho objetivo La lista de sucesores hará posible, si se ha encontrado un camino mejor a un nodo ya existente, propagar la mejora a sus sucesores A los grafos de este tipo se les denomina **grafos O**, ya que cada una de sus ramas representan caminos alternativos para la resolución del problema

Para poder implementar un procedimiento de búsqueda sobre un grafo se necesitan dos listas de nodos

- **ABIERTOS**. Nodos que se han generado y a los que se les ha aplicado la función heurística pero que aún no han sido examinados (es decir, no se han generado sus sucesores). La lista ABIERTOS es, en realidad, un cola con prioridad en la que los elementos con mayor prioridad son aquellos que tienen un valor más prometedor de la función heurística Para manipular la lista pueden utilizarse las técnicas usuales de manipulación de colas de prioridad.
- **CERRADOS** Nodos que ya han sido examinados Es necesario mantener estos nodos en memoria si lo que se desea es hacer una búsqueda sobre un grafo y no sobre un árbol, debido a que cuando se genera un nuevo nodo, se debe verificar si ese nodo se había generado con anterioridad

También se necesita una función heurística que haga una estimación de los méritos de cada uno de los nodos que se van generando Esto permite que el algoritmo examine primero los caminos más prometedores.

Existe una variante de la búsqueda del primero mejor, denominada **búsqueda dirigida** (beam search), en la que solo son considerados los n estados más prometedores para futuras consideraciones Este procedimiento es más eficiente en términos de memoria, pero introduce la posibilidad de perder alguna solución si se poda el árbol demasiado pronto.

Funciona por pasos, expandiendo un nodo en cada paso hasta que se genere un nodo que se corresponde con un nodo objetivo En cada paso se toma el nodo más prometedor que se tenga en ese momento y que no se haya expandido Se generan los sucesores del nodo elegido, se les aplica la función heurística y se les añade a la lista de nodos abiertos Después de verificar si alguno de ellos se había generado con anterioridad Al realizar esta verificación, se puede garantizar que los nodos solo aparecen una vez en el grafo, aunque varios nodos pueden apuntar hacia él como su sucesor

- **Agendas**

Una **agenda** es una lista de las tareas que un sistema puede realizar Normalmente, a cada tarea se le asocian dos objetos una lista de las razones por las que se propone la tarea (con frecuencia denominadas **justificaciones**) y un valor que represente el peso total de la evidencia que dice que la tarea sería útil

Una importante cuestión que surge en los sistemas conducidos mediante agendas es la forma de encontrar la tarea más prometedora en cada ciclo Una de las formas de realizarlo es sencilla. mantener una agenda ordenada por valores, así, cuando se crea una nueva tarea, se inserta en la agenda en su lugar adecuado Cuando se cambian las justificaciones de una tarea, se recalcula su valor y se le traslada a su nueva posición en la lista Sin embargo, este método provoca el gasto de una gran cantidad de tiempo en mantener la agenda en perfecto orden, lo cual es innecesario Lo único que necesita saberse es cuál es el primer elemento correcto Cuando se propone una tarea o se añade una nueva justificación a una tarea existente, se calcula el nuevo valor y se compara con algunos de los más altos en la agenda (entre cinco y diez) Si es mejor, se inserta el nodo en su lugar adecuado en la cabeza de la lista En caso contrario, se deja donde estaba o simplemente se inserta al final de la agenda Además, de vez en cuando se debe recorrer la agenda y reordenarla adecuadamente

Una estructura de control conducida por agenda es también útil si algunas tareas (o nodos) proporcionan una evidencia negativa sobre los méritos de otras tareas (o nodos) Estas evidencias pueden representarse mediante justificaciones con pesos negativos. Si se utilizan estos pesos negativos, puede resultar importante verificar no sólo la posibilidad de trasladar una tarea de cabeza hacia abajo si aparecen nuevas justificaciones negativas

El mecanismo de agenda asume que si existe una buena razón para hacer algo en un cierto momento, entonces también existirá la misma buena razón para hacerlo más tarde, a no ser que surja algo mejor durante ese período de tiempo. Pero éste no es siempre el caso, en especial para aquellos sistemas que interactuan con las personas

Las estructuras de control dirigidas por agenda resultan de gran utilidad. Constituyen un mecanismo muy adecuado para integrar en un mismo programa información proveniente de distintas fuentes, puesto que cada una de ellas no hace más que añadir tareas y justificaciones a la agenda A medida que los programas de IA se van haciendo cada vez más complejos y va creciendo su base de conocimiento esto se convierte en una ventaja particularmente significativa

- **Grafos Y-O**

Otro tipo de estructura, el **grafo (o árbol) Y-O**, es útil para la representación de la solución en problemas que pueden resolverse descomponiéndolos en un conjunto de problemas más pequeños, cada uno de los cuales debe, a su vez, resolverse. Esta descomposición o reducción genera arcos denominados **arcos Y** Un arco Y puede apuntar a cualquier número de nodos sucesores, de forma que todos ellos deben resolverse para que el arco apunte a una solución Al igual que ocurre con un grafo O, pueden surgir varios arcos de un solo nodo, indicando los diversos caminos en los que se puede resolver el problema original Esta es la razón del porqué a este tipo de grafos se les denomina grafos Y-O en lugar de grafos Y

Para encontrar soluciones en un grafo Y-O, se necesita un algoritmo similar a una búsqueda de el primero mejor, pero con la capacidad de manipulación apropiada de los arcos Y Este algoritmo debería encontrar un camino a partir del nodo inicial del grafo hacia un conjunto de nodos que represente los estados solución Nótese que puede resultar necesario considerar más de un estado solución ya que cada brazo de un arco Y puede conducir a su propio nodo solución

Con el fin de describir un algoritmo de búsqueda sobre un grafo Y-O, es necesario utilizar un valor que de denomina **inutilidad**. Si el coste estimado de una solución se hace mayor que el valor de inutilidad, la búsqueda se abandona El valor de inutilidad debe elegirse de forma que se corresponda con un umbral tal que cualquier solución con un coste mayor sea demasiado cara para ser práctica, incluso en el caso de que puede encontrarse.

Verificación de Restricciones

Muchos de los problemas de IA pueden tratarse como problemas de **verificación de restricciones (constraint satisfaction)** donde el objetivo consiste en describir algún estado del problema que satisfaga un conjunto dado de restricciones

Al tratar un problema como una verificación de restricciones, es frecuentemente posible una reducción sustancial en la cantidad de búsquedas que se necesitan si se compara con un método que intente formar directamente soluciones parciales mediante la elección de valores específicos de una eventual solución.

La verificación de restricciones es un procedimiento de búsqueda que funciona en un espacio de conjuntos de restricciones El estado inicial contiene las restricciones que se dan originalmente en la descripción del problema Un estado objetivo es aquel que ha satisfecho las restricciones "suficientemente", donde "suficientemente" debe definirse para cada problema en particular

El proceso de verificación de restricciones consta de dos pasos. Primero se descubren las restricciones y se propagan tan lejos como sea posible a través del sistema. Entonces, si todavía no hay una solución, la búsqueda comienza. Se hace una suposición sobre algo y se añade como una nueva restricción Entonces, la propagación continua con esta nueva restricción, y así sucesivamente.

La propagación se hace necesaria por el hecho de que normalmente existen dependencias entre las restricciones Estas dependencias aparecen porque muchas restricciones hacen referencia a mas de un objeto, y muchos objetos participan en más de una restricción La propagación de restricciones también surge debido a la presencia de reglas de inferencia que permiten que se infieran restricciones adicionales a partir de las que se tenían La propagación de restricciones termina por una razón de entre dos posibles. La primera, porque se detecte una contradicción Si esto ocurre, entonces no existe una solución consistente con todas las restricciones conocidas. Si la contradicción se refiere únicamente a aquellas restricciones que se dan como parte de la especificación del problema entonces no existe solución. La segunda razón posible para que termine el proceso es que la propagación se realice de tal forma que pueden hacerse más cambios basándose en el conocimiento actual que se posea Si ocurre así, y todavía no se ha especificado adecuadamente una solución, entonces será necesario que se realice algún proceso de búsqueda para desbloquear el proceso

En este punto, comienza el segundo paso Para ver la forma de fortalecer las restricciones, pueden realizarse algunas hipótesis Una vez que se ha hecho esto, debe comenzar de nuevo la propagación de las restricciones a partir de este nuevo estado Si se encuentra una solución, se muestra. Si aún se necesitan más restricciones, se hacen. Si se detecta alguna contradicción puede usarse una vuelta-atrás para intentarlo con una suposición diferente y comenzar con ella

Este proceso merece que se hagan un par de observaciones Todo lo que se necesita de la propagación de restricciones es que no se produzca falsas restricciones. No es necesario que produzca todas las legales

Una segunda observación consiste en que normalmente existen dos tipos de restricciones Las primeras son **sencillas** son una lista de posibles valores para un objeto Las del segundo tipo son más **complejas** describen relaciones entre o en medio de objetos Los dos tipos de restricciones desempeñan el mismo papel en el proceso de verificación de restricciones.

Análisis de Medios y Fines

Las estrategias de búsqueda pueden razonar tanto hacia adelante como hacia atrás, pero para un problema dado, se debe elegir una dirección u otra. No obstante, a menudo es apropiado una mezcla de ambas direcciones Esta estrategia mixta haría posible la resolución, en primer lugar, de las principales partes del problema y, después, volver atrás y resolver los pequeños problemas que surjan al "pegar" juntos los trozos grandes Una técnica conocida como **análisis de medios y fines** (**means-ends analysis**) supone una ayuda para lograrlo

El proceso de análisis de medios y fines se centra en la detección de diferencias entre el estado actual y el estado objetivo. Una vez que se ha aislado una diferencia, debe encontrarse un operador que pueda reducirla. Es posible que tal operador no pueda aplicarse al estado actual por lo tanto, se crea el subproblema que consiste en alcanzar un estado en el que pueda aplicarse dicho operador Este tipo de encadenamiento hacia atrás, en donde se seleccionan los operadores y se producen subobjetivos para establecer las precondiciones del operador, recibe el nombre de **realización de subobjetivos para un operador**

Sin embargo, es posible que el operador no produzca exactamente el estado objetivo que se desea En este caso, se tiene un segundo subproblema que consiste en llegar desde ese estado hasta un objetivo Pero si se ha elegido correctamente la diferencia y el operador es realmente eficaz al reducir la diferencia, estos dos problemas serán más fáciles de resolver que el problema original El proceso de análisis de medios puede entonces aplicarse recursivamente

El análisis de medios y fines cuenta con un conjunto de reglas que pueden transformar un estado problema en otro Estas reglas normalmente no se presentan con las descripciones completas de los estados en cada uno de sus lados En lugar de esto, se presentan con un lado izquierdo que describe las condiciones que deben cumplirse para que pueda aplicarse la regla (a estas condiciones se les denomina **precondiciones de la regla**), y un lado derecho que describe aquellos aspectos del estado problema que cambiarán al aplicar la regla Existe una estructura de datos separada denominada **tabla de diferencias** que ordena las reglas atendiendo a las diferencias que pueden reducir

Aplicación

Realizar una aplicación que permita a un turista que visite el estado de Tabasco, encontrar la ruta para desplazarse de la capital del estado a alguna de las siguientes ciudades

Comalcalco
Emiliano Zapata
Frontera
Jalpa de Méndez
Jonuta
José Colomo
Mactun
Macuspana
Nacajuca
Paraíso
Tenosique

NOTA Ver código del programa en el Anexo 1 Búsqueda de Soluciones.

Capítulo

2

Sistemas Expertos

Introducción

En la década de los cincuenta surgió un interés especial por parte de los pedagogos y psicólogos, por encontrar los métodos generales de resolución de problemas, con el fin de que estos métodos se pudieran enseñar a los estudiantes y con ello se mejorara su preparación

Con la difusión de las primeras computadoras, en la segunda mitad de la década de los cincuenta, los estudios ya realizados en el campo de la resolución de problemas se intentaron de implementar a las computadoras

En la década de los sesenta, aparecen numerosos trabajos sobre el método general y universal de la resolución desarrollados sobre computadoras, de ellos el más famoso es "**Resolución General de Problemas**" de **Newell**, **Shaw** y **Simon**, Universidad de Carnege Mellon, **1957**.

Uno de los problemas que surgieron en aquel entonces fue, la aparición de la explosión combinatoria en los cálculos exhaustivos que limitaba la profundidad en los mismos y el número de conocimientos que se podían procesar, es decir, se calculaban todas las posibles soluciones para luego elegir la óptima. Aparecen entonces los primeros algoritmos de poda (**Algoritmo alfa-beta** de John McCarthy 1961).

Durante está década los planteamientos en el campo de la resolución de problemas cambian. No conociéndose los mecanismos generales de resolución de la mente humana, se pensó en simular los mismos para campos muy concretos del conocimiento El manejo eficaz de los conocimientos dio entonces sus primeros éxitos: los **Sistemas Expertos**.

El precursor de los **Sistemas Expertos** (**SE**) actuales es el sistema **DENTRAL** (Universidad de Stanford **1967**). Utilizaba para la representación del conocimiento las reglas de producción, su modo de funcionamiento se acercaba más a la filosofía de la resolución automática que a la de los Sistemas Expertos El sistema era capaz de determinar la estructura química de un compuesto orgánico a partir de los resultados obtenidos mediante un espectrógrafo de masas. Un desarrollo mejorado del mismo todavía se utiliza en la industria química

De los SE que se construyeron en esta época son famosos: el **PROSPECTOR** (Stanford Research Institute **1974**), sistema experto en prospecciones mineras especialmente las petrolíferas que cuenta entre sus méritos el descubrimiento en 1980, en el estado de Washington, de un importante yacimiento de molibdeno que fue posteriormente confirmado mediante prospecciones, y el **MYCIN** (Universidad de Stanford **1977**) sistema experto en el diagnóstico y terapia de enfermedades infecciosas de origen bacteriano que contaba con unas 400 reglas.

En la década de los ochenta, se iniciaron dos grandes líneas de investigación y de desarrollo, que son

- Divulgación o popularización de lo SE como una metodología que puede resolver de una forma adecuada múltiples problemas
- Generalización de lo SE que permitan ampliar el campo de conocimientos

En estos pocos años de vida de los SE surgió una nueva rama de la IA: **la Ingeniería del Conocimiento, del Saber** o **Cognoscitiva**, que es la parte de la IA que estudia los SE o basados en el conocimiento

Definición

Los SE se emplean para ejecutar una variedad muy complicada de tareas que anteriormente solo podían llevarse a cabo por un número limitado de personas expertas A través de la aplicación de las técnicas de la IA, los SE captan el conocimiento básico que permite a una persona desempeñarse como un experto frente a problemas complicados

La mayoría de los SE existentes nacen de la colaboración de **expertos humanos** e **ingenieros del conocimiento** El primero aporta el conocimiento en el área de interés y el segundo colabora poniendo ese conocimiento en forma tal que el sistema sea capaz de asimilarlo. No se trata de una etapa fácil, ya que experto humano e ingeniero hablan lenguajes distintos y la comunicación es difícil

Un experto o especialista humano es una persona que es competente en un área determinada del conocimiento o del saber. Algunas de las características generales y externas de los expertos humanos son

- Escaso número.
- Largo periodo de aprendizaje
- No están siempre disponibles, pues son humanos y cuando se jubilan o mueren se llevan con ellos todos sus conocimientos.

La forma más rápida de formar a un especialista es mediante el **aprendizaje formal** o ***académico*** (**"*conocimiento profundo*"**) en un principio y posteriormente un **aprendizaje *informal*** o práctico (**"conocimiento superficial"**)

Respecto a los SE no resulta fácil dar una definición, entre otras cosas, porque el concepto de SE va evolucionando, ya que, a medida que se va progresando, sus funciones se van ampliando y resulta un concepto cambiante

ESTA TESIS NO DEBE SALIR DE LA BIBLIOTECA

Algunas definiciones que se han dado de los SE son.

> "Un sistema experto, o sistema basado en el conocimiento, es un conjunto de programas de computadora que son capaces, mediante la aplicación de conocimientos, de resolver problemas en un área determinada del conocimiento o saber y que ordinariamente requerirían de la inteligencia humana." (1)

Edward Feigenbaum, de la Universidad de Stanford definió, en el Congreso Mundial de Inteligencia Artificial un sistema experto como

> "un programa de computadora inteligente que usa un conocimiento y procedimientos de inferencia para resolver problemas que son los suficientemente difíciles como para su resolución". Hoy, con los avances conseguidos, resultaría más correcto definir un sistema experto como: "un sistema informático que simula el proceso de aprendizaje, de memorización, de razonamiento, de comunicación y de acción de un experto humano en una determinada rama de la ciencia, suministrando, de esta forma, un consultor que puede sustituirle con unas ciertas garantías de éxito". (2)

Estas características le permiten almacenar datos y conocimiento, hacer conclusiones lógicas, tomar decisiones, aprender de la experiencia y de los datos existentes, comunicarse con otros especialistas humanos o SE, explicar el porqué de las decisiones tomadas y realizar acciones

Ventajas de los Sistemas Expertos

Mejoras en la productividad

La ventaja principal de los SE es que mejoran la productividad. En esencia, la **productividad** se refiere a la cantidad de trabajo que se realiza en un período de tiempo dado. Los SE pueden ayudar a mejorar la productividad porque ponen una gran cantidad de conocimientos valiosos a su alcance para poderlos aplicar cuando se necesiten, ayudando a realizar el trabajo más rápidamente o permitiendo desarrollar más trabajo en el mismo intervalo de tiempo.

Los SE también permiten ahorrar tiempo y dinero Con los conocimientos fácilmente accesibles, los problemas se pueden ir resolviendo a medida que se van planteando y las decisiones se pueden tomar rápidamente Además, los SE pueden ahorrar gastos ayudando a resolver los problemas en menos tiempo evitando costosos errores y decisiones inapropiadas. Los SE no sólo pueden mejorar la cantidad de trabajo, sino también su calidad

Conservación de conocimientos importantes

Los conocimientos de un especialista son muy importantes En la mayoría de los casos, un especialista ha invertido una gran cantidad de tiempo y trabajo para lograr sus conocimientos. Es necesaria una educación formal y años de experiencia para formar a un especialista Aunque es difícil valorar los conocimientos de un especialista, es evidente que tienen un valor, principalmente para las empresas

Los SE permiten conservar los valiosos conocimientos de un especialista. El especialista es la primera fuente para desarrollar un sistema experto que recoja sus conocimientos y, quizá, los de otros Aunque el especialista desaparezca, sus conocimientos se podrán utilizar si se han incorporado adecuadamente a un sistema experto Los resultados pueden ser una ventaja competitiva muy provechosa

Los SE proporcionan una nueva forma de conservar los conocimientos Los especialistas a menudo escriben libros que ayudan a difundir sus conocimientos Sin embargo, los libros suelen ser más teóricos que prácticos Los SE proporcionan un formato único para conservar ciertos tipos de conocimiento Para crear un sistema experto, se determina la esencia de los problemas que se pretenden resolver, se obtiene y programa el conocimiento necesario para su resolución El resultado es un software que resuelve directamente estos problemas

Mejora del aprendizaje y la comprensión

Los SE también sirven de ayuda en el momento de entender cómo un especialista resuelve un problema o cómo aplica sus conocimientos Para crear un sistema experto, el diseñador debe determinar exactamente qué conocimientos son necesarios y cómo se emplean. A menudo, los especialistas no saben exactamente cómo están resolviendo los problemas. Los especialistas dan por ciertos sus conocimientos y nunca analizan cómo los aplican. Pero para crear un SE, el diseñador debe descubrir esos detalles, lo que conduce a una mejor comprensión del razonamiento de la mente humana

Los SE también ayudan a mejorar la capacidad de aprendizaje Una persona que utiliza regularmente un SE para resolver problemas, se familiarizará bastante con la materia en cuestión Si se obtiene la suficiente experiencia usando el sistema experto, la capacidad del usuario se aproximará a la del especialista

Eliminación de operaciones incómodas y monótonas

Otro de los logros más importantes de los SE es el incremento de fiabilidad que se consigue gracias al tratamiento de datos computarizados y a veces a la intervención de grupos de expertos.

Disponibilidad

Están siempre disponibles a cualquier hora del día y de la noche, y de forma ininterrumpida.

Mantenimiento

Son fáciles de reprogramar por lo que pueden perdurar y crecer en el tiempo de forma indefinida

Clases de Sistemas Expertos

Las clases de Sistemas Expertos son

Sistemas independientes

Un **SE independiente** es aquel que puede funcionar por sí solo y ocupa la totalidad de los recursos de la computadora El programa se carga desde una unidad de disco duro o flexible y se ejecuta. La mayoría de los SE existentes son de este tipo

Sistemas híbridos

Los SE que se combinan o integran con rutinas algorítmicas se denominan **sistemas híbridos** Para realizar las funciones de resolución de problemas, el SE puede tener necesidad de realizar cálculos u otras tareas que se ejecutan mejor con rutinas algorítmicas. Los algoritmos están incorporados a los sistemas expertos, que se refieren a ellos cuando lo necesitan

También es posible incorporar un SE en un programa algorítmico convencional. Las funciones principales de tratamiento las realiza el programa convencional, con referencias ocasionales al SE cuando el proceso así lo requiera. Por ejemplo, se puede disponer de un SE que tome decisiones financieras cruciales a partir de los cálculos que realiza un programa de contabilidad general. El programa de contabilidad puede requerir la ayuda del programa de inferencia del SE cuando haya que tomar decisiones

Software encadenado

Otro tipo de SE mixto es el que encadena programas de carácter convencional y **SE múltiples** u otros programas de IA En general los SE necesitan obtener datos de alguna fuente para poder resolver los problemas o para poder tomar decisiones Muchos SE disponen de enlaces con otros paquetes de software, como hojas de cálculo o sistemas de bases de datos De esta forma, los SE pueden accesar a los datos que manejan estos programas

También es posible encadenar SE múltiples En los grandes sistemas informáticos que disponen de un mainframe conectado a muchas computadoras personales, pueden emplearse pequeños SE en las computadoras personales y otro SE (mucho mayor) relacionado en el mainframe. Estos SE encadenados pueden intercambiar información entre sí. Cada uno de los sistemas menores de las computadoras personales pueden tomar decisiones de carácter local, mientras que el sistema del mainframe puede resolver problemas mayores utilizando los sistemas de las computadoras personales como fuente de datos

Sistemas dedicados

Otro tipo de SE es el que está incorporado a un ordenador **cerrado** o **dedicado** Como en el caso de los sistemas independientes, se emplea un único ordenador para resolver el problema Los sistemas de control de armamento o de control de procesos industriales son ejemplos de SE dedicados

Sistema de tiempo real

Los programas de tiempo real están diseñados para responder rápidamente a las peticiones y realizar las operaciones necesarias de forma casi inmediata Un ejemplo es el sistema de reserva de plazas de una línea aérea, que permite a un agente de viajes determinar de forma inmediata el estado de un vuelo, el número de plazas disponibles, el número e importe de los descuentos aplicables y cualquier otro tipo de información Con la información el agente de viajes puede hacer una reservación y esta quedará automáticamente actualizada en el sistema El proceso se efectúa de forma tan rápida que el usuario no percibe el retraso (de ahí su nombre de **sistema de tiempo real**)

Un sistema de tiempo real en un avión de caza o bombardero puede ayudar a un piloto a tomar decisiones. El SE controla el radar, los sensores de infrarrojo y otros sistemas de detección para conocer la situación y decidir la acción apropiada Dado que el piloto necesita la información al instante, se requiere un sistema de tiempo real

Otro ejemplo de sistema de tiempo real son los cálculos implicados en un sistema de control Una computadora dedicada controla, por ejemplo, una fábrica química, típicamente necesita funcionar en tiempo real para obtener resultados apropiados La computadora puede detectar *distintos parámetros, como temperatura, presión, velocidad de flujo, etc. La computadora utiliza* estas señales para determinar cómo se ha de modificar el sistema. Las modificaciones que introduce la computadora dan lugar a nuevas señales de control Por ejemplo, si la temperatura es muy alta, los sensores detectan el hecho y la computadora producirá la señal de control necesaria para ajustar la temperatura al nivel deseado Funcionando en tiempo real, una computadora debe tomar la información de los sensores y procesarla lo suficientemente rápido para que las señales de control se generen oportunamente para ser útiles.

Arquitectura de los Sistemas Expertos

Los elementos que conforman un SE son una base de conocimientos, una motor de inferencia, una base de datos y una interfaz con el usuario, pero otras características varían según las aplicaciones

La base de conocimientos

La **base de conocimientos** contiene información específica de un campo o especialidad. La mayoría de los SE utilizan reglas de producción, por lo que la base de conocimientos se denomina con frecuencia **base de reglas** Algunos SE utilizan también redes semánticas y marcos de referencia.

Una característica clave de los SE es que su base de conocimientos es independiente del sistema de inferencia que se emplea para resolver los problemas

La base de conocimientos tiene una estructura acorde con la naturaleza del conocimiento Este formato es conocido por el programa de inferencia, de modo que puede acceder a las partes **si** y **entonces** de las reglas de forma independiente, según lo requiera la estrategia de resolución de problemas

Dado que la base de conocimiento esta separada de los programas algorítmicos del motor de inferencia, a medida que se dispone de nuevos conocimientos o cuando los existentes quedan obsoletos es relativamente fácil efectuar los cambios necesarios Lo único que hay que hacer es añadir nuevas reglas, eliminar las viejas o corregir las existentes, no es necesario reprogramar el SE en su totalidad

No existen normas específicas para almacenar reglas en una base de conocimientos. De hecho, dado que el motor de inferencia busca en la base de reglas, las reglas pueden almacenarse virtualmente en cualquier orden Aunque las reglas se almacenan generalmente en alguna secuencia jerarquica logica, esto no es necesario Se pueden incluir las reglas en cualquier secuencia y el motor de inferencia las utilizará en el orden adecuado para resolver un problema.

El motor de inferencia

El **motor de inferencia** es el sistema de software que ubica los conocimientos e infiere nuevos usando la base de conocimientos. También denominado "**programa de control**" o "**interprete de reglas** ", el programa funciona con los datos suministrados por el usuario para recorrer la base de conocimientos hasta alcanzar una conclusión Las estrategias de control que se implementan pueden estar basadas en el uso de búsquedas con encadenamiento progresivo, regresivo, en profundidad, en anchura o en una mezcla de éstas. Las funciones de comparación de patrones del motor de inferencia comparan los datos de entrada con la parte **si o entonces** de las reglas de la base de conocimientos Cuando se produce una coincidencia una parte del proceso de razonamiento queda completa. El motor de inferencia continúa su búsqueda hasta que se alcanza una solución.

El motor de inferencia controla todo el proceso Sus dos funciones básicas son **inferencia** y **control** La inferencia se refiere al examen de las reglas y a la ejecución de los programas de comparación de patrones, mientras que el control se refiere a la secuencia en que se examinan las reglas El motor de inferencia pregunta al usuario los datos iniciales y entonces procede a buscar en la base de conocimientos las reglas que coinciden con los datos de entrada. El intérprete de reglas determina la secuencia en que se examinan éstas, y solicita información adicional de entrada si no puede tomar una decisión basada en los hechos y reglas disponibles El motor de inferencia automatiza este proceso y es completamente invisible cuando funciona en respuesta a una consulta

Encadenamiento regresivo

La mayoría de los SE emplean técnicas de **encadenamiento regresivo** en los procesos de inferencia Cuando se emplea encadenamiento regresivo, el SE intenta probar una hipótesis. En el encadenamiento regresivo el motor de inferencia analiza la parte **entonces** de la regla y después intenta probar la parte **si** Así, se recorre la base de conocimientos en busca de reglas que concluyan esa parte de **si** Pero si ninguna es posible, entonces se solicita al usuario que proporcione la *información necesaria. Si la hipótesis necesaria no está disponible, el motor de inferencia busca* otras reglas o concluye que la hipótesis que está intentando verificar no puede probarse, entonces intenta probar la siguiente hipótesis y luego las demás de la secuencia hasta que consigue probar alguna Por supuesto, si no se dispone de suficientes datos de entrada, el sistema no podrá ofrecer una conclusión

Encadenamiento progresivo

Si se implementa una secuencia de control de **encadenamiento progresivo**, el motor de inferencia parte de los hechos disponibles en la base de datos y busca los hechos que se encuentran en la parte **si** de las reglas Si la parte **si** de la regla coincide con un hecho en la base de datos, la regla se satisface La parte **entonces** de la regla se considera cierta y el nuevo hecho inferido se almacena en la base de datos Con esta nueva información, el motor de inferencia intenta encontrar este hecho recién inferido en la parte **si** de otra regla En este momento, el SE proporciona su respuesta

Interfaz con el usuario

La **interfaz con el usuario** es una colección de programas que funciona dentro del motor de inferencia y de la base de conocimientos a fin de proporcionar los medios adecuados para mantener una comunicación en dos sentidos, entre el software y el usuario El SE puede hacer preguntas que el usuario contesta tecleando las respuestas Algunas interfaces de usuario funcionan mediante menús, que plantean preguntas de respuesta múltiple, pidiendo al usuario que elija la respuesta correcta entre las posibilidades expuestas Esto facilita una entrada de datos rápida y sencilla

La interfaz con el usuario también entra en juego durante el proceso de inferencia. En caso de que el intérprete de reglas no pueda alcanzar una decisión por falta de información, lo notificará al usuario Esto se puede hacer tomando una parte de la regla con la que se está trabajando y convirtiéndola en una pregunta Cuando el usuario proporciona los datos necesarios, el motor de inferencia puede continuar el proceso de razonamiento.

Cuando se termina el proceso de razonamiento, se presenta el resultado Los SE no siempre pueden alcanzar una respuesta correcta, de modo que en la pantalla pueden no aparecer conclusiones o sólo se presentará la estimación o aproximación alcanzada. El resultado también puede ser una respuesta acompañada de un factor de certidumbre que la califique.

Bases de datos

La **base de datos**, denominada en ocasiones **base de datos global** o **memoria de trabajo**, es la parte de la memoria de la computadora que se emplea para mantener un registro de los datos recibidos, conclusiones intermedias y datos generados. El motor de inferencia utiliza la base de datos como un bloc de notas para apuntar lo que sucede en el sistema

Los datos iniciales se almacenan en la base de datos A medida que el intérprete va examinando las reglas, se van almacenando las conclusiones derivadas de éstas en la base de datos. El motor de inferencia utiliza estas conclusiones intermedias como nuevos datos para buscar nuevos patrones Al final de un trabajo, la base de datos contiene la cadena de hechos completa, que incluye no sólo los iniciales, sino también los datos alcanzados en el camino hacia la decisión final

Subsistema de explicación

Muchos SE contienen una sección diseñada para explicar al usuario la línea de razonamiento seguida para alcanzar una conclusión

La explicación consiste en una identificación de los pasos en el proceso de razonamiento y de una justificación de cada uno de ellos. Con un **subsistema de explicación**, los usuarios pueden hacer preguntas del tipo ¿por qué? o ¿cómo?, y el sistema proporcionará la respuesta adecuada Por ejemplo, si el sistema solicita datos adicionales, el usuario puede querer saber por qué. Normalmente, el sistema responderá que necesita esa información para evaluar una regla determinada e incluso puede indicar la regla que ésta intentando resolver

El subsistema de explicación es un excelente instrumento para propósitos educativos. La mayoría de los SE proporcionan una respuesta rápida a un problema, pero en general no suelen dar explicaciones a menos que se les pidan Resolviendo un conjunto de problemas y al mismo tiempo solicitando una explicación del proceso seguido, los usuarios pueden comprender el proceso de razonamiento

Los usuarios pueden utilizar el subsistema de explicación para propósitos de depuración Durante el desarrollo, éste puede emplearse como una forma de conseguir información retrospectiva acerca de la construcción y secuencia de las reglas, permitiendo a los usuarios comprobar el sistema de forma rápida en el caso de problemas prácticos

Esta posibilidad de explicación de los SE es una característica valiosa cuando hay que tomar decisiones críticas Si los usuarios pueden cuestionar las decisiones del sistema y explorar las alternativas posibles, están utilizando sus propios conocimientos para resolver el problema, y el hecho de tomar una decisión se transforma en una decisión conjunta

Pero no todas las aplicaciones requieren un subsistema de explicación y no todas garantizan la posibilidad de disponer de uno Por ejemplo, un piloto de combate que utilice un SE para tomar decisiones críticas en la ejecución de maniobras o en la elección de armas no tiene tiempo para cuestionar o analizar la decisión.

Subsistema de adquisición de conocimientos

Casi todos los SE contienen un programa o un conjunto de programas que permite a los usuarios añadir los datos o modificar la base de reglas Muchos campos en los que se emplean los SE son muy dinámicos. Es decir, el conocimientos está cambiando constantemente, por lo que es necesario modificar la base de conocimientos para reflejar esos cambios Los subsistemas de adquisición de conocimientos proporcionan los medios adecuados para añadir nuevas reglas y editar las ya existentes

El proceso para adquirir nuevo conocimiento puede tomar una de tres formas fundamentales

La primera forma es la **actualización manual de conocimientos** En este caso la actualización se lleva a cabo por un ingeniero en conocimientos, quien interpreta la información ofrecida por un experto en el área y actualiza la base de conocimientos mediante el uso de un sistema limitado de actualización.

La segunda forma, el experto en el área **ingresa directamente el conocimiento** realizado sin la mediación de un ingeniero en conocimientos, en este caso el sistema de actualización de conocimientos debe ser mucho más elaborado

En la tercera forma, **aprendizaje mecánico**, el sistema genera nuevos conocimientos en forma automatica y se basa en generalizaciones deducidas de experiencias anteriores.

Tipos de Sistemas Expertos

A continuación se examinan algunos tipos de sistemas expertos.

Sistemas de análisis e interpretación

Los SE son muy eficaces para analizar una gran cantidad de información, interpretarla y proporcionar un resumen o una recomendación a partir de la misma. Cuando existen grandes cantidades de información relacionada con un problema, es difícil para los humanos recordarla y utilizarla Los SE pueden realizar un análisis más detallado que los humanos a menudo proporcionando unas recomendaciones mejores o una mejor comprensión de la situación

Los SE realizan un trabajo de interpretación a partir de la información recibida desde un teclado, desde otro programa de la computadora o desde sensores electrónicos. Una vez que la información esta disponible, los programas de inferencia utilizan esos datos junto con la información contenida en una base de conocimientos para intentar comprender los datos, y a continuación proporciona una explicación para los datos u obtiene conclusiones apropiadas.

Un ejemplo de sistema de análisis e interpretación es el de planificación financiera, que recibe como entrada todos los datos financieros acerca de un cliente junto con sus propósitos. El sistema analiza todas las fuentes de datos y las oportunidades de inversión. Las posibles opciones de inversión son normalmente muy grandes y es difícil, además de lento, para una persona obtener un plan de acción adecuado Las organizaciones de planificación financiera utilizan computadoras para ayudar a sus clientes a tomar decisiones acertadas

Sistemas de predicción

Los SE son muy buenos para predecir resultados futuros Utilizando los datos de entrada acerca de una situación dada, un SE de predicción puede deducir consecuencias futuras o resultados a partir del conocimiento que tiene Los **sistemas de predicción** producen resultados especialmente buenos a la hora de sugerir las consecuencias más probables de unas determinadas condiciones La base de conocimientos suele contener las tendencias de los datos y la información histórica, así como los patrones cíclicos apropiados Aplicando estos conocimientos a los datos de entrada, pueden predecirse fiablemente los hechos más probables Los sistemas meteorológicos de inversión en bolsa son buenos ejemplos de sistemas de predicción

PROSPECTOR es un ejemplo de sistema de predicción Esta diseñado para ayudar a los geólogos a localizar depósitos de mineral Contiene reglas acerca de los depósitos minerales, de las rocas y del tipo de mineral Cuando se le proporcionan datos acerca de una determinada zona geológica puede predecir la probabilidad de distintos tipos de minerales presentes en el lugar de sondeo

Sistemas de diagnóstico y depuración

Ambas técnicas se utilizan en la reparación y detección de fallas en equipos. También se emplean mucho en medicina de diagnóstico

Un SE de **diagnóstico** reciben los datos acerca del comportamiento del dispositivo, sistema o individuo De hecho, la mayoría de los SE formulan listas de preguntas en un intento de acumular los datos necesarios para llegar a una conclusión Estos datos de entrada toman la forma de síntomas, características físicas. prestaciones registradas e irregularidades o funciones indeseadas. Con esta información, el programa de inferencia recorre la base de conocimientos para detectar problemas.

Mientras que algunos SE realizan sólo diagnósticos, otros también incluyen características de depuración, lo que significa que recomiendan las acciones adecuadas para corregir los problemas y deficiencias descubiertos

Uno de los usos más comunes de los SE es el **diagnóstico y depuración** de los sistemas complejos Es muy difícil para un individuo localizar un problema en sistemas tan grandes y complejos, pero con la ayuda de un SE los problemas se pueden localizar y resolver de forma más rápida

Uno de estos sistemas es el **Automated Cable Expertise (ACE)**, desarrollado en los laboratorios Bell de New Jersey. Se utiliza para ayudar a detectar defectos en las redes telefónicas de mantenimiento, y los datos de las reparaciones determinan la situación de las fallas de los cables y sugiere las acciones de mantenimiento que deben realizarse.

Existen muchos SE dedicados en medicina, cada uno especializado en una determinada enfermedad o afección, que pueden ayudar a los médicos a realizar un diagnóstico más rápido en el tratamiento de sus pacientes Un ejemplo es **PUFF**, un sistema de ayuda a los médicos a diagnosticar las enfermedades de pulmón Se introduce el historial clínico y los datos de análisis específicos y PUFF produce un informe acerca de los posibles problemas pulmonares del paciente y de su tratamiento.

Sistemas de control

Las computadoras se usan frecuentemente para controlar un proceso y tomar las acciones apropiadas en respuesta a su estado Las computadoras que controlan las fábricas automatizadas o factorías químicas son un ejemplo Otra buena aplicación para un SE es el sistema de control de una central nuclear

El control consiste fundamentalmente en observar los datos que proporcionan los sensores Se emplea una gran variedad de sensores para transformar las variaciones de distintas magnitudes físicas en señales eléctricas que pueden ser interpretadas por un ordenador Se prepara un programa para determinar cuáles de las señales controladas se encuentran dentro de los límites deseados En un sistema experto, la base de conocimientos contiene reglas para decidir qué hacer con las magnitudes con valores fuera de límite, se activa un programa de control en la computadora para devolver los datos a su nivel correcto

Este proceso de control generalmente emplea técnicas de retroalimentación, es decir, información detectada en el sistema que se está controlando, y se informa a la computadora lo que sucede. La computadora continúa controlando el sistema hasta que recibe información de que el sistema falla o llega a alguna condición deseada. A este proceso se le conoce como **sistemas de control** en bucle cerrado, éstos permiten automatizar totalmente algunos procesos Las señales de salida de la computadora controlan el sistema mientras los sensores determinan su estado. Entre tanto, la computadora realiza las operaciones necesarias para controlar el sistema

Todavía existen pocos sistemas de tiempo real en este campo, pero hay un prototipo de demostración denominado **FALCON** que se desarrolló para probar este concepto FALCON controla los sensores, dispositivos de medición y otros datos de entrada de una factoría química. Analiza los datos e intenta identificar las causas de los problemas que pueden resolver los operarios de factoría.

Sistemas de enseñanza

Un SE puede utilizarse para la enseñanza empleando el conocimiento que forma el núcleo del sistema

En la actualidad hay muchos sistemas de instrucción asistida por computadora Sin embargo, estos programas tienen un contenido y una secuencia fijos que no es lo más apropiado para el estilo de aprendizaje de todos los estudiantes Un SE puede mejorar esta situación.

Para lograr este propósito, un SE evalúa el nivel de conocimientos y comprensión de cada estudiante Con esta información se puede ajustar el proceso educativo a sus necesidades

SOPHIE es un prototipo de investigación de un sistema educativo Fué diseñado para enseñar a los técnicos cómo detectar y reparar las averías de los circuitos electrónicos SOPHIE simula el funcionamiento de un circuito electrónico y permite introducir fallos en su comportamiento para su análisis.

Aplicaciones de los Sistemas Expertos

La mayoría de los SE que se encuentran en fase de desarrollo o los ya existentes se emplean en alguno de los siguientes campos finanzas, ingeniería e investigación, fabricación, informática, fines gubernamentales o fines militares

Los SE en las finanzas

Se necesita un especialista en finanzas para analizar y aplicar la información disponible, y es aquí donde se pueden aplicar los sistemas expertos Utilizando sistemas expertos, las computadoras pueden analizar los datos financieros y dar respuestas a las preguntas, soluciones a los problemas o recomendaciones para tomar una decisión.

Las aplicaciones bancarias

La mayoría de los bancos disponen de un criterio estándar para evaluar las operaciones de préstamo Un asesor recoge la información necesaria y, basándose en el conocimiento del prestatario y en las normas generales del banco y límites de riesgo, decide conceder o denegar el crédito. Estos conocimientos se pueden incorporar a un sistema experto, de modo que la decisión no tenga por qué tomarla un asesor, sino otras personas con menos experiencia para tomar decisiones de este tipo

Impuestos

Actualmente se dispone de SE que pueden ayudar a las personas y a las empresas a tomar las mejores decisiones para reducir los impuestos. Dada la complejidad del sistema fiscal, se necesita un especialista para que sugiera las mejores estrategias posibles

La experiencia de especialistas fiscales y de auditores se está agrupando de forma que otras personas con menos experiencia y conocimientos puedan aconsejar a los clientes acerca de sus mejores estrategias fiscales

Bolsas de valores

Se están desarrollando SE para ayudar a los agentes de bolsa a recomendar las operaciones más apropiadas y para la compra y venta de acciones Los SE pueden ayudar en el análisis de las compañías, mercados y la economía empleando conocimientos de carácter general o heurístico Esto puede ayudar a mejorar la calidad de las recomendaciones que los agentes de bolsa hacen a sus clientes

Planificación financiera

La planificación financiera consiste en el análisis de la situación económica de una persona o empresa y supone una gran variedad de consideraciones, incluyendo impuestos e inversiones inmobiliarias.

Conociendo sus propósitos financieros, los SE pueden formular planes específicos a fin de alcanzar estos objetivos dentro de las restricciones impuestas por los recursos financieros y por las condiciones externas

Los SE de planificación financiera combinan los conocimientos de los asesores financieros, los asesores fiscales, los especialistas en seguros, los agentes inmobiliarios y los economistas, entre otros

Sistemas de gestión financiera

Se están desarrollando SE para ayudar a los ejecutivos a tomar las mejores decisiones en la gestión de sus compañías Estos SE realizan una gran variedad de análisis financieros y proporcionan recomendaciones para la gestión

Otros SE pueden emplearse para evaluar los planes financieros. Se puede analizar un plan para una nueva compañía producto o mercado, identificar los problemas potenciales y hacer recomendaciones apropiadas Dichos SE pueden ayudar en la planificación del capital, en las fusiones y en las adquisiciones Estos SE pueden tratar los conocimientos acerca del estado de la industria, las opiniones de los clientes, éxitos y fracasos en operaciones similares en el pasado

Las compañías de seguros

Un agente de seguros experto puede ayudar a una persona a calcular el riesgo que asume al adoptar una determinada política de seguros Otro SE empleado en seguros permite a las compañías tener la certeza de que sus clientes han llenado las pólizas correctamente.

Los SE en la ciencia y la ingeniería

Los ingenieros utilizan sistemas basados en el conocimiento para realizar diseños asistidos por computadora (CAD) para crear nuevos productos Los ingenieros y los técnicos están creando SE que ayuden a diagnosticar y resolver problemas en sistemas complejos.

Los SE de reparación y mantenimiento

Ejemplos de estos sistemas son las computadoras de gran tamaño, los sistemas telefónicos y las redes de comunicación de datos Los militares también emplean SE como ayuda en la tarea de corregir fallas en los sistemas de armamento Por ejemplo, la marina de los Estados Unidos utiliza SE para localizar fallas en los sistemas de radar

La Ford emplea un SE como ayuda en el diagnóstico de los problemas en las herramientas mecanizadas y en los robots de producción Cuando una de estas máquinas sufre una avería, puede parar toda la cadena de producción. Un SE puede ayudar a mantener en marcha las líneas de producción

La General Motors emplea SE en el diagnóstico y reparación de sistemas eléctricos o de las computadoras que utilizan la mayoría de los automóviles modernos.

Muchos de los fabricantes de computadoras utilizan SE de diagnóstico en la fabricación y en los procesos de mantenimiento IBM utiliza un SE como ayuda para localizar problemas en las unidades de disco durante las últimas fases de la fabricación Con el uso de los SE los técnicos pueden reparar estas unidades defectuosas de forma rápida con el mínimo desperdicio de piezas de repuesto

La General Electric Corporation diseñó y construyó uno de los primeros sistemas basados en el conocimiento para detectar fallas El sistema se diseñó para mejorar el mantenimiento de las locomotoras diesel y eléctricas

La General Electric también desarrolló un SE como ayuda de mantenimiento y búsqueda de fallas de los motores de los aviones a reacción Dicho sistema permitía a los mecánicos menos cualificados diagnosticar los problemas de las máquinas, así como realizar reparaciones.

Los SE en medicina

En el campo médico se hace más uso de los SE que en muchos otros campos Se han diseñado programas para ayudar a los médicos a diagnosticar una enfermedad particular y, en algunos casos, a prescribir un tratamiento

El más antiguo de los sistemas médicos es **MYCIN** Muchos sistemas han surgido a partir de MYCIN y el propio trabajo ha animado a otros a crear sistemas parecidos. En la siguiente tabla se presentan algunos de los SE de los que se dispone hoy en día en el campo de la medicina

Sistema	Propósito
ONCOCIN	Ayuda a los médicos a tratar con quimioterapia a los enfermos de cáncer, eligiendo el tratamiento adecuado según el diagnóstico y la historia clínica del paciente.
PUFF	Ayuda a identificar la enfermedades de pulmón
CASNET/GLAUCOMA	Ayuda a diagnosticar el glucoma y otras enfermedades oculares, y a prescribir un tratamiento.
GUIDON	Sistema educacional para enseñar a elegir una terapia antimicrobiana para las infecciones bacterianas.
Drug Interaction Critic	Sistema de consulta que ayuda a los médicos a elegir los medicamentos que van a recetar cuando ya se están empleando otros, para evitar incompatibilidades o efectos secundarios.

La principal ventaja de los SE en medicina es que pueden ayudar a los médicos a diagnosticar con rapidez enfermedades complicadas Dado que existen tantas enfermedades diferentes, es difícil para un médico identificarlas todas. Otro beneficio es que los SE también permiten a los médicos confirmar una sospecha o explorar un diagnóstico alternativo en los casos difíciles

Los SE en las fábricas

El uso de SE para mejorar los sistemas de fabricación comienza en el diseño y montaje de cadenas de producción que optimicen el rendimiento. Los especialistas en operaciones de fabricación pueden aplicar sus conocimientos en estos temas a medida que se construyen las nuevas fábricas o se realizan mejoras en las líneas de producción existentes

En las operaciones de fabricación ya en curso, los SE optimizan el proceso de producción ayudando a identificar los parámetros críticos para que reciban la atención adecuada Los SE ayudan a los ingenieros de planta a evaluar las operaciones alternativas y los procesos que pueden mejorar los planes de producción Dichos sistemas pueden ayudar a los jefes de producción a comprender mejor el impacto de factores como tiempo de espera, gestión de materiales y control de inventarios.

La mayoría de los sistema expertos de fabricación son, sin embargo, de carácter más especializado Enfocan un problema de productividad específico de una fábrica y de un producto. Por ejemplo, se han desarrollado varios SE como ayuda en el montaje de placas de circuitos impresos. Uno de estos sistemas ayuda a decidir la secuencia de pasos a realizar manualmente en placas complejas de gran tamaño El sistema clasifica la información de diseño y las especificaciones y presenta a los trabajadores de ensamblado una secuencia de acciones paso a paso Se supone que un SE de este tipo puede reducir en gran medida los porcentajes de defectos y, por tanto, incrementar simultaneamente la cantidad y la calidad de la producción

Aplicaciones gubernamentales y militares

Las fuerzas aéreas

SE para asistencias al personal de vuelo, reconocimiento de blancos, fabricación y diseño.

Los sistemas de asistencia al personal de vuelo ayudan a los pilotos a realizar su trabajo de forma más eficiente. Uno de los programas, denominado "**socio del piloto**", ayuda a los pilotos a planificar sus misiones y los alerta de los peligros de combate. También controla los subsistemas del avión para detectar problemas e informar de ellos al piloto

Las fuerza aéreas están utilizando ya SE para el mantenimiento de los aviones de combate. La mayoría de estos sistemas funcionan en computadoras personales y están diseñados para diagnosticar problemas motores y aerodinámica de los aviones Dichos sistemas ayudan al personal de tierra a localizar y reparar rápidamente los problemas que puedan presentarse Estos sistemas obtienen sus datos directamente de los sensores electrónicos instalados en los motores o en el equipo aerodinámico del fuselaje Estos datos pasan mediante un cable hasta la computadora en la que funciona el sistema experto El sistema diagnostica entonces el problema.

El ejército

El ejército de tierra está utilizando SE para detectar problemas en sus equipos Los SE actuales funcionan sobre armas, equipos de transporte y sistemas de comunicaciones Un ejemplo de SE de este tipo es el diseñado para detectar problemas en los carros blindados **M-1** Este carro de combate de alto nivel tecnológico tiene docenas de subsistemas y miles de conexiones eléctricas

La marina

La marina está empleando SE para el diagnóstico y reparación de armas y sistemas de comunicaciones La marina está empezando también a emplear SE en el análisis de presupuestos

Aplicación

Realizar un Sistema Experto que le sugiera a un turista los diferentes lugares de la República Mexicana que puede visitar de acuerdo a alguna de las siguientes opciones

Playa
Colonial
Zona Arqueológica

Además de proporcionar información adicional de algún estado

NOTA Ver código del programa en el Anexo 2 Sistemas Expertos

Capítulo

3

Procesamiento de Lenguaje Natural

Introducción

Se denominan **lenguajes naturales** aquellos que sirven a las personas para comunicarse tanto de forma oral como por escrito. Ejemplos de estos lenguajes son el inglés, francés, el castellano o el chino Por el contrario, el **lenguaje de las computadoras** puede parecer difícil de entender para los humanos, esto es debido a que estos **lenguajes artificiales** estan diseñados como sentencias con un formato rigido que facilitan a los compiladores la comprensión de los programas y los convierten en secuencias de instrucciones para la computadora Además, por el hecho de ser estructuralmente más simples, los lenguajes de programación únicamente puede expresar los conceptos importantes de programación del tipo "si ocurre A entonces haz B".

La investigación de la utilización de las computadoras en el estudio del lenguaje, comenzó poco después de que las computadoras surgieran como máquinas reales que resolvían problemas fácilmente, esto sucedió al rededor de **1940** La capacidad de una computadora para usar y manipular todo tipo de símbolos fue rápidamente aplicada en los textos escritos para recopilar índices de palabras básicamente estadísticas de aparición de palabras en un texto, frecuencia de la letras, utilización de los signos de puntuación, etc

En **1949**, **Warren Weaver** propuso utilizar las computadoras para resolver problemas de traducción entre lenguajes naturales El resultado de esta investigación, que se **denomino traducción automática**, intentaba simular las funciones que realizaría el traductor humano: consultar cada palabra en un diccionario bilingüe elegir la palabra adecuada para el lenguaje de salida y, realizando esto para todas y cada una de las palabras de una frase, colocarlas en un orden adecuado para su comprensión.

Aunque en un principio puede parecer una idea sencilla, aparecen problemas imprevistos, que normalmente son: ¿Qué palabra del lenguaje de salida, de todas las posibles a escoger como traduccion es la correcta?, ¿Cuál es el orden que debe ocupar dentro de una frase una palabra determinada?

Estos fueron los motivos fundamentales por los que se abandono esta enfoque. Surgen en este momento, un nuevo foco de la IA. **la comprensión**, si la computadora pudiera comprender el sentido de una frase, sería capaz, posiblemente, de contestar preguntas que se cuestionarían sobre esta frase Sin embargo, este enfoque no queda libre de problemas, la naturaleza de la comprensión en si misma es un problema muy difícil.

Hacia **1960**, el procesamiento del lenguaje natural es influido por los múltiples desarrollos científicos de estos años, incluidos los lenguajes de programación de alto nivel En esa época, los investigadores de IA desarrollaron un nuevo grupo de programas, procurando dar con las causas que impiden la traducción automatica Estos investigadores comienzan a tratar el lenguaje natural como una compleja habilidad del conocimiento, el cual se estructura en diferentes partes se trata por un lado la estructura de las frases en el lenguaje (qué parte debe ocupar cada palabra, qué partes debe tener una frase), el sentido de las palabras (qué concepto describe un palabra, qué función cumplen en una frase), un modelo del oyente (qué acciones debe segur la persona que recoge las frases del lenguaje natural), las reglas de una conversación (no hablar cuando se escucha, ser claro en el mensaje a transmitir) y, por último, información adicional sobre cada palabra, formando una estructura

Definición

El **procesamiento de lenguaje natural**, abreviado normalmente como **PLN**, intenta hacer a la computadora capaz entender órdenes escritas en lenguaje humano estándar

La tarea del entendimiento del lenguaje computarizado es muy difícil por el hecho de que las más simples ideas pueden ser parafraseadas en varios caminos

Algunos de los problemas que se presentan son los siguientes

1 Ambiguedad

- **Ambigüedad referencial** Cuando la gente habla y escribe, usa pronombres para evitar repeticiones de sustantivos. Los pronombres pueden crear confusión, de cualquier modo
- **Ambigüedad estructural** Las palabras en una oración pueden ser agrupadas en diferentes caminos para cambiar el significado de la idea

2 Entendimiento de párrafos

Existen mucho caminos para expresar la misma idea.

3 Complejidad

Uno de los aspectos más difíciles es adaptar la complejidad y flexibilidad del lenguaje natural

Proceso de Comprensión del Lenguaje Natural

Los componentes del proceso se dividen en .

Análisis morfológico Se analizan los componentes de las palabras individuales, y se separan de las palabras los constituyentes que no forman parte de ellas, como los símbolos de puntuación.

Análisis sintáctico Se transforman secuencias lineales de palabras en estructuras que muestran la forma en que se relacionan las palabras entre sí. Pueden rechazarse secuencias de palabras si infringen las reglas del lenguaje mediante las cuales puedan combinarse esas palabras

Análisis semántico. Se asignan significados a las estructuras creadas por el analizador sintáctico En otras palabras, se realiza una correspondencia entre las estructuras sintácticas y los objetos del dominio de la tarea Se rechazan aquellos objetos para los cuales no es posible una tal correspondencia (**semántica anómala**)

Integración del discurso El significado de una frase individual puede depender de las frases procedentes y pude influenciar el significado de las frases posteriores

Análisis pragmático. Se reinterpreta la estructura que representa lo que se dijo para determinar lo que significa realmente

En ocasiones las fases se desarrollan en secuencia, mientras que otras veces se realizan a la vez Si se realizan en secuencia, una puede pedir ayuda a las demás Un procesador sintáctico por sí mismo no es capaz de elegir entre estas opciones, por lo que la decisión debe realizarse teniendo en cuenta algún modelo del mundo en el que algunas frases tienen sentido y otras no

Análisis Morfológico

El proceso normalmente asignará categorías sintácticas a todas las palabras de la oración Esto se suele hacer en este momento porque las interpretaciones de los afijos (prefijos y sufijos) suelen depender de la categoría sintáctica de la palabra completa.

Análisis Sintáctico

El **análisis sintáctico** debe utilizar los resultados del análisis morfológico para construir una descripción estructurada de la oración. El objetivo de este proceso, denominado **análisis *(parsing)***, es convertir la lista lineal de palabras que constituyen la oración en una estructura que defina las unidades que representa la lista lineal. Lo que realmente es importante aquí es que una secuencia lineal se convierte en una estructura jerárquica y que esta estructura se corresponde con las unidades de la oración (tales como sintagmas nominales) que se corresponderán con unidades con significado cuando se realice el análisis semántico Aunque no todos los sistemas sintácticos realizan lo anterior, un buen resultado que se logra es crear un conjunto de entidades que se denominan **marcas de referencia (reference markers)** Estas marcas de referencia indican el lugar en el cual acumular la información referente a las entidades conforme se va disponiendo de ella.

Análisis Semántico

El análisis **semántico** incluye dos importantes acciones

Debe hacer corresponder las palabras concretas con los objetos apropiados de la base de conocimiento o de datos

Debe crear las estructuras correctas para que se corresponda la forma en que los significados de las palabras individuales se combinen con los demás.

Integración del Discurso

En este punto, se ha resuelto la clase de cosas a las cuales se refiere la oración

Análisis de la Pragmática

El paso final hacia una comprensión eficaz consiste en decidir qué hacer como resultado Una posibilidad es almacenar lo dicho como un hecho y hacerlo Para algunas oraciones, en donde el efecto deseado es claramente declarativo, ésta es precisamente la situación correcta

El paso final del procesamiento de la **pragmática** consiste en traducir, cuando sea necesario, la representación basada en conocimiento en una orden para que la ejecute el sistema.

Procesamiento Sintáctico

El procesamiento sintáctico es el paso en el cual una oración lineal de entrada se convierte en una estructura jerárquica que corresponde a las unidades de significado de la oración. Juega un importante papel en muchos sistemas de comprensión del lenguaje principalmente por dos razones:

El procesamiento semántico funciona sobre los constituyentes de la oración. Si no existe un paso de análisis sintáctico, el sistema semántico debe identificar sus propios constituyentes Por otro lado, si se realiza un análisis sintáctico, se restringe el número de constituyentes a considerar por el semántico Puede desempeñar un importante papel como reductor de la complejidad global del sistema

Aunque frecuentemente se puede extraer el significado de una oración sin usar hechos gramaticales, no siempre es posible hacerlo

Aunque existen muchas formas de realizar un análisis sintáctico, casi todos los sistemas que se utilizan realmente tienen dos componentes principales:

Una representación declarativa, denominada **gramática**, de los hechos sintácticos sobre el lenguaje Esta gramática describe la estructura de las cadenas de caracteres de un lenguaje concreto

Un procedimiento, denominado **analizador** (**parser**), que compara la gramática con las oraciones de entrada para producir estructuras analizadas

Esta estructura se llama **árbol de análisis**.

El proceso de análisis realiza dos cosas:

- Determina que frases se aceptan como bien formadas sintácticamente (es decir, gramaticales) y cuales no
- Asignar una estructura a las frases sintácticamente bien formadas

Gramática y Analizadores

La forma mas usual de representar la gramática es mediante un conjunto de reglas de producción Los símbolos que son susceptibles de expandirse mediante reglas se denominan **símbolos no terminales**. Aquellos símbolos que se corresponden directamente con las cadenas que deben encontrarse en la oración de entrada reciben el nombre de **símbolos terminales**.

Sin tener en cuenta la base teórica de la gramática, el proceso de análisis sintáctico toma reglas de las gramática y las compara con la oración de entrada Cada regla que se empareje añade algo a la estructura de la oración que se está construyendo La estructura más simple que se puede construir es un **árbol de análisis** (**parse tree**), en el que simplemente se almacenan las reglas y la forma en que se emparejan

Cada nodo del árbol de análisis se corresponde con una palabra de la entrada o con un símbolo no terminal de la gramática Cada nivel del árbol de análisis corresponde con la aplicación de una regla gramatical Una gramática especifica dos cosas sobre el lenguaje:

Su pobre capacidad generativa, que es el conjunto de las oraciones contenidas en el lenguaje. Este conjunto (denominado conjunto de **oraciones gramaticales**) se construye con aquellas oraciones que se pueden emparejar completamente por una serie de reglas de la gramática.

Su fuerte capacidad generativa, que son las estructuras que se asignan a cada oración gramatical del lenguaje

Análisis Sintáctico Descendente Frente a Análisis Sintáctico Ascendente

Para analizar una frase es necesario encontrar una forma de generar una oración a partir del símbolo inicial. Esto puede hacerse de dos formas:

- **Análisis descendente** se comienza con el símbolo inicial y se aplican las reglas de la gramática hacia delante hasta que los símbolos terminales del árbol se correspondan con los componentes de la frase analizadora

- **Análisis ascendente**: Se comienza con la frase a analizar y se aplican las reglas de la gramática hacia atrás hasta producir un único árbol cuyos terminales sean las palabras de la oración y cuyo nodo raíz sea el símbolo inicial.

En ocasiones, estos dos enfoques se combinan para formar un único método denominado **análisis ascendente con filtraje descendente**. En este método el análisis sintáctico trabaja esencialmente de forma descendente (es decir, las reglas de la gramática se aplican hacia atrás). Sin embargo, el analizador puede eliminar inmediatamente los constituyentes que nunca podrán combinarse en estructuras útiles de alto nivel sin más que hacer uso de tablas precalculadas para una gramática concreta

Encontrar una Interpretación o Encontrar Muchas

El proceso de comprensión de una oración es un proceso de búsqueda en el que debe explorarse un gran universo de posibles interpretaciones para encontrar una que cumpla todas las restricciones impuestas por una oración concreta Al igual que en otros procesos de búsqueda, debe decidirse si se exploran todos los posibles caminos o, si por el contrario, se explora sólo aquel que tenga la mayor probabilidad y devolver como respuesta únicamente el resultado de seguir ese camino

Suponiendo que un procesador de oraciones mira de una en una las palabras que forman una oración de entrada, de izquierda a derecha, y que hasta ahora se ha procesado:

" Have the students who missed the exam "

En este punto, el procesador podría seguir los dos caminos siguientes:

"Have" es el verbo principal de una oración imperativa tal como:

"Have the students who missed the exam take it today."

o

"Have" es un verbo auxiliar de una oración interrogativa tal como:

"Have the students who missed the exam taken it today?" (2)

Existe cuatro formas de tratar frases como ésta:

Todos los caminos Seguir todos los posibles caminos y construir todos los posibles componentes intermedios Muchos de los componentes se ignorarán más adelante al no aparecer las demás entradas necesarias para usarlos. La principal desventaja de este enfoque es que puede llegar a ser muy ineficiente, ya que incorpora la construcción de muchos constituyentes erróneos y se llega a muchos callejones sin salida.

El mejor camino con vuelta atrás Seguir sólo un camino cada vez, pero almacenar, en cada punto de elección, la información necesaria para realizar otra elección si el camino elegido no lleva a una interpretación completa de la oración. Existen dos inconvenientes importantes con este enfoque El primero de ellos es que se puede malgastar una gran cantidad de elección, aunque sólo sea necesario volver atrás sobre unos pocos de esos puntos. El segundo es que con frecuencia suele analizarse el mismo constituyente varias veces

Mejor camino con recolocaciones Seguir un único camino a la vez, pero cuando se detecte un error, los componentes que ya se han formado se recolocan explícitamente. Su mayor inconveniente es que necesita que existan interacciones entre las reglas de la gramática, que deben explicarse en las reglas para trasladar componentes de un sitio a otro El intérprete frecuentemente es adecuado, en lugar de ser sencillo y estar dirigido exclusivamente por la gramática.

Esperar y ver. Seguir sólo un camino, pero en lugar de tomar decisiones sobre la función de cada componente conforme van apareciendo, postergar la decisión hasta que se disponga de la suficiente información para realizar una decisión correcta Este enfoque es muy eficiente, pero tiene el inconveniente de que si la cantidad de información almacenada necesaria es mayor que el tamaño del buffer, el intérprete no funcionará

Existen muchos tipos diferentes de sistemas analizadores sintácticos. A continuación se mencionan cuatro que se han utilizado ampliamente en sistemas de comprensión del lenguaje:

- **Analizadores de diagramas**, que proporcionan una manera de evitar el backup, almacenando los constituyentes intermedios de forma que se puedan reutilizar a lo largo de caminos de análisis distintos
- **Gramática de cláusulas definidas**, en las que las reglas de la gramática se escriben como cláusulas de **PROLOG** y se utiliza el intérprete de PROLOG para realizar el análisis descendente y primero en profundidad
- **Una maquina de estados finitos** es una estructura en donde la computación se modela como la transición de un estado a otro, de entre un número finito de ellos. Se marca un estado como el estado inicial Se pueden marcar uno o mas estados como estados finales Si se encuentra el final de la cadena de entrada y la máquina esta en un estado final, se acepta la entrada como una frase gramatical Los estados de la máquina están conectados mediante arcos, cada uno de los cuales esta etiquetado con un símbolo de entrada cuya presencia, como siguiente carácter en la cadena de entrada, permitirá que suceda la transición asociada.
- **Redes de transiciones aumentadas (augmented transition networks o ATNs)**, en las que el proceso de análisis sintáctico se describe como la transición desde un estado inicial hasta un estado final en una red de transiciones que corresponde con una gramática del inglés.

Redes de Transiciones Aumentadas

Una **red de transiciones aumentada (ATN)** es un procedimiento de análisis sintáctico descendente que permite incorporar al sistema varias clases de conocimiento de forma que pueda funcionar eficientemente Una ATN es similar a una máquina de estados finitos en la que se ha ampliado el tipo de etiquetas que se pueden asociar a los arcos que definen las transiciones Los arcos pueden etiquetarse con una combinación arbitraria de lo siguiente

- Palabras específicas
- Categorías de palabras
- Llamadas a otras redes que reconozcan componentes importantes de una oración.
- Procedimientos que realizan comprobaciones arbitrarias tanto de la entrada actual como de los componentes de la oración que ya se han identificado
- Procedimientos que constituyen estructuras que formarán parte del análisis final

Sólo es necesario que una subred aparezca una sola vez aunque se use en más de un sitio. Una red se puede llamar recursivamente Se puede usar cualquier número de registros internos para que contengan el resultado del análisis sintáctico Se puede construir el resultado de una red a partir de los valores contenidos en varios registros del sistema Un estado puede ser tanto un estado final, cuando se encuentra una oración completa, o un estado intermedio, en el que sólo se ha reconocido una parte de la oración. Finalmente, el contenido de un registro puede modificarse en cualquier momento

Además las ATNs pueden usarse de muchas formas·

- Se pueden intercambiar los contenidos de los registros.
- En los arcos se pueden colocar comprobaciones arbitrarias

Gramáticas con Unificación

Las **gramáticas ATN** tienen un importante componente procedimental. La gramática describe el orden en el que deben construirse los constituyentes A las variables se les dan valores explícitos, y se les tiene que haber asignado un valor antes de ser referenciadas. Este carácter procedimental limita la eficacia de las gramáticas ATN en algunos casos

Cuando un analizador sintáctico aplica reglas a una oración, lleva a cabo dos tipos de acciones principales

- Emparejamiento de los constituyentes de la oración con las reglas de la gramática
- Construcción de la estructura (que se corresponde con el resultado de combinar los constituyentes)

El **emparejamiento** y la **construcción de estructuras** son las operaciones que la unificación lleva a cabo de forma natural. Por lo tanto, un candidato evidente para representar las gramáticas es alguna estructura en la que se puede definir un operador de unificación. **Los grafos acíclicos dirigidos (GAD)** tiene exactamente esta propiedad. Cada GAD representa un conjunto de pares atributo-valor

Dos grafos son unificables si, recursivamente, todos sus subgrafos son unificables El resultado de una unificación llevada a cabo con éxito es un grafo compuesto por la unión de los subgrafos de las dos entradas, indicando todos los enlaces realizados Este proceso toca fondo cuando un subgrafo no es un par atributo-valor, sino que es un valor de un atributo. En este momento, es necesario definir qué significa unificar dos valores. Dos valores idénticos son unificables

Los ATN tienen también algunos inconvenientes:

Pueden ser caros de ejecutar si se requiere gran cantidad de vueltas atrás

Aunque puede utilizarse el conocimiento semántico para rechazar caminos posibles incorporándolo a las comprobaciones de los arcos, no es fácil usar dicho conocimiento para ayudar a elegir el más probable de entre diversos caminos de forma que puede explorarse en primer lugar No hay forma de usar funciones heurísticas

El proceso de análisis fracasará a menos que el sistema conozca todas las palabras de la frase y que la estructura completa de la misma empareje exactamente con un camino de la red.

Análisis Semántico

La realización de un análisis a una oración es sólo un primer paso para comprenderla. Todavía falta fabricar una representación del significado de la oración Como la compresión es una correspondencia, primero se tiene que definir el lenguaje al que se intenta llegar.

Cuando el lenguaje natural se usa como un lenguaje de interfaz con otro programa (como un sistema de peticiones a una base de datos o un sistema experto), el lenguaje destino debe ser una entrada para el otro programa Así, el diseño del lenguaje destino está influido por este programa

El propósito principal del procesamiento semántico es la creación de una representación en el lenguaje destino del significado de una oración, juega también otro papel importante Impone restricciones a las representaciones que se pueden construir y, debido a las conexiones estructurales que deben existir entre la estructura sintáctica y semántica, también proporciona una forma de selección entre distintos análisis sintácticos que compiten entre sí.

Procesamiento Léxico

El primer paso que debe dar cualquier sistema de procesamiento semántico es buscar las palabras individuales en un **diccionario (o lexicon)** y extraer sus significados. Desafortunadamente, muchas palabras tienen varios significados, y puede que no sea posible elegir el correcto mirando la palabra por separado

El proceso mediante el cual se determina el significado correcto de una palabra recibe el nombre de **desambigüación del sentido de una palabra** o **desambigüación léxica** Se logra asociando a cada palabra del diccionario una información sobre el contexto en el que puede aparecer cada aceptación de esa palabra Por ejemplo:

> "El significado de la palabra "diamond" en la oración "I'll meet you at the diamond" (nos encontramos en el campo de béisbol) podría determinarse fácilmente si se recordara como parte de la entrada léxica para *at* el hecho de que *at* necesita un objeto de TIEMPO o UBICACIÓN". (3)

Este tipo de propiedades del sentido de las palabras se denomina **marcas semánticas** También son marcas semánticas útiles las siguientes

- OBJETO-FISICO
- OBJETO-ANIMADO
- OBJETO-ABSTRACTO

Desafortunadamente para resolver el problema de la desambiguación léxica de forma completa, es necesario introducir cada vez más y más marcas semánticas, y cada vez más precisas

Procesamiento a Nivel de Oración

Se han desarrollado varios enfoques para tratar el problema de la creación de una representación semántica para una oración, entre los que se encuentran los siguientes

- **Gramáticas semánticas**, que combinan conocimiento sintáctico, semántico y pragmático en un único conjunto de reglas en forma de gramática El resultado del análisis en una gramática de este tipo es una descripción semántica de la oración, en lugar de una representacion sintáctica
- **Las gramáticas de casos**, en donde el analizador construye una estructura que contiene cierta información semántica, aunque es necesaria una interpretación posterior.
- **Análisis conceptual**, en el que se combina conocimiento sintáctico y semántico en un único sistema de interpretación conducido mediante conocimiento semántico Con este enfoque, el análisis sintáctico está subordinado a una interpretación semántica, la cual normalmente se utiliza para establecer fuertes expectativas para las estructuras de una oración concreta
- **Interpretación semántica aproximadamente composicional**, en la que se aplica un procesamiento semántico al resultado del trabajo del analizador sintáctico Esto puede lograrse o bien de forma creciente, conforme se construyen los constituyentes, o bien todo a la vez, cuando se haya construido una estructura que se corresponde con una oración completa.

Gramáticas Semánticas

Una **gramática semántica** es una gramática libre de contexto en donde la elección de los no terminales y las reglas de producción se gobiernan mediante una función tanto semántica como sintáctica Además, existe normalmente una acción semántica asociada a cada regla de la gramática. El resultado del análisis y la aplicación de todas las acciones semánticas asociadas, es el **significado** de la oracion. Este fuerte emparejamiento entre las acciones semánticas y las reglas de la gramática funciona porque las reglas de la gramática están diseñadas teniendo en cuenta conceptos semánticos claves

Una gramática semántica puede usarse exactamente en las mismas situaciones en que puede hacerlo una gramática estrictamente sintáctica

Las principales ventajas de las gramáticas semánticas son las siguientes:

- Al completar el análisis, el resultado se puede usar inmediatamente sin el paso adicional de procesamiento que se necesitaba si no se hubiera realizado una interpretacion semántica durante el análisis.
- Pueden evitarse muchas ambiguedades que surgirían durante un análisis estrictamente sintáctico ya que algunas de estas interpretaciones no tendrían sentido semánticamente y, por tanto, no podrían generarse con la gramática semántica.
- Puede ignorarse los aspectos sintácticos que no afecten a la semántica.

Existen, sin embargo, algunas desventajas al usar las gramáticas semánticas

- El número de reglas necesarias puede ser muy grande al perder muchas generalizaciones sintácticas
- Como el número de reglas de la gramática puede ser muy elevado, el proceso de análisis puede ser muy costoso

Gramática de Casos

Las reglas de la gramática se escriben para representar regularidades sintácticas, en lugar de semánticas Pero las estructuras producidas por las reglas se corresponden con relaciones semánticas, en lugar de relaciones estrictamente sintácticas

Es importante hacer notar que la información semántica realmente se introduce en la sintaxis del lenguaje En la **gramática de casos** se describen relaciones entre verbos y sus argumentos Un caso gramatical puede indicar varios casos semánticos

No existe un acuerdo claro sobre cuáles serían los casos semánticos correctos, aunque algunos obvios son los siguientes.

- **Agente** causante de una acción (normalmente animado).
- **Instrumento** causa del evento o el objeto usado para causar el evento (normalmente inanimado)
- **Dativo** entidad afectada por la acción (normalmente animada).
- **Factitivo**: objeto o resultado de un evento
- **Locativo**: lugar del evento
- **Fuente** lugar a partir del cual se mueve algo.
- **Objetivo** lugar hacia el cual se mueve algo.
- **Beneficiario**: ser en cuyo beneficio ocurre el evento (normalmente animado).
- **Tiempo**: momento en el que ocurre el evento.
- **Objeto**: entidad que cambia sobre la que se actúa, en términos generales.

El proceso de análisis en una representación de casos está fuertemente condicionada por las entradas léxicas asociadas a cada verbo. Los lenguajes tienen reglas para hacer corresponder las estructuras de casos subyacentes con las formas sintácticas superficiales

Los análisis que hacen uso de una gramática de casos normalmente están **conducidos por expectativas** Una vez que se ha localizado el verbo de la oración, éste se usa para predecir los sintagmas nominales que aparecerán y para determinar la relación de estos sintagmas con el resto de la oración

Análisis Conceptual

El **análisis conceptual (conceptual parsing)**, al igual que las gramáticas semánticas, es una estrategia diseñada para encontrar en un solo paso tanto la estructura como el significado de una oracion El análisis conceptual lo dirige un diccionario que describe el significado de las palabras como estructuras de **dependencias conceptuales (CD)**

El análisis de una oración con una representación de dependencias conceptuales es similar al proceso del análisis con gramática de casos En ambos sistemas, el proceso de análisis está fuertemente dirigido por un conjunto de expectativas que se establecen sobre la base del verbo principal de la oración. Pero con la representación de un verbo en CD es de más bajo nivel que la de un verbo en una gramatica de casos, CD normalmente proporciona un grado mayor de potencia predictiva El primer paso para realizar una correspondencia entre una oración y su representación CD es un procesador sintáctico que extraiga el verbo y el nombre principal. También determina la categoría sintáctica y la clase aparente del verbo (es decir, de estado, transitivo o intransitivo) El procesador conceptual estará entonces en funcionamiento Utiliza un diccionario de verbos, que contienen una entrada por cada entorno en el que puede aparecer un verbo.

Una vez elegida la entrada correcta, el procesador conceptual analiza el resto de la oración buscando los componentes que se llenarán en las ranuras vacías de la estructura del verbo.

Debido a la gran cantidad de información semántica que se usa en el proceso de comprensión, las oraciones ambiguas desde un punto de vista puramente sintáctico, pueden tener una única interpretación. Desafortunadamente, la cantidad de información semántica necesaria para que el sistema funcione correctamente es inmensa Todas las reglas simples tiene excepciones.

Interpretación Semántica Aproximadamente Composicional

Si se realiza estrictamente un análisis sintáctico de la oración, una forma sencilla de generar una **interpretación semántica** es la siguiente

1 Buscar las palabras del diccionario que contengan una o más definiciones para el mundo, definida en términos de la representación destino elegida Estas definiciones tienen que describir la forma de representar la idea que se corresponde con la palabra, asi como la forma en que la idea que representa la palabra se puede combinar con las ideas que representan las demás palabras de la oración.

2 Utilizar la información sobre la estructura que contiene la salida del analizador sintáctico para generar restricciones adicionales, además de las extraídas del diccionario, sobre la forma en que las palabras individuales pueden combinarse para formar estructuras más grandes con significado

Semántica de Montage

La idea principal que existe detrás de la semántica composicional es que en cada paso del proceso de análisis sintáctico, existe un correspondiente paso de interpretación semántica Cada vez que los constituyentes sintácticos se combinan para formar una unidad sintáctica más grande, sus interpretaciones semánticas correspondientes se combinan para formar una unidad semántica más grande Las reglas necesarias para combinar estructuras semánticas están asociadas con las correspondientes reglas para combinar estructuras sintácticas La palabra "**composicional**" es adecuada para definir este enfoque porque trata el significado de cada constituyente de una oración como una composición de los significados de sus constituyentes con el significado de la regla que se usa para crearlo La principal base teórica de este enfoque es la lógica moderna.

La Interacción entre la Sintaxis y la Semántica

Si se utiliza un enfoque composicional de la semántica, entonces para cada constituyente sintáctico se aplican las reglas de interpretación semántica que eventualmente producirán una interpretación de la oración completa. Al implantar un sistema se tienen que tomar algunas decisiones sobre la forma de controlar las interacciones entre los procesadores sintácticos y semánticos Las posiciones extremas son

- Cada vez que se genera un constituyente sintáctico se aplica inmediatamente una interpretación semántica para él
- Esperar hasta que se haya analizado la oración completa, y entonces interpretarla toda

Comprensión de Frases Múltiples

El entender una única frase suele consistir en asignar significados a las palabras individuales y, a continuación, asignar una estructura a la frase completa. Pero el comprender un conjunto de frases. ya sea una porción de un texto o una parte de un diálogo, también requiere que se descubran las relaciones entre las frases. Hay un gran número de tales relaciones que pueden ser importantes en un texto concreto, incluyendo

- Objetos idénticos

> "Guillermo tenia un balón rojo. Juan lo quería.
>
> La palabra *lo* debería identificarse como refiriéndose al balón rojo. Las referencias de este tipo se llaman referencias anafóricas o anáforas." (4)

- Partes de objetos
- Partes de acciones
- Objetos involucrados en acciones
- Cadenas causales
- Secuencias de planificación.

El Procesamiento de la Pragmática

Existe un conjunto de relaciones importantes que pueden surgir entre las frases y las partes de sus contextos del discurso, entre las que están.

- Entidades idénticas
- Partes de entidades
- Partes de acciones.
- Entidades involucradas en acciones
- Elementos de conjuntos
- Nombres de individuos
- Encadenamientos causales
- Planificación de secuencias
- Presuposiciones implícitas.

Para poder reconocer este tipo de relaciones entre las oraciones, es necesaria gran cantidad del conocimiento acerca del mundo. Los programas capaces de comprender oraciones múltiples se apoyan o bien en grandes bases de conocimiento o en fuertes restricciones sobre el dominio del discurso, de forma que sólo sea necesaria una base de conocimiento más limitada La forma de organizar este conocimiento es crítica para el éxito del programa de comprensión

Aplicación

Realizar una aplicación que permita interactuar en lenguaje natural a un usuario y la computadora, para obtener el número telefónico de su directorio personal

NOTA Ver código del programa en el Anexo 3: Procesamiento de Lenguaje Natural.

Capítulo

4

Percepción Visual

Introducción

Las máquinas de visión precisas, abren un nuevo dominio de aplicaciones de las computadoras. Desde hace veinte años, los investigadores en inteligencia artificial han desarrollado e implementado técnicas de visión artificial útiles en muchos campos, como navegación de robots móviles, procesos de fabricación, diagnóstico médico, interpretación de fotografías y seguimiento de proyectiles militares

La **percepción visual** puede ser definida como los procesos de obtención, caracterización e interpretación de información de imágenes tomadas de un mundo tridimensional. Estos procesos también llamados **visión por computadora** pueden a su vez ser subdivididos en seis áreas principales

1 Captación
2 Preprocesamiento
3 Segmentación
4 Descripción
5 Reconocimiento
6 Interpretación

La **captación** es el proceso a través del cual se obtiene una imagen visual. El **preprocesamiento** incluye técnicas tales como la reducción de ruido y realce de detalles. La **segmentación** es el proceso que divide una imagen en objetos que sean de interés. Mediante los **procesos de descripción** se obtienen características (por ejemplo, tamaño, forma) convenientes para diferenciar un tipo de objeto de otro. El **reconocimiento** es el proceso que identifica estos objetos (por ejemplo, un llave inglesa, un tornillo, un soporte de motor). Finalmente la **interpretación** le asocia un significado a un conjunto de objetos reconocidos

El propósito de un sistema de visón artificial es obtener de una imagen la información necesaria y útil para la ejecución de una tarea. En el caso más simple, la información se refiere solamente a la posición y orientación de un objeto aislado; en otros casos, se deben reconocer los objetos y determinar sus relaciones espaciales Esto puede observarse como un proceso en el que se desarrolla la descripción de la escena que se está viendo a partir de una hilera de imágenes La descripción debe ser apropiada para la aplicación particular Esto es, las características visuales y relevantes deben ser descartadas, mientras que las relaciones necesarias entre partes de objetos deben ser deducidas de su proyección óptica.

Ideas de cuatro campos relacionados han contribuido al rápido progreso de la percepción visual el procesamiento de imágenes, análisis de medición, el reconocimiento de patrones y análisis de imagenes

Algunas de las acciones que se pueden realizar en estos campos son

- **El procesamiento de imágenes** Tiene que ver con la producción de nuevas imágenes a partir de las ya existentes Normalmente estas nuevas imágenes se obtienen aplicando técnicas de teoría de sistemas lineales y para mejorar la apariencia de las imágenes originales de cara a la persona. Para la visión artificial lo que le interesa es la generación de descripciones simbólicas y el uso de estas descripciones para permitir a la computadora que controla el sistema tomar las decisiones apropiadas
- **Análisis de medición**. Para imágenes que contengan un unico objeto, determinado la extensión bidimensional del objeto representado
- **Reconocimiento de patrones** Para imágenes de un solo objeto, clasificando el objeto dentro de una categoría a parte de un número finito de posibilidades.
- **Análisis de imágenes** Para imágenes que contengan varios objetos, localizando los objetos en la imagen, clasificándolos y construyendo un modelo tridimensional de la escena

Arquitectura para la visión

En una arquitectura para la visión, el primer paso es convertir la señal de vídeo analógica en imagen digital El siguiente paso es extraer características de la imagen, como bordes y regiones. Los bordes pueden ser detectados por algoritmos que buscan series de pixeles adyacentes con valores diferentes Ya que los valores de los pixeles se ven afectados por varios factores, los bordes pequeños con orientaciones similares pueden ser agrupados con los grandes. Las regiones, por otro lado, se encuentran agrupando pixeles similares La detección de bordes y regiones son procesos intensivos en lo que se refiere a computadoras Después se infieren orientaciones en tercera dimensión para las distintas regiones Textura, iluminación y datos sobre distancias son útiles para esta tarea. También pueden ser válidos los supuestos sobre el tipo de objetos que se presentan gráficamente. Tras hacer esto, las superficies son reunidas en objetos tridimensionales Los objetos pequeños se combinan para formar objetos compuestos más grandes En este punto, la imagen se segmenta en entidades discretas. El paso final consiste en comparar estas entidades con la base de conocimiento para quedarse con la interpretación más probable para ellos

Generalmente, hay dos métodos mediante los que se implementan los sistemas de visión. En el primer metodo, el objetivo es deducir una imagen a las líneas que forman el contorno de cada objeto Este método usa varios filtros para obtener información de la imagen y aumentadores del contraste para convertir todas las partes de la imagen en blanco o negro Esto se llama algunas veces una **imagen binaria** porque no hay áreas grises (cada punto en la imagen es blanco o negro).

La producción actual de imágenes binarias se deja generalmente a un preprocesador, al que le concierne dar una interpretación a la imagen. Aunque el filtrado de imágenes se puede hacer digitalmente, en los sistemas simples es común hacerlo usando circuitos analógicos que producen imágenes de alto contraste

La ventaja de las imágenes binarias es que proporcionan fronteras estrictamente definidas que una computadora puede conocer, es claro dónde comienza y dónde termina cada objeto Las imágenes de alto contraste son más comunes en entornos controlados donde se sabe por adelantado que sólo se van a ver unos pocos objetos seleccionados En la mayoría de los casos, este es el tipo de implementación usado por los sistemas bidimensionales de procesamiento de imagen

El segundo método mediante el que se implementan sistemas de visión intenta dar a la computadora una visión de la imagen más parecida a la visión humana Este método proporciona información a la computadora sobre el brillo de parte de la imagen Esto permite derivar dos rasgos importantes de la imagen superficies y sombras. La computadora puede usar ambos para proporcionar información tridimensional sobre las imágenes y ayudar a resolver conflictos cuando un objeto cubre parcialmente a otro Este tipo de procesamiento de la imagen se usa en sistemas de visión tridimensionales

Virtualmente todos los sistemas de visión usan las imágenes en blanco y negro en vez de en color por dos razones primero, el color generalmente no es necesario y, segundo, la adición de información de color exige incluso demandas mayores sobre la computadora y el software que están procesando la imagen

Sistemas bidimensionales

Los **sistemas de visión bidimensional** requieren un entorno estrictamente controlado y restringido para funcionar apropiadamente porque procesan todas las imágenes como si fueran planas De hecho, el procesamiento de imagen bidimensional es llamado a **veces procesamiento de imagen plana** por esta razón Debido a que sólo se representan dos dimensiones, generalmente se usa una imagen binaria de alto contraste.

Esto hace que los objetos se reduzcan a sus contornos Los sistemas bidimensionales son completamente comunes en entornos tales como líneas de ensamblajes automatizados, donde es de interés la orientación, localización o reconocimiento de ciertas formas bidimensionales específicas.

Un problema común de los sistemas bidimensionales es que un sistema puede tener dificultad reconociendo un objeto cuando está parcialmente bloqueado o cubierto por otro objeto. Debido a que la información tridimensional no es accesible, a la computadora se le da lo que parece ser información errónea o conflictiva y puede que la computadora no sea siempre capaz de resolver un conflicto correctamente.

Sistemas tridimensionales

El objetivo de los **sistemas de visión tridimensional** es tratar correctamente todos los problemas de la visión que generan los objetos que infieren con otro objeto por ejemplo, estando encima o enfrente de él

Los sistemas tridimensionales deben resolver varios problemas que no existían en el restringido enfoque bidimensional Primero, la imagen que procesa la computadora contienen mucha información En una imagen binaria de alto contraste, cada pixel puede ser almacenado en un bit porque o bien es blanco o bien es negro (Un **pixel** es un punto discreto de la imagen de vídeo) Sin embargo, conseguir información tridimensional requiere información sobre el brillo relativo de cada pixel. Dependiendo de varias restricciones, puede haber desde unos pocos grados de brillo hasta varios cientos de ellos

Problemas de reconocimiento comunes

Uno de los problemas más difíciles al crear un sistema de visión es **el reconocimiento de objetos superpuestos** El problema no es que una computadora no pueda decir que un objeto está enfrente de otro Generalmente, las sombras y las diferencias en el sombreado dan suficientes pistas. La diferencia real es programar la computadora para reconocer los objetos parciales como lo que son. Por ejemplo, si se enseña a la computadora que un triángulo tiene tres lados y tres vértices, y si la computadora ve un triángulo que tiene uno de sus vértices obscurecido por otro objeto (o quizá fuera del campo de visión de la cámara), ¿cómo puede saber la computadora que está viendo un triángulo?

Hay muchos enfoques a este problema, pero quizá la solución más cercana a la forma de actuar del ojo humano, es llamada **alucinación controlada**. En este método, la computadora, guiada por la información inicial, postula que está viendo un triángulo e intenta verificarlo usando algunos medios, tales como que las dos líneas se intersecarán en algún punto de la región oscura.

Otro problema difícil es programar la computadora para reconocer clases de objetos, que un árbol es un árbol o que una casa es una casa. Es mucho más fácil hacer que la computadora reconozca un objeto específico que hacerla reconocer objetos de una clase La razón de esto es que puede dar a objetos específicos un conjunto estricto de limitaciones que deben cumplir, pero se debe mantener una definición de clase completamente general y cubrir todas las ligeras variaciones.

Ninguna visión general del procesamiento de imágenes está completa sin una breve mención a las ilusiones ópticas y sus efectos sobre los sistemas de visión informáticos. Por ejemplo, una carretera muy larga y recta, parece estrecharse y desaparecer finalmente; esto es, las cosas parecen más pequeñas cuanto más lejos están.

Mientras que algunos aspectos del entendimiento de imagen se reducen a análisis de medición y a reconocimiento de patrones, el problema en su totalidad queda sin resolver debido a las dificultades que se incluyen a continuación

Una imagen es bidimensional, mientras que el mundo es tridimensional. Necesariamente se pierde alguna información cuando se crea una imagen

El valor de un único pixel se ve afectado por muchos fenómenos diferentes como el color del objeto, la fuente de luz, el ángulo y la distancia de la cámara, la polución del aire, etc. Es muy difícil desligar estos efectos.

Como resultado de esto, las imágenes son sumamente ambiguas. Dada una imagen simple, se puede construir cualquier número de mundos tridimensionales que podrían dar lugar a la imagen Además de que el procesamiento es lento

Los sistemas de visión actuales restringen sus aplicaciones a situaciones en las que los objetos pueden ser fácilmente aislados en una imagen Típicamente, los objetos se presentan sobre un fondo con alto contraste, sin partes ocultas, con luz controlada para eliminar sombras y otros factores que hacen difícil la segmentación Los objetos son reconocidos extrayendo características de dos dimensiones de los mismos, que son comparadas con objetos prototipo de dos dimensiones.

Técnicas de reconocimiento

Hay varias formas en la que la computadora puede reconocer objetos A continuación se describen tres métodos distintos de reconocimiento de objetos.

Reconocimiento por Angulos

En un entorno altamente controlado es posible identificar correctamente un triángulo o un cuadrado midiendo el ángulo de un vértice. Limitar los tipos de triángulos a uno isósceles hace simple distinguir un triángulo de un cuadrado sólo tiene que chequear dos puntos en cada vértice. Si los puntos corresponden a ángulos rectos, el objeto tiene que ser un cuadrado Si corresponden a ángulos de 60 grados, el objeto debe ser un triángulo

Para identificar los objetos requiere, que estén en unas pocas posiciones, no pueden ser rotados aleatoriamente

El reconocimiento por Angulos puede aplicarse a objetos de diferente tamaño.

Reconocimiento por Puntos-Clave

Una forma de que el programa identifique objetos que tienen siempre el mismo tamaño es examinar sólo unos pocos puntos clave Las localizaciones clave se eligen de forma que un objeto y sólo uno satisfagan las condiciones

La técnica falla al rotar el objeto y, cuando un objeto está encima del otro porque intenta cada punto sólo una vez

Este requiere un entorno más restrictivo incluso que la versión basada en los ángulos y se confunde fácilmente

Reconocedor Delta-D

Esta basado en cambios de dirección. Por ejemplo, comenzando en cualquier punto de un objeto triángulo, se siguen las líneas hasta que vuelve al punto de partida, habrá cambiado de direccion tres veces El número de veces que cambia la dirección es igual al número de vértices del objeto Cada vez que se encuentre un vértice, hay un cambio de dirección. Estos cambios son a menudo llamados **valores delta** en matemáticas y en informática. Debido a que esta técnica se basa en un cambio en la dirección, este método es llamado **delta-D**, donde la **D** representa dirección

Una de las mejores características de esta técnica es que operará correctamente sin importar en qué posición están los objetos Esta característica hace atractivo al reconocedor delta-D en muchas situaciones del mundo real

Otro punto bueno de la estrategia delta-D es que no importa el tamaño de los objetos Esto significa que puede ser usado para reconocer una clase de objetos

Debido a que el reconocedor delta-D es mucho más general que los dos enfoques previos, no opera tan eficientemente, sin embargo, el cambio de eficiencia a cambio de fiabilidad es mejor.

El reconocedor delta-D puede confundir fácilmente los objetos superpuestos, pero no confundirá un tipo de objeto con otro

Aplicación

Realizar un sistema de visión que identifique el objeto (triángulo o cuadrado) mostrado en pantalla utilizando el reconocimiento Delta-D

NOTA Ver código del programa en el Anexo 4. Percepción Visual

Capítulo

5

Robótica

Historia

En los años treinta los diseñadores de aeronaves desarrollaron el auto-piloto para aeroplanos El radio control remoto de aeronaves fue desarrollado al mismo tiempo La primera máquina que se parece al robot industrial moderno fue probablemente la máquina automática de pintado al spray diseñada por **Pollard** y **Roseland** Esta máquina fue entrenada siendo guiada a través de una tarea a medida que registraba su información de movimiento en un fonógrafo. Los robots entonces reproducían la información continuamente como ellos repetían su tarea de pintar. El controlador usaba una computadora analógica

Durante los años finales de la década de los cuarenta comenzaron programas de investigación en Oak Ridge y Argonne National Laboratories para desarrollar manipuladores mecánicos controlados de forma remota para manejar materiales radiactivos Estos sistemas eran del tipo "**maestro-esclavo**", diseñados para reproducir fielmente los movimientos de mano y brazos realizados por un operario humano El manipulador maestro era guiado por el usuario a través de una secuencia de movimientos, mientras que el manipulador esclavo duplicaba a la unidad maestra tal como le era posible

En los años cuarenta la mayor parte de los robots de fabricación casera fueron unidades mecánicas en las cuales la programación se determino en el equipo

A mediados de los años cincuenta, el acople mecánico se sustituyo por sistemas eléctricos e hidráulicos. Alrededor de **1953**, **Seiko** de Japón desarrollo lo que parecía ser un sistema de robot miniatura para ensamblar un amplio rango de partes mecánicas de reloj. **George C. Devol** desarrolló un dispositivo que llamó **dispositivo de transferencia programada articulada**, un manipulador cuya operación podía ser programada y que podía seguir una secuencia de pasos de movimientos determinados por las instrucciones en el programa Posteriores desarrollos de este concepto por **Devol y Joseph F. Engelberger** condujo al primer robot industrial en **1959**. La clave de este dispositivo era el uso de una computadora en conjunción con un manipulador para producir una máquina que podía ser enseñada para realizar una variedad de tareas de forma automática. Al contrario de las máquinas de automatización de uso dedicado, estos robots se podían reprogramar y cambiar de herramienta a un costo relativamente bajo para efectuar otros trabajos cuando cambiaban los requisitos de fabricación. Además comienzan a aparecer los primeros robots industriales en plantas de manufactura americana.

Al comienzo de la década de los sesenta, **H. A. Ernst** publicó el desarrollo de una mano mecánica controlada por computadora con sensores táctiles. Este dispositivo, llamado el MH-1, podía sentir bloques y usar esta información para controlar la mano de manera que apilaba los bloques sin la ayuda de un operario. Este trabajo es uno de los primeros ejemplos de un robot capaz de conducta adaptiva en un entorno razonable no estructurado.

En los años cincuenta y sesentas, el transistor y circuitos integrados permitieron a los robots de fabricación casera usar circuitos electrónicos para alguna construcción en la programación. Simultáneamente la industria fue construyendo equipo de automatización y de control numérico, el campo militar estuvo trabajando en dirección por inercia (o navegación), sistemas para planes y proyectiles.

A finales de los años setenta, **McCarthy** y sus colegas en el Stanford Artificial Intelligence Laboratories publicaron el desarrollo de una computadora con manos, ojos y oídos, (es decir, manipuladores, cámaras de televisión y micrófonos). Demostraron un sistema que reconocía mensajes hablados, veía bloques distribuidos sobre una mesa, y los manipulaba con instrucciones.

En **1970**, una tercera generación de computadoras reemplazo los transistores por circuitos integrados. Este hecho hizo disponible la computadora para usarla como un controlador lógico programable en máquinas automatizadas y de control numérico. Robots industriales, los cuales habían empezado a aparecer en los sesentas, fueron obteniendo un sólido punto de apoyo en la industria.

A finales de los ochentas surgió una segunda generación de robots industriales con más poderosos sensores.

Definición

La robótica es la ciencia o tecnología de robots, su diseño, manufactura, aplicación y uso. También se define como la ciencia de robotología, la cual es la razón por la que los robots se ensamblan para llevar a cabo una tarea.

Cuando la gente oye la palabra robot, la respuesta es normalmente una imagen visual del hardware, es decir, los dispositivos mecánicos y electrónicos que componen el aspecto físico del robot. Sin embargo, cuando más profundiza una persona en las tareas de un robot, más se hace patente que lo que forma un robot es el enlace del software con el hardware. El software otorga al dispositivo su inteligencia, y es esta inteligencia lo que distingue a un robot de otras formas de automatización.

> "La palabra *robot* proviene de la palabra checa *robota*, que significa trabajo. Una de las definiciones que se da a un robot es: un dispositivo automático que efectúa funciones ordinariamente asignadas a los seres humano.

> Una definición que describe a los robots industriales es: un robot es un manipulador reprogramable multifuncional diseñado para mover materiales, piezas o dispositivos especializados, a través de movimientos programados variables para la realización de una diversidad de tareas. " (5)

En otras palabras, un **robot** es un manipulador reprogramable de uso general con sensores externos que pueden efectuar diferentes tareas de montaje. Un robot debe poseer inteligencia que se debe normalmente a los algoritmos de computadora asociados con su sistema de control y sensorial.

Propiedades y Características de un Robot

Existen dos propiedades básicas que debe cumplir un robot

1 La **versatilidad**, que abarca la potencialidad geométrica y mecánica de un robot, es decir, por un lado, su actitud física para ejecutar tareas diversas y, por otro, la ejecución diversificada de una misma tarea La versatilidad implica una primera característica. todo robot debe parecer una estructura geométrica variable

2 La **autoadaptividad al entorno**, se trata de la capacidad de iniciativa de un robot para desarrollar de manera correcta tareas no especificadas, a pesar de las modificaciones imprevistas del entorno Esta propiedad conduce al robot a

- Ser capaz de captar el entorno que le rodea.
- Ser capaz de reflexiones para analizar el espacio de trabajo y elaborar una estrategia de ejecución
- Utilizar modos automáticos de operación

Los sensores de un robot son inexactos, y sus dispositivos son de precisión limitada. Siempre existe un grado de incertidumbre acerca de dónde está situado el robot exactamente, y en qué posición están los objetos y obstáculos en relación con él Los dispositivos del robot también son de precisión limitada

Muchos robots tienen que actuar en tiempo real El robot de un areoplano de guerra, no puede permitirse una exploración optima o dejar de inspeccionar el mundo

El mundo real es impredecible, dinámico e incierto Un robot no puede esperar proporcionar una descripción del mundo correcta y completa Esto significa que un robot tiene que tener en cuenta el margen que existe entre idear y ejecutar sus planes Este margen tiene varios aspectos. Por un lado, un robot puede no poseer suficiente información acerca del mundo como para realizar ningún proyecto útil En este caso, primeramente debe realizar una actividad de recopilación de información. Además, una vez que empieza la ejecución de un plan, el robot debe inspeccionar continuamente el resultado de sus acciones Si los resultados no son los esperados, entonces puede ser necesario volver a realizar el proyecto

Tipos de Robots

Hay básicamente dos tipos de robots. El primer tipo incluye a los de **sitio fijo**, robots industriales de ensamblaje, como los utilizados para ensamblar coches Esta clase de robot debe operar sólo en un entorno altamente controlado que ha sido diseñado explícitamente para él El segundo tipo consiste en robots autónomos Estos robots son diseñados para operar en el **mundo real** (el mundo que lo rodea)

Brazos de robot

Virtualmente todos los robots industriales son simples brazos de robot. **El brazo robótico** es el dispositivo fundamental manipulador de cualquier robot industrial o autónomo

El brazo de robot común está modelado según el brazo humano Mecánicamente, un robot se compone de un brazo y una muñeca más una herramienta. El brazo generalmente se puede mover con 3 grados de libertad o ejes La combinación de los movimientos posiciona a la muñeca sobre la pieza de trabajo La muñeca normalmente consta de tres movimientos giratorios La combinación de estos movimientos orienta a la pieza de acuerdo a la configuración del objeto. Estos tres últimos movimientos se denominan a menudo **elevación (pitch)**, **desviación (yaw)** y **giro (roll)**. Cada **eje** o **articulación** como es llamado comúnmente, es operado por su propio motor separado, o como en el caso de grandes brazos por un cilindro hidráulico

La mayor dificultad para controlar un brazo de este tipo no está en hacer el movimiento preciso, el hardware se ocupa de eso, sino en la coordinación de las articulaciones Para que un brazo de robot trabaje adecuadamente, todas las articulaciones deben trabajar unidas, simultáneamente, como un brazo humano

La **cinemática del brazo del robot** trata con el estudio analítico de la geometría del movimiento de un brazo de robot con respecto a un sistema de coordenadas de referencia fijo sin considerar las fuerzas o momentos que originan el movimiento.

La **dinámica del robot**, por otra parte, trata con la formulación matemática de las situaciones del movimiento del brazo Las ecuaciones dinámicas del movimiento de un robot son un conjunto de ecuaciones matemáticas que describen la conducta dinámica del robot. Tales ecuaciones de movimiento son útiles para simulación en computadora del movimiento del brazo, el diseño de ecuaciones de control apropiadas para robot y de evaluación del diseño y estructura cinemática del robot.

Robots Industriales

Un nivel más sofisticado de control puede conseguirse añadiendose servomecanismos que pueden dirigir la posición de cada grado de libertad para tomar cualquier valor. La suma de llamadas del servo control por retroalimentación implementada con sensores, como potenciometros, codificadores, etc , que miden la posición de cada articulación. Las medidas de posición son comparadas con las posiciones a donde se desea ir, y cualquier diferencia es corregida con señales enviadas a los apropiados controladores de articulaciones.

La capacidad de actuación de robots por servomecanismos se ve acrecentada por el hecho de añadir memoria electrónica para almacenar programas y circuitos de control digital Cada robot se programa para guiarse de una secuencia de posiciones prefijadas Un tutor humano usa una caja manual de control, con un botón de control de velocidad que le permite controlar cada articulación del robot El programador humano usa una caja de control para guiar el robot a la posición deseada para cada paso del programa. A esto se le conoce como **aprendizaje manual**

Cuando el programa debe volver hacia atrás, el sistema de control simplemente ordena a cada articulación moverse a la posición recordada por cada paso Una vez que el robot se encuentra dentro de la operación de la línea de producción, repite el programa almacenado una y otra vez, moviéndose desde uno de los puntos almacenados hasta el siguiente, de acuerdo con el ciclo prefijado y cronometrado, así hasta la finalización con el último paso o como respuesta a una señal de reloj externa a la máquina. Este tipo de robots se denomina **robot punto a punto** ya que el camino exacto que el robot define es únicamente unos cuantos puntos seleccionados

La utilización de una computadora con capacidades aritméticas y lógicas hace posible el implementar altos niveles de control Con una computadora es posible programar al robot para mover su brazo en líneas rectas, o a lo largo de otros caminos geométricos, entre los puntos memorizados.

Un planteamiento más general para resolver los problemas de comunicación hombre-robot es la utilización de programación de alto nivel. Un **lenguaje de control robótico** es un lenguaje de computadora específicamente diseñado para controlar un robot Además de contener las ordenes esperadas, tales como control de bucle y sentencias condicionales, un lenguaje de control robótico incluye también ordenes que controlan el movimiento del robot. Es este control de movimiento lo que distingue a un lenguaje de control robótico de otros lenguajes de programación de propósitos generales Un lenguaje de control robótico contiene una base de datos predefinida que se usa para obtener información espacial acerca de cada movimiento que el robot realizará

Es importante comprender que un lenguaje de control robótico no esta diseñado para reemplazar al aprendizaje manual, sino más bien para complementarlo. Por consiguiente, un lenguaje de control robótico debe proveer un interfaz al aprendizaje manual El método típico de enseñar al robot la información espacial necesaria es por medio del uso del aprendizaje manual, y luego el lenguaje de control para preescribir la forma en que el robot debería usar esa información. Generalmente, se da un nombre simbólico a cada localización especifica, para que el programa pueda referirse a ella

Muchos robots industriales se utilizan ampliamente en tareas de fabricación y de ensamblaje, tales como manejo de material, soldaduras por arco y de punto, montajes de piezas, pinturas al spray, carga y descarga de máquinas controladas numéricamente, exploraciones espaciales y submarinas y el manejo de materiales peligrosos Estos robots definen cuatro movimientos·

- Coordenadas Cartesianas (3 ejes lineales).
- Coordenadas Cilíndricas (dos ejes lineales y un eje rotacional).
- Coordenadas Esféricas (Un eje lineal y dos ejes rotacionales).
- Coordenadas de Revolución o articulados (3 ejes rotacionales)

El problema de control de un robot se puede dividir en dos problemas coherentes· el subproblema de planificación de movimiento (o trayectoria) y el subproblema del control del movimiento.

La curva espacial que la mano del robot sigue desde una localización inicial (posición y orientación) hasta un final se llama **trayectoria o camino** La planificación de la trayectoria (o planificador de trayectoria) interpola y/o aproxima la trayectoria deseada por una clase de funciones polinomiales y genera una secuencia de puntos de "consignas de control" en función del tiempo para movimiento del robot desde la posición inicial hasta el destino

En general, el problema de control de movimientos consiste en 1) Obtener los modelos dinámicos del robot, 2) Utilizar estos modelos para determinar leyes o estrategias de control para conseguir la respuesta en el funcionamiento del sistema deseado.

En el campo de la robótica, la mayoría de los esfuerzos se han aplicado a crear y mejorar robots industriales de ensamblaje Como estos robots se usan en un entorno controlado, pueden ser considerablemente menos inteligentes que los robots autónomos

Por ahora, los robots industriales sólo pueden desempeñar aquellas tareas para las que han sido explícitamente programados

Robots Autónomos

Un **robot autónomo** es mucho más complejo que los robots industriales porque debe ser mucho más listo Si un robot autónomo va a operar con éxito en el entorno incontrolado del mundo real, necesitará varias habilidades que el robot industrial no requería, por ejemplo, necesitará sensores que le permitan oír y ver y debe comprender el lenguaje natural y lo que significa. Dotar al robot de estas dos últimas capacidades no es una proeza pequeña Además, el robot debe ser capaz de resolver problemas, lo cual es quizá la tarea de programación más complicada de todas. Esto es necesario para que el robot pueda adaptarse a varias situaciones porque claramente no se puede programar un robot por adelantado para cada posible suceso.

El mayor obstáculo que impide la creación de un robot autónomo es el hecho de que los programadores no hayan desarrollado todavía las técnicas de software necesarias.

Sensores del Robot

La utilización de **mecanismos sensores** externos permiten a un robot interaccionar con su entorno de una manera flexible, esto esta en contraste con operaciones preprogramadas en las cuales a un robot se le "enseña" para efectuar tareas repetitivas mediante un conjunto de funciones programadas, la cual es la forma más predominante de operación de los robots industriales actuales. La utilización de tecnología sensorial para dotar a las máquinas con un grado mayor de inteligencia a tratar con su entorno es realmente un tema de investigación y desarrollo activo en campo de la robótica.

La función de los sensores del robot se pueden dividir en dos categorías principales: **estado interno** y **estado externo** Los sensores del estado interno tratan con la detección de variables tales como la posición de la articulación del brazo, que se utiliza para controlar el robot. Por otra parte, los sensores de estado externo tratan con la detección de variables tales como alcance, proximidad y contacto Los sensores externos se utilizan para guiado de robots, así como para la identificación y manejo de objetos

Inteligencia del Robot

Un problema básico en robótica es la planificación de movimiento para resolver alguna tarea preespecificada, y luego controlar al robot cuando ejecuta las ordenes necesarias para conseguir esas acciones **Planificación** significa decidir un curso de acción antes de actuar Un plan es así una representación de un curso de acción para lograr un objetivo dado.

Un planificador de robot intenta encontrar una trayectoria desde el mundo del robot inicial hasta un mundo del robot final. El camino consiste en una secuencia de operaciones que se consideran primitivas para el sistema Una solución a un problema podría ser la base de una secuencia correspondiente de acciones físicas en el mundo físico. La planificación de robots, que proporciona la inteligencia y la capacidad de resolución de problemas a un sistema robótico, es un area de investigación

Aplicación

Realizar un simulador de robot que utilice el sistema de visión implementado en el capítulo anterior, para guardar adecuadamente el objto reconocido de un conjunto de diez muestras en la caja correspondiente

NOTA: Ver código del programa en el Anexo 5 Roboótica.

Pies de página

(1) Sistemas Expertos Una metodología de programación, Sánchez y Beltrán Juan Pablo; pag 18
(2) Sistemas Expertos, Castillo Enrique y Alvarez Elena, pag. 14
(3) Inteligencia Artificial; Rich Elaine, pag. 439
(4) Natural Language Processing The PLNLP Approach, JENSEN, Karen, pag. 26
(5) Robótica control, detección, visión e inteligencia. González R C., pag.1

Inteligencia Artificial.
RICH, Elaine.
Ed Mc Graw-Hill
España, 1997
2a ed.
703 pp

Principios de Inteligencia Artificial y Sistemas Expertos
ROLSTON, David W
Ed Mc Graw-Hill
Colombia, 1990.
255 pp

Sistemas Expertos Una Metodología de Programación.
SÁNCHEZ y Beltrán Juan Pablo.
Ed Macrobit
USA, 1990
261 pp

Turbo Prolog Programación Avanzada.
SCHILDT, Herbert.
Ed Mc Graw-Hill
México, 1990
313 pp

Biblioteca de Informática.
Volumen 7.
Ed. Limusa, Grupo Noriega Editores
México 1992
2435 pp

Anexos

Código de Programas

Anexo

1

Búsqueda de Soluciones

```
/* Encuentra la ruta de un lugar turistico en Tabasco */

database
   camino(symbol, symbol)
   visitado(symbol)

predicates
   encontrar_salida
   mover(symbol, symbol)
   añadir_camino(symbol)
   borrar
   despliega(symbol)
   ruta(symbol,symbol)

goal
   assert(camino(villahermosa,nacajuca)),
   assert(camino(nacajuca,jalpa_de_mendez)),
   assert(camino(jalpa_de_mendez,comalcalco)),
   assert(camino(comalcalco,paraiso)),
   assert(camino(paraiso,frontera)),
   assert(camino(villahermosa,macuspana)),
   assert(camino(macuspana,emiliano_zapata)),
   assert(camino(emiliano_zapata,tenosique)),
   assert(camino(villahermosa,jose_colomo)),
   assert(camino(jose_colomo,jonuta)),
   assert(camino(jonuta,frontera)),
   assert(camino(jonuta,mactun)),
   assert(camino(mactun,tenosique)),
   makewindow(1,112,113," BUSQUEDA DE SOLUCIONES ",0,0,25,80),
   encontrar_salida, nl,
   write("Desea encontrar otra solucion s/n "),
   readln(R),
   R=n
```

```
clauses
   encontrar_salida:-
     makewindow(2,48,135," Lugares turisticos de Tabasco ",3,3,20,34),nl,
     write("  Partiendo de Villahermosa"),nl,
     write("  A dond, quiere dirigirse?"),nl,nl,
     write("    comalcalco"), nl,
     write("    emiliano_zapata"), nl,
     write("    frontera"), nl,
     write("    jalpa_de_mendez"), nl,
     write("    jonuta"), nl,
     write("    jose_colomo"), nl,
     write("    mactun"), nl,
     write("    macuspana"), nl,
     write("    nacajuca"), nl,
     write("    paraiso"), nl,
     write("    tenosique"), nl,
     makewindow(3,48,7,"",3,42,20,35),
     write("Lugar a donde se quiere dirigir  "), nl,
     readln(Lugar),
     ruta(villahermosa,Lugar),
     not(despliega(Lugar))

   encontrar_salida :-
     write("No existen mas soluciones o "),nl,
     write("No existe el lugar"), fail

   ruta(A,B) :-
     mover(A,B).

   ruta(_,_) :-
     borrar

   /* intenta un movimiento cuando el camino es directo */
   mover(T,T2):-
     camino(T,T2),
     añadir_camino(T)

   /* intenta un movimiento cuando el camino no es directo */
   mover(T,T2):-
     camino(T,X),
     X<>T2,
     añadir_camino(T),
     mover(X,T2).

   /* regresa un nivel antes, ya que encontro un camino sin salida */
   mover(_,_) :-
     borrar, fail.
```

```
añadir_camino(T):-
  not(visitado(T)),
  assert(visitado(T)), !

añadir_camino(_)

/*borra la base de datos visitado cuando no encuentra un camino */
borrar :-
  retract(visitado(_)),
  fail,!

/* muestra el camino recorrido */
despliega(Lugar) :-
  assert(visitado(Lugar)),
  write("La ruta es  "), nl,
  visitado(A),
  write(A), nl,
  fail,!
```

Anexo

2

Sistemas Expertos

```
/* Suguiere lugares turísticos de la República Mexicana */


predicates
   visita (symbol)
   estado_info (symbol, symbol)
   final (symbol)
   inicio

goal
   inicio

clauses
   final (X) .- X=s. inicio

   final (_) .- nl,write ("              Buen viaje !!!")

   inicio -
      makewindow(1,115,121,"SISTEMA EXPERTO",0,0,25,80),
      makewindow(2,95,121," Que tipo de lugar le gustaria visitar ",1,1,23,42),
      write("  playa colonial zona_arqueologica ").nl,nl,
      write("      Lugar "),
      makewindow(3,112,0,"",4,19,1,20),nl,
      readln (Lugar),
      visita (Lugar),
      shiftwindow(2),
      cursor(20,3),
      write (" Desea realizar otro viaje s/n ?"),
      readln(R),
      final (R)

   visita(X) - X=playa,
            shiftwindow(2),
            clearwindow,
            write("Estados de la república que cuentan con playas:"),nl,nl,
            write("baja_california    baja_california_sur"),nl,
```

```
        write("campeche          colima"),nl,
        write("chiapas           guerrero"),nl,
        write("jalisco           nayarit"),nl,
        write("quintana_roo      sinaloa"),nl,
        write("sonora            tamaulipas"),nl,nl,
        write("Elija un estado para mas referencia: "),nl,
        readln(Estado),
        makewindow(4,95,121,"PERFIL DEL ESTADO",1,45,15,34),
        write(Estado,":"),nl,nl,
        estado_info(Estado, X)

visita(X) :- X=colonial,
        shiftwindow(2),
        clearwindow,
        write("Estados de la república con ciudades coloniales "),nl,nl,
        write("aguascalientes    coahuila "),nl,
        write("colima            distrito_federal"),nl,
        write("durango           estado_de_mexico"),nl,
        write("guanajuato        guerrero"),nl,
        write("hidalgo           jalisco"),nl,
        write("michoacan         morelos"),nl,
        write("nuevo_leon        oaxaca"),nl,
        write("puebla            queretaro"),nl,
        write("san_luis_potosi   tabasco"),nl,
        write("tlaxcala          veracruz"),nl,
        write("yucatan           zacatecas"),nl,nl,
        write("Elija un estado para mas referencia  "),nl,
        readln(Estado),
        makewindow(4,95,121,"PERFIL DEL ESTADO",1,45,15,34),
        write(Estado," "),nl,nl,
        estado_info(Estado,X).

visita(X) :- X=zona_arqueologica,
        shiftwindow(2),
        clearwindow,
        write("Estados de la república que tienen zonas arqueologicas."),nl,nl,
        write("campeche          chiapas"),nl,
        write("chihuahua         distrito_federal"),nl,
        write("estado_de_mexico  guerrero"),nl,
        write("hidalgo           morelos"),nl,
        write("nayarit           nuevo_leon"),nl,
        write("oaxaca            puebla"),nl,
        write("queretaro         quintana_roo"),nl,
        write("san_luis_potosi   tabasco"),nl,
        write("tlaxcala          tamaulipas"),nl,
        write("veracruz          yucatan"),nl,
        write("zacatecas"),nl,nl,
```

```
        write("Elija un estado para mas referencia· "),nl,
        readln(Estado),
        makewindow(4,95,121,"PERFIL DEL ESTADO",1,45,15,34),
        write(Estado," "),nl,nl,
        estado_info(Estado,X)

visita(_) -
        shiftwindow(2),
        clearwindow,
        cursor(10,5),nl,
        write("Introdujo un lugar no valido"),nl,
        write("Presione cualquier tecla para continuar    "),
        readchar(_),
        inicio.

estado_info(Estado,X) if Estado=aguascalientes,
                write("Estado con ciudades coloniales y lugares para practicar actividades
acuáticas"),
        visita (X)

estado_info(Estado,X) if Estado=baja_california,
        write ("Conformado de playas en las que se practican actividades acuáticas"),
        visita (X)

estado_info(Estado,X) if Estado=baja_california_sur,
        write ("Conformado de playas en las que se practican actividades acuáticas y lugares
para
                la cacería"),
        visita (X).

estado_info(Estado,X) if Estado=campeche,
        write ("Cuenta con zonas arqueológicas, playas y lugares para la cacería"),
        visita (X)

estado_info(Estado,X) if Estado=coahuila,
        write ("Estado con ciudades coloniales y lugares para la caza y pesca"),
        visita (X).

estado_info(Estado,X) if Estado=colima,
        write ("Estado con ciudades coloniales y playas en las que se practican actividades
                acuáticas"),
        visita (X)

estado_info(Estado,X) if Estado=chiapas,
        write ("Cuenta con zonas arqueologicas, playas, grutas y cascadas"),
        visita (X)
```

```
estado_info(Estado,X) if Estado=chihuahua,
        write ("Cuenta con zonas arqueológicas, grutas y manantiales, lugares para la caza y
              practica de actividades acuáticas"),
        visita (X)

estado_info(Estado,X) if Estado=distrito_federal,
        write ("Estado con ciudades coloniales y zonas arqueológicas"),
        visita (X)
estado_info(Estado,X) if Estado=durango,
        write ("Estado con ciudades coloniales y lugares para la caza y pesca"),
        visita (X)

estado_info(Estado,X) if Estado=guanajuato,
        write ("Estado con ciudades coloniales y lugares con aguas termales"),
        visita (X)

estado_info(Estado,X) if Estado=guerrero,
        write ("Cuenta con playas, zonas arqueológicas, grutas y ciudades coloniales"),
        visita (X)

estado_info(Estado,X) if Estado=hidalgo,
        write ("Estado con ciudades coloniales, zonas arqueológicas y lagos"),
        visita (X)

estado_info(Estado,X) if Estado=jalisco,
          write ("Cuenta con ciudades coloniales, playas y cascadas, lugares para la caza y
practica
              de actividades acuáticas"),
        visita (X)

estado_info(Estado,X) if Estado=michoacan,
        write ("Estado con ciudades coloniales, lagos y bosques"),
        visita (X).

estado_info(Estado,X) if Estado=morelos,
        write ("Estados con ciudades coloniales, lagos y bosques"),
        visita (X)

estado_info(Estado,X) if Estado=nayarit,
        write ("Cuenta con zonas arqueológicas, playas, lagos y lugares para la caza"),
        visita (X)

estado_info(Estado,X) if Estado=nuevo_leon,
        write ("Estado con ciudades coloniales, zonas arqueológicas, aguas termales y grutas,
              lugares para la caza y practica de actividades acuáicas"),
        visita (X)
```

```
estado_info(Estado,X) if Estado=oaxaca,
        write ("Estado con ciudades coloniales, zonas arqueolgicas, lagos y playas"),
        visita (X)

estado_info(Estado,X) if Estado=puebla,
        write ("Estado con ciudades coloniales, zonas arqueológicas, bosques, manantiales,
aguas
            termales, volcanes y lugares para la caza"),
        visita (X)

estado_info(Estado,X) if Estado=queretaro,
        write ("Cuenta con zonas arqueolgicas y ciudades coloniales"),
        visita (X).

estado_info(Estado,X) if Estado=quintana_roo,
        write ("Cuenta con playas, lagos y zonas arqueolgicas"),
        visita (X)

estado_info(Estado,X) if Estado=san_luis_potosi,
        write ("Cuenta con zonas arqueológicas, manantiales, grutas, cascadas, aguas
termales,
            lugares para la caza y practica de actividades acuáticas"),
        visita (X).

estado_info(Estado,X) if Estado=sinaloa,
        write ("Conformado de playas en las que se practican actividades acuáicas"),
        visita (X)

estado_info(Estado,X) if Estado=sonora,
        write ("Cuenta con ciudades coloniales, playas, desierto y lugares para la practica de
            actividades acuáticas"),
        visita (X)

estado_info(Estado,X) if Estado=tabasco,
        write ("Estado con ciudades coloniales, zonas arqueológicas y lagos"),
        visita (X)

estado_info(Estado,X) if Estado=tamaulipas,
        write ("Cuenta con zonas arqueológicas, playas, grutas, manantiales, lugares para la
caza y
            practica de actividades acuáticas"),
        visita (X)

estado_info(Estado,X) if Estado=tlaxcala,
        write ("Cuenta con zonas arqueológicas y ciudades coloniales"),
        visita (X)
```

```
estado_info(Estado,X) if Estado=veracruz,
        write ("Estado con zonas arqueológicas, playas, lagos y ciudades coloniales"),
        visita (X)

estado_info(Estado,X) if Estado=yucatan.
        write ("Estado con zonas arqueológicas, playas, lagos y ciudades coloniales"),
        visita (X)

estado_info(Estado,X) if Estado=zacatecas,
        write ("Cuenta con zonas arqueológicas y ciudades coloniales"),
        visita (X)

estado_info(_,_)
```

Aenxo

3

Procesamiento de Lenguaje Natural

```
/* Despliega el nombre y número telefónico solicitado en lenguaje natural*/


database
   palabra(symbol,symbol)
   telefono(symbol,real)

predicates
   siguiente_palabra(string,string,symbol)
   encontrar_delim(string,integer,integer)
   quitar_espacios(string,string)
   inicio
   procesar(string)
   obtener_orden(string,string,symbol)
   obtener_nombre(string,string,symbol)
   obtener_modificador(string,string,symbol)
   obtener_ruido(string,string,symbol)
   terminador(string)
   realizar(symbol,symbol,symbol)
   borrar
   final(symbol)

goal
   assert(palabra(orden,dame)),
   assert(palabra(orden,quiero)),
   assert(palabra(orden,necesito)),
   assert(palabra(nombre,pilar)),
   assert(palabra(nombre,maribel)),
   assert(palabra(nombre,hector)),
   assert(palabra(nombre,claudia)),
   assert(palabra(nombre,nora)),
   assert(palabra(nombre,laura)),
   assert(palabra(nombre,marco)),
   assert(palabra(nombre,gilberto)),
   assert(palabra(nombre,amilcar)),
   assert(palabra(nombre,carolina)),
```

```
    assert(palabra(nombre,antonio)),
    assert(palabra(nombre,eduardo)),
    assert(palabra(nombre,arturo)),
    assert(palabra(nombre,estela)),
    assert(palabra(nombre,patricia)),
    assert(palabra(nombre,noemi)),
    assert(palabra(nombre,magda)),
    assert(palabra(nombre,elizabeth)),
    assert(palabra(nombre,liliana)),
    assert(palabra(nombre,argelia)),
    assert(palabra(nombre,guadalupe)),
    assert(palabra(nombre,carlos)),
    assert(palabra(modificador,todos)),
    assert(telefono(amilcar,2202020)),
    assert(telefono(antonio,7804787)),
    assert(telefono(argelia,7943715)),
    assert(telefono(arturo,8797646)),
    assert(telefono(carlos,7889381)),
    assert(telefono(carolina,8745228)),
    assert(telefono(claudia,7182181)),
    assert(telefono(eduardo,6962107)),
    assert(telefono(elizabeth,5411559)),
    assert(telefono(estela,7743662)),
    assert(telefono(gilberto,5922380)),
    assert(telefono(guadalupe,7426338)),
    assert(telefono(hector,6499694)),
    assert(telefono(laura,3928216)),
    assert(telefono(liliana,8556158)),
    assert(telefono(magda,7803106)),
    assert(telefono(marco,6138364)),
    assert(telefono(maribel,6261020)),
    assert(telefono(noemi,7885318)),
    assert(telefono(nora,3919376)),
    assert(telefono(patricia,7326788)),
    assert(telefono(pilar,7630454)),
    inicio.

clauses
    inicio -
    makewindow(1,31,117," PROCESAMIENTO DE LENGUAJE NATURAL ",0,0,25,80),
    cursor(1,1),
    date(Year,Month,Day),
    write(Day,"/","0",Month,"/",Year),
    cursor(1,69),
    time(Hours,Minutes,Seconds,_),
    write(Hours,":",Minutes," ",Seconds),
    makewindow(4,113,117," DIRECTORIO TELEFONICO ",5,15,15,50),
```

```
write("A quien quieres llamar .? "),nl,
readln(Persona),
procesar(Persona),
sound(20,262),
sound(20,262),
sound(20,262),
cursor(6,4),
write("Deseas llamar a alguien mas (s/n) ? "),
readln(R),
final(R)

inicio -
 nl,
 write(" DIRECTORIO TELEFONICO CERRADO "),
 sound(20,139),
 sound(20,139),
 sound(20,139).

final(X) .-
 X=s, inicio
 final(_) -
 borrar

/* Proceso para analizar oración*/
procesar(Persona) -
 obtener_orden(Persona,S2,C),
 obtener_modificador(S2,S3,M),
 obtener_nombre(S3,_,N),
 realizar(C,M,N).

/* Proceso para obtener las palabras claves de la oración*/
obtener_orden(Persona,S2,C) -
 siguiente_palabra(Persona,S2,C),
 palabra(orden,C)

obtener_orden(Persona,S2,C) -
 obtener_ruido(Persona,S1,_),
 obtener_orden(S1,S2,C)

obtener_orden(Persona,S2,C) .-
 S2=Persona,
 C=dame.

obtener_modificador(Persona,S2,M) -
 siguiente_palabra(Persona,S2,M),
 palabra(modificador,M).
```

```
obtener_modificador(Persona,S2,M) -
  obtener_ruido(Persona,S1,_),
  obtener_modificador(S1,S2,M).

obtener_modificador(Persona,S2,M) -
  S2=Persona,
  M=todos.

obtener_nombre(Persona,S2,N) -
  siguiente_palabra(Persona,S2,N),
  palabra(nombre,N)

obtener_nombre(Persona,S2,N) -
  obtener_ruido(Persona,S1,_),
  obtener_nombre(S1,S2,N)

obtener_nombre(Persona,Persona,N) -
  N=ninguno

/* Proceso para eliminar palabras sin efecto en la oración */
obtener_ruido(Persona,S2,X) -
  siguiente_palabra(Persona,S2,X),
  not(palabra(_,X))

/* Proceso para encontrar el termino de la oración */
terminador(Persona) -
  frontchar(Persona,CH,_),
  CH=' '

/* Proceso que ejecuta la orden solicitada */
realizar(dame,_,N) -
  telefono(N,P),nl,
  write("El número telefónico de ",N," es ",P),
  nl

realizar(dame,todos,_) -
  sound(20,220),
  sound(20,233)
  sound(20,247),
  sound(20,262),
  write("Lo siento no se encuentra en el directorio"),nl,
  write("vuelve a intentarlo.") .

realizar(quiero,_,N) -
  telefono(N,P),nl,
  write("El número telefónico de ",N," es ",P),
  nl
```

```
realizar(quiero,todos,_) -
  sound(20,220),
  sound(20,233),
  sound(20,247),
  sound(20,262),
  write("Lo siento no se encuentra en el directorio"),nl,
  write("vuelve a intentarlo ")
realizar(necesito,_,N) -
  telefono(N,P),nl,
  write("El n£mero telef¢nico de ",N," es ",P),
  nl.

realizar(necesito,todos,_) -
  sound(20,220),
  sound(20,233),
  sound(20,247),
  sound(20,262),
  write("Lo siento no se encuentra en el directorio"),nl,
  write("vuelve a intentarlo ")

realizar(_,_,_).

/* Proceso para encontrar palabras de la oraci¢n */
encontrar_delim(Persona,Count,C) -
  frontchar(Persona,CH,S2),
  CH<>' ', CH<>',
  C2=C+1,

encontrar_delim(S2,Count,C2)

encontrar_delim(_,Count,Count)

siguiente_palabra(Persona,S2,W) -
  encontrar_delim(Persona,Count,0),!,
  Count>0,
  frontstr(Count,Persona,W,S3),
  quitar_espacios(S3,S2).

quitar_espacios(Persona,S2) -
  frontchar(Persona,Ch,S2),
  Ch=' '

quitar_espacios(Persona,Persona)
```

Anexo

4

Percepción Visual

/*Reconoce un objeto en pantalla (triangulo o cuadrado) utilizando el Reconocedor Delta-D*/

```
database
   punto(integer,integer)
   incrementos(integer,integer)
   cambios(symbol)

predicates
   inicio
   ventana
   hacer_triangulo(integer,integer,integer)
   hacer_izquierda(integer,integer,integer)
   hacer_derecha(integer,integer,integer)
   hacer_base(integer,integer,integer)
   hacer_cuadrado(integer,integer,integer)
   hacer_arriba_abajo(integer,integer,integer)
   hacer_derechalado(integer,integer,integer)
   hacer_izquierdalado(integer,integer,integer)
   borra(integer,integer)
   borra_puntos
   dibuja
   haz_dibujo(real)
   reconocer
   encontrar_figura(integer,integer)
   encontrar_punto(integer,integer,integer,integer)
   es_figura(integer,integer)
   seguir_forma(integer,integer)
   seguir(integer,integer,integer,integer)
   resultado(integer)

goal
   assert(incrementos(1,1)),
   assert(incrementos(-1,1)),
   assert(incrementos(1,-1)),
   assert(incrementos(1,0)),
   assert(incrementos(0,1)),
```

```
    assert(incrementos(-1,-1)),
    assert(incrementos(-1,0)),
    assert(incrementos(0,-1)),
    inicio

clauses
    inicio -
      ventana,
      dibuja,
      reconocer

    /*Intenta reconocer el objeto en pantalla*/
    reconocer -
      encontrar_figura(6,30),
      borra_puntos

    reconocer

    encontrar_figura(R,C) -
      encontrar_punto(R,C,A,B),
      es_figura(A,B)

    /*Encuentra el punto de inicio del objeto en pantalla*/
    encontrar_punto(A,B,R,C) -
      scr_char(A,B,Ch),
      Ch='*',
      R=A,C=B

    encontrar_punto(A,B,R,C) -
      B1=B+1,
      B1<60,!,
      encontrar_punto(A,B1,R,C)

    encontrar_punto(A,_,R,C) -
      A1=A+1,
      A1<20,!,
      encontrar_punto(A1,30,R,C)

    encontrar_punto(_,_,_,_) :- fail.

    es_figura(R,C) -
      seguir_forma(R,C),
      borra(0,N),!,
      resultado(N).

    es_figura(_,_) :-
      borra(0,_),fail
```

```
/*Sigue el contorno hasta regresar al punto inicial contando el numero de cambios*/
seguir_forma(R,C) .-
 assert(punto(R,C)),
 R1=R+1,
 scr_char(R1,C,Ch),
 Ch='*',
 seguir(R1,C,1,0),!

seguir_forma(R,C) -
 R1=R+1,
 C1=C+1,
 scr_char(R1,C1,Ch),
 Ch='*',
 seguir(R1,C1,1,1),!.

/*Sigue cada linea*/
seguir(R,C,_,_) -
 punto(Y,X),
 R=Y, C=X        /*verifica si regresa al inicio de la figura*/

seguir(R,C,I,J) .-  /*sigue en la misma direccion*/
 R1=R+I,
 C1=C+J,
 scr_char(R1,C1,Ch),
 Ch='*',
 assert(punto(R,C)),
 seguir(R1,C1,I,J)

seguir(R,C,_,_) .-
 assert(cambios(once)),   /*encuentra el cambio de direccion*/
 incrementos(I,J),
 R1=R+I,
 C1=C+J,
 not(punto(R1,C1)),
 scr_char(R1,C1,Ch),
 Ch='*',

seguir(R1,C1,I,J).

/*Borra el contenido de la base cambios y devuelve el numero de cambios de direccion que
  encontro en el objeto en pantalla*/
borra(Cont,N) -
 retract(cambios(_)),
 N1=Cont+1,
 borra(N1,N).
```

```
borra(Cont,N) -
  N=Cont

/*Despliega el nombre del objeto que se despliega en pantalla*/
resultado(N) -
  N=2,
  cursor(16,21),
  write("El objeto en pantalla es un triángulo")

resultado(N) -
  N=3,
  cursor(16,21),
  write("El objeto en pantalla es un cuadrado")

resultado(_) -
  cursor(16,21),
  write("No se puede reconocer el objeto en pantalla").

/*Borra el contenido de la base de datos puntos*/
borra_puntos -
  retract(punto(_,_)),
  fail

borra_puntos

/*Dibuja un triangulo o cuadrado aleatoriamente en la pantalla aleatoriamente*/
dibuja -
  random(X),
  haz_dibujo(X)

haz_dibujo(X) -
  X>0 5,
  hacer_cuadrado(7,35,5)

haz_dibujo(_) -
  hacer_triangulo(7,40,5)

hacer_triangulo(R,C,N) -
  hacer_izquierda(R,C,N),
  hacer_derecha(R,C,N),
  hacer_base(R,C,N)

hacer_izquierda(R,C,N) -
  N<>0,
  cursor(R,C),
  write("*"),
  R1=R+1,
```

```
  C1=C-1,
  N1=N-1,
  hacer_izquierda(R1,C1,N1)
hacer_izquierda(_,_,_)

hacer_derecha(R,C,N) :-
  N<>0,
  cursor(R,C),
  write("*"),
  R1=R+1,
  C1=C+1,
  N1=N-1,
  hacer_derecha(R1,C1,N1)

hacer_derecha(_,_,_)

hacer_base(R,C,N) :-
  R1=R+N,
  C1=C-N,
  cursor(R1,C1),
  write("***********")

hacer_cuadrado(R,C,N) :-
  hacer_arriba_abajo(R,C,N),
  hacer_izquierdalado(R,C,N),
  hacer_derechalado(R,C,N)

hacer_arriba_abajo(R,C,N) :-
  cursor(R,C),
  write("***********"),
  R1=R+N,
  cursor(R1,C),
  write("***********")

hacer_izquierdalado(R,C,N) :-
  N<>0,
  cursor(R,C),
  write("*"),
  N1=N-1,
  R1=R+1,
  hacer_izquierdalado(R1,C,N1)

hacer_izquierdalado(_,_,_).
```

```
hacer_derechalado(R,C,N) -
 N<>0,
 C1=C+9,
 cursor(R,C1),
 write("*"),
 N1=N-1,
 R1=R+1,
 hacer_derechalado(R1,C,N1)


hacer_derechalado(_,_,_)

ventana -
 makewindow(1,95,117," PERCEPCION VISUAL ",0,0,25,80)
```

Anexo

5

Robótica

/*Simula colocar un objeto en el lugar adecuado de un conjunto de 10 muestras (triángulos y/o cuadrados) utilizando el Reconocedor Delta-D*/

```
database
   punto(integer,integer)
   incrementos(integer,integer)
   cambios(symbol)
   cant_tria(symbol)
   cant_cua(symbol)

predicates
   inicio
   ventana
   hacer_triangulo(integer,integer,integer)
   hacer_izquierda(integer,integer,integer)
   hacer_derecha(integer,integer,integer)
   hacer_base(integer,integer,integer)
   hacer_cuadrado(integer,integer,integer)
   hacer_arriba_abajo(integer,integer,integer)
   hacer_derechalado(integer,integer,integer)
   hacer_izquierdalado(integer,integer,integer)
   borra(integer,integer)
   borra_puntos
   dibuja
   haz_dibujo(real)
   control(integer)
   reconocer
   verificar
   encontrar_figura(integer,integer)
   encontrar_punto(integer,integer,integer,integer)
   es_figura(integer,integer)
   seguir_forma(integer,integer)
   seguir(integer,integer,integer,integer)
   resultado(integer)
   cuenta_bolsa(integer,integer)
   cuenta_bolsa1(integer,integer)
   delay(integer)
```

```
goal
   assert(incrementos(1,1)),
   assert(incrementos(-1,1)),
   assert(incrementos(1,-1)),
   assert(incrementos(1,0)),
   assert(incrementos(0,1)),
   assert(incrementos(-1,-1)),
   assert(incrementos(-1,0)),
   assert(incrementos(0,-1)),
   inicio

clauses
   inicio -
     control(1)

   control(Limit) -
     Limit<11,
     delay(1000),
     reconocer,
     cursor(1,75),
     write(Limit),
     sound(30,220),
     Cuenta=Limit+1,
     control(Cuenta).

   control(_) -
     verificar,
     cursor(19,30)

   delay(N) -
     N>0,!,N1=N-1,delay(N1)
     delay(0)

   /*Intenta reconocer el objeto en pantalla*/
   reconocer -
     ventana,
     dibuja,
     encontrar_figura(6,30),
     borra_puntos

   verificar -
     cuenta_bolsa(0,Cta_bolsa),
     cursor(15,5),
     write("El numero de triangulos"),
     cursor(16,5),
     write("  que guardo el robot"),
```

```
  cursor(17,5),
  write("   en la caja es ",Cta_bolsa),
  cuenta_bolsa1(0,Cta_bolsa1),
  cursor(15,51),
  write("El numero de cuadrados"),
  cursor(16,51),
  write(" que guardo el robot"),
  cursor(17,51),
  write("   en la caja es ",Cta_bolsa1)

cuenta_bolsa(Ctab,Cta_bolsa) :-
  retract(cant_tria(_)),
  Tria=Ctab+1,
  cuenta_bolsa(Tria,Cta_bolsa)

cuenta_bolsa(Ctab,Cta_bolsa) :-
  Cta_bolsa=Ctab

cuenta_bolsa1(Ctab1,Cta_bolsa1) :-
  retract(cant_cua(_)),
  Cuadra=Ctab1+1,
  cuenta_bolsa1(Cuadra,Cta_bolsa1)

cuenta_bolsa1(Ctab1,Cta_bolsa1) :-
  Cta_bolsa1=Ctab1

encontrar_figura(R,C) :-
  encontrar_punto(R,C,A,B),
  es_figura(A,B)

/*Encuentra el punto de inicio del objeto en pantalla*/
encontrar_punto(A,B,R,C) :-
  scr_char(A,B,Ch),
  Ch='*',
  R=A,C=B

encontrar_punto(A,B,R,C) :-
  B1=B+1,
  B1<60,!,

encontrar_punto(A,B1,R,C)

encontrar_punto(A,_,R,C) :-
  A1=A+1,
  A1<20,!,
  encontrar_punto(A1,30,R,C)
```

```
encontrar_punto(_,_,_,_) .- fail

es_figura(R,C) -
  seguir_forma(R,C),
  borra(0,N),!,
  resultado(N)

es_figura(_,_)

/*Sigue el contorno hasta regresar al punto inicial contando el numero de cambios*/
seguir_forma(R,C) .-
  assert(punto(R,C)),
  R1=R+1,
  scr_char(R1,C,Ch),
  Ch=='*',
  seguir(R1,C,1,0),!.

seguir_forma(R,C) -
  R1=R+1,
  C1=C+1,
  scr_char(R1,C1,Ch),
  Ch=='*',
  seguir(R1,C1,1,1),!.

/*Sigue cada linea*/
seguir(R,C,_,_) -
  punto(Y,X),
  R=Y, C=X.      /*verifica si regresa al inicio de la figura*/

seguir(R,C,I,J) -  /*sigue en la misma direccion*/
  R1=R+I,
  C1=C+J,
  scr_char(R1,C1,Ch),
  Ch=='*',
  assert(punto(R,C)),
  seguir(R1,C1,I,J).

seguir(R,C,_,_) -
  assert(cambios(once)),  /*encuentra el cambio de direccion*/
  incrementos(I,J),
  R1=R+I,
  C1=C+J,
  not(punto(R1,C1)),
  scr_char(R1,C1,Ch),
  Ch=='*',
  seguir(R1,C1,I,J)
```

```
/*Borra el contenido de la base cambios y devuelve el numero de cambios de direccion que
  encontro en el objeto en pantalla*/
borra(Cont,N) :-
 retract(cambios(_)),
 N1=Cont+1,
 borra(N1,N)

borra(Cont,N) :-
 N=Cont

/*Despliega el nombre del objeto que se despliega en pantalla*/
resultado(N) :-
 N=2,
 assert(cant_tria(bolsa))

resultado(N) :-
 N=3,
 assert(cant_cua(bolsa1))

resultado(_) :-
 cursor(20,23),
 write("No se puede reconocer el objeto en pantalla")

/*Borra el contenido de la base de datos puntos*/
borra_puntos :-
 retract(punto(_,_)),
 fail

borra_puntos

/*Dibuja un triangulo o cuadrado aleatoriamente en la pantalla aleatoriamente*/
dibuja :-
 random(X),
 haz_dibujo(X)

haz_dibujo(X) :-
 X>0.5,
hacer_cuadrado(7,35,5)

haz_dibujo(_) :-
hacer_triangulo(7,40,5)

hacer_triangulo(R,C,N) :-
 hacer_izquierda(R,C,N),
 hacer_derecha(R,C,N),
 hacer_base(R,C,N).
```

```
hacer_izquierda(R,C,N) -
 N<>0,
 cursor(R,C),
 write("*"),
 R1=R+1,
C1=C-1,
N1=N-1,
hacer_izquierda(R1,C1,N1)

hacer_izquierda(_,_,_)

hacer_derecha(R,C,N) -
 N<>0,
 cursor(R,C),
 write("*"),

 R1=R+1,
 C1=C+1,
 N1=N-1,
 hacer_derecha(R1,C1,N1)

hacer_derecha(_,_,_).

hacer_base(R,C,N) -
 R1=R+N,
 C1=C-N,
 cursor(R1,C1),
 write("**********")

hacer_cuadrado(R,C,N) .-
 hacer_arriba_abajo(R,C,N),
 hacer_izquierdalado(R,C,N),
 hacer_derechalado(R,C,N)

hacer_arriba_abajo(R,C,N) .-
 cursor(R,C),
 write("**********"),
 R1=R+N,
 cursor(R1,C),
 write("**********")

hacer_izquierdalado(R,C,N) -
 N<>0,
 cursor(R,C),
 write("*"),
 N1=N-1,
 R1=R+1,
 hacer_izquierdalado(R1,C,N1).
```

```
hacer_izquierdalado(_,_,_)

hacer_derechalado(R,C,N) -
  N<>0,
  C1=C+9,
  cursor(R,C1),
  write("*"),
  N1=N-1,
  R1=R+1,
  hacer_derechalado(R1,C,N1).

hacer_derechalado(_,_,_)

ventana -
  makewindow(1,31,113," R O B O T I C A ",0,0,25,80).
```

Conclusiones

Actualmente la Inteligencia Artificial es una fuente importante para el desarrollo de aplicaciones comerciales, debido a que muchas instituciones tanto publicas como privadas utilizan ya algunos trabajos de las áreas de la Inteligencia Artificial.

A nivel académico se tiene gran interés por la formación de profesionales y especialistas en la Inteligencia Artificial Cada vez hay más divulgación en publicaciones, conferencias, congresos y en páginas de INTERNET a fines a este tema Además de que ya se cuenta con instituciones dedicadas especialmente a la investigación y difusión de todo lo que involucra a la Inteligencia Artificial

Dentro de las áreas de la Inteligiencia Artificial, cabe mencionar que la búsqueda de soluciones sirve como apoyo para desarrollar aplicaciones para otras áreas En cuanto a los sistemas expertos las aplicaciones son más comunes, debido a que es una de las áreas más investigadas y difundidas. Con respecto a la comprensión del lengueje natural es importante ya que se facilita la comunicación entre hombre y máquina Finalmente la percepción visual y la robótica son dos áreas que se complementan, ya que por medio de la primera los robots son capaces de interactuar con su entorno de una manera inteligente

Con respecto a los lenguajes de programación, PROLOG es un lenguaje con mucho futuro, ya que por ser de tipo declartivo se adapta a las aplicaciones de la Inteligencia Artificial y es muy utilizado por muchos investigadores en el mundo, un ejemplo de ello es en la investigación de los japoneses para el proyecto de la Quinta Generación

En el presente trabajo de tesis no se mencionaron todas las áreas de la Inteligencia Artificial, debido que el objetivo del mismo es presentar una visión general de los orígenes, futuro y campos de estudio de la misma. Además de presentar a PROLOG como un lenguaje potente para el desarrollo de aplicaciones de la Inteligencia Artificial