INTELIGENCIA ARTIFICIAL

Teoría y proyectos

TERCERA EDICION

incluye software

M. C. RICARDO FUENTES COVARRUBIAS

Colima, Col., Agosto de 2002.

CAPITULO CONTENIDO PAGINA

CAPITULO 1
ANTECEDENTES

CONCEPTO DE LA INTELIGENCIA ARTIFICIAL

"Es la ciencia de hacer máquinas que hacen cosas que realizadas por el hombre requieren el uso de inteligencia". Marvin Minsky.

Breve historia de la Inteligencia Artificial.

La historia de la Inteligencia Artificial se remonta a los primeros años de la década del 40, con trabajos teóricos fundamentalmente dado el estado de la Informática en ese momento, cuando los matemáticos Warren McCullock y Walter Pitts, específicamente en 1943, desarrollan los algoritmos matemáticos necesarios para posibilitar el trabajo de clasificación, o funcionamiento en sentido general, de una red neuronal. En 1949 Donald Hebb desarrolló un algoritmo de aprendizaje para dichas redes neuronales creando, conjuntamente con los trabajos de McCullock y Pitts, la escuela coneccionista. Esta escuela se considera actualmente como el origen de lo que hoy se conoce como Inteligencia Artificial (IA).

Sin embargo el modelo coneccionista resultó poco tratado hasta hace unos años, dando paso en su lugar al razonamiento simbólico basado en reglas de producción, lo cual se conoce popularmente como Sistemas Expertos.

Otros resultados de interés obtenidos en esa época lo constituye el trabajo de Herbert Simon, un político experto en organización y estructuración de recursos humanos, el cual propone la teoría de "satisfacción". Dicha teoría planteaba, de forma concreta, la posibilidad de tomar decisiones sin tener que estudiar toda la información disponible a partir de las opciones existentes; estas decisiones se toman siguiendo las llamadas "reglas del pulgar", o *rules of thumb*, las cuales representan criterios sencillos que permiten decidir, de forma rápida, que vía de solución tomar ante un cierto problema. Estas reglas del pulgar condujeron en 1945 a las primeras nociones sobre **Heurísticas** presentada por George Polya; la heurística constituyen desde entonces uno de los conceptos fundamentales en IA.

En el año 1955 Herbert Simon, el físico Allen Newell, uno de los padres de la IA actual, y J.C. Shaw, programador de la RAND Corp. y compañero de Newell, desarrollan el primer lenguaje de programación orientado a la resolución de problemas de la IA, el IPL-11. Un año más tarde estos tres científicos desarrollan el primer programa de IA al que llamaron "Logic Theorist", el cual era capaz de demostrar teoremas matemáticos. Este programa demostró 38 de los 52 teoremas

del segundo capítulo de "Principia Mathematica" de Russel y Whitehead, uno de ellos incluso de manera más elegante a la propuesta por los autores.

En el mismo año surge la IA como disciplina de la Informática en la Conferencia de Computación de Dartmouth. En dicha conferencia se estableció como conclusión fundamental la posibilidad de simular inteligencia humana en una máquina; en nuestra opinión este sigue siendo en nuestros días el objetivo fundamental de la IA, sin embargo en aquellos años se planteó la hipótesis del sistema físico de símbolos la cual se puede traducir como:

> *"Nuestro cerebro no posee un acceso directo al mundo exterior. Solamente podemos operar sobre una representación interna suya la cual se corresponde con una colección de estructuras de símbolos. Dichas estructuras pueden tomar la forma de un patrón físico cualquiera, por ejemplo un vector de interruptores eléctricos dentro de un ordenador, o un conjunto de neuronas activadas en un cerebro biológico. Un sistema inteligente (cerebro u ordenador) puede operar sobre las estructuras con el objetivo de transformarlas en otra construcción. El pensamiento consiste en la extensión, o desarrollo, de estas estructuras, descomponiéndolas y reformándolas, destruyendo algunas y creando nuevas. La inteligencia entonces no constituye nada más que la habilidad de procesar estructuras de símbolos. Existe en un entorno diferente al hardware que le da soporte, lo trasciende y puede tomar diferentes formas físicas."*

Los centros de investigación fundamentales para la IA durante sus primeros años, de 1956 a 1963, fueron la Universidad Carnegie Mellon, el Massachusetts Institute of Technologie (o MIT), la Universidad de Stanford e IBM, siendo los temas de investigación fundamentales el desarrollo de heurísticas y el aprendizaje de máquina.

En 1957 Newell y Simon continúan su trabajo con el desarrollo del GPS o General Problems Solver. GPS era un sistema orientado a la resolución de problemas; a diferencia del "Logic Theorist", el cual se orientó a la demostración de teoremas matemáticos, GPS no estaba programado para resolver problemas de un determinado tipo, razón a la cual debe su nombre.

A finales de los años 50 y comienzos de la década del 60 se desarrolla un programa orientado a la lectura de oraciones en inglés y la extracción de conclusiones a partir de su interpretación, al cual su autor, Robert K. Lindsay, denomina "Sad Sam". Como dijimos "Sad Sam" podía leer oraciones del tipo "Jim es hermano de John" y "La madre de Jim es Mary", a partir de ella el sistema concluía que Mary debía ser también la madre de John. Este sistema representó un enorme paso de avance en el objetivo principal de la IA, la simulación de inteligencia humana por una máquina, pues era capaz de tomar una pieza de información, interpretarla, relacionarla con información anteriormente almacenada y analizada y sacar conclusiones lógicas.

En el mismo período de tiempo se reportan trabajos importantes de Herbert Gelernter, de IBM, quien desarrolla un "Demostrador Automático de Teoremas de la Geometría", así mismo Alex Bernstein desarrolla un programa para el juego de ajedrez que se considera en antepasado de "Deep Blue".

En 1961 se desarrolla SAINT (Simbolic Automatic INTegrator) por James Slagle el cual se orienta a la demostración simbólica en el área del álgebra.

En 1964 Bertrand Raphael construye el sistema SIR (Semantic Information Retrieval) el cual era capaz de comprender oraciones en inglés. A continuación mostramos un ejemplo, traducido, de interacción entre SIR y un usuario:

Usuario: Todos somos personas

SIR: Comprendo

Usuario: Un dedo es una parte de la mano

SIR: Comprendo

Usuario: Cada persona tiene dos manos

SIR: La oración anterior es ambigua. Por favor reconstrúyala.

Usuario: En cada persona hay dos manos

SIR: Comprendo

Usuario: ¿Cuantos dedos tiene John?

SIR: La oración anterior es ambigua, pero asumo que TIENE significa TIENE COMO PARTES. No se si los dedos forman parte de John.

Usuario: John es un chico

SIR: Comprendo

Usuario: ¿Cuántos dedos tiene John?

SIR: La oración anterior es ambigua, pero asumo que TIENE significa TIENE COMO PARTES, ¿cuántos dedos por mano?.

Usuario: Cada mano tiene cinco dedos.

SIR: Comprendo

Usuario: ¿Cuántos dedos tiene John?

SIR: La respuesta es diez.

En la década del 60 se comienza en el MIT el estudio de la visión artificial, lo cual implica no solo captar imágenes a través de una cámara, sino también la comprensión, por parte del ordenador, de lo que estas imágenes representan. Un resultado importante en este trabajo lo constituye el "mundo de micro-bloques", en el cual un robot era capaz de percibir un conjunto de bloques sobre una mesa, moverlos y apilarlos; el éxito se debió a los investigadores Larry Roberts, Gerald

Sussman, Adolfo Guzman, Max Clowes, David Huffman, David Waltz, Patrick Winston, y Berthold Horn.

Posteriormente se obtuvieron resultados tan importantes que eclipsaron, en parte, el éxito obtenido por GPS. Entre ellos el que quizás sea de mayor resonancia es el sistema SCHRDLU de Terry Winograd, pues permitía interrogar y dar órdenes a un robot que se movía dentro de un mundo de bloques. A continuación mostramos un ejemplo traducido:

Usuario: Toma un bloque rojo.

SCHRDLU: Okay.

Usuario: Busca un bloque más pequeño que el que tienes y ponlo dentro de la caja.

SCHRDLU: Asumo que PONLO se refiere al bloque más pequeño que el que tengo.

Usuario: ¿Qué contiene la caja?

SCHRDLU: La pirámide azul y el bloque azul.

Usuario: ¿Qué sostiene a la pirámide?

SCHRDLU: La caja.

Usuario: ¿Cuántos bloques no están sobre la mesa?

SCHRDLU: Cuatro.

Usuario: ¿Es al menos uno de ellos más estrecho que aquel que te dije que recogieras?

SCHRDLU: Sí, el cubo rojo.

Usuario: ¿Lo sostiene algo?

SCHRDLU: Sí, la mesa.

Usuario: ¿Puede la mesa recoger bloques?

SCHRDLU: No.

Usuario: La pirámide azul es mía.

SCHRDLU: Comprendo.

Usuario: Todos los bloques que no son rojos son míos, pero no poseo nada que sostenga una pirámide.

SCHRDLU: Comprendo.

Usuario: ¿Poseo una caja?

SCHRDLU: No

Usuario: ¿Poseo algo de la caja?

SCHRDLU: Si, dos cosas: El bloque azul y la pirámide azul.

Aunque parezca impresionante la capacidad del sistema para razonar y ejecutar acciones, no debemos perder de vista el hecho que el robot se mueve en un mundo muy simple de figuras geométricas, y que las relaciones entre ellas son muy limitadas. En el mundo real existen tantos objetos diferentes y relaciones entre ellos, que tratar de llevar este sistema a un entorno real resulta prácticamente imposible.

En los primeros años de la década del 60 Frank Rosemblatt desarrolla, en la Universidad de Cornell, un modelo de la mente humana a través de una red neuronal y produce un primer resultado al cual llama Perceptron. Este sistema era una extensión del modelo matemático concebido por McCullock y Pitts para las neuronas, y funcionaba basándose en el principio de "disparar" o activar neuronas a partir de un valor de entrada el cual modifica un peso asociado a la neurona, si el peso resultante sobrepasa un cierto umbral la neurona se dispara y pasa la señal a aquellas con las que está conectada. Al final, en la última capa de neuronas, aquellas que se activen definirán un patrón el cual sirve para clasificar la entrada inicial.

Este trabajo constituye la base de las redes neuronales de hoy en día, sin embargo a raíz de su desarrollo sufrió fuertes críticas por parte de Marvin Minsky y Seymour Papert lo cual provocó que la mayoría de los investigadores interesados en el tema lo abandonarán, y este no se retomara hasta los años 80.

A continuación mostramos algunos de los hechos más importantes de la IA hasta el año 1994, esto no constituye un resumen completo de todas las actividades realizadas pero sí no aporta una visión de algunos de los resultados fundamentales obtenidos a lo largo de estos años. Algunos elementos pueden parecer de poca importancia pero los hemos incluido por el papel posterior que han jugado.

1917. Karel Capek acuña el término "robot", que significa "trabajador" en checo, pero la traducción inglesa de 1923 mantiene la palabra

1928. John Von Neuman desarrolla su teorema "mínimos y máximos" utilizado posteriormente en juegos.

1930. Shannon demuestra elementos de la lógica booleana a partir de circuitos.

1931. Vannevar Bush construye el Analizador Diferencial Analógico en el MIT.

1937. Turing publica sus trabajos sobre la máquina que lleva su nombre.

1940. Atanasoff y Berry construyen el primer ordenador eléctrico, el ABC.

1941. El primer ordenador en Gran Bretaña, Robinson, basado en relays se usa para decodificar mensajes de los Nazi a partir de algoritmos de Turing.

1942. Use construye en Alemania el primer ordenador programable.

1943. McCullock y Pitts proponen la arquitectura de redes neuronales para la simulación de la inteligencia.

1944. Aiken termina la construcción de "Mark I" el primer ordenador programable en Norteamérica.

1945. Vannevar Bush publica "As we may think ...", o "Cabría imaginar ... ", en Atlantic Monthly el cual sienta las bases de lo que hoy conocemos como Hipertexto, Multimedia e Hipermedia.

1946. Eckert & Mauchley construye "ENIAC" el primer ordenador digital programable.

1949. Shannon desarrolla la Teoría de la Información base fundamental de la Informática y varias de sus áreas.

1950. Shannon propone el primer programa de ajedrez

1950. Turing publica "Computing machinery and Intelligence".

1951. Eckert & Mauchley comercializan UNIVAC el primer ordenador en venta.

1956. Newell, Shaw, y Simon crean "IPL-11" el primer lenguaje de programación para IA.

1957. Se aprueba el nombre Inteligencia Artificial para esta área en la conferencia de Darmouth.

1958. Newell, Shaw, y Simon crean "The Logic Theorist" para la resolución de problemas matemáticos.

1959. Ulam desarrolla "MANIAC I" el primer programa de ajedrez que logra derrotar a un ser humano.

1960. Chomsky escribe "estructuras Sintácticas".

1957. Newell, Shaw, y Simon crean GPS.

1958. McCarthy introduce el lenguaje "LISP", para procesamiento simbólico de la información.

1959. Minsky y McCarthy crean el laboratorio de IA en el MIT.

1959. Rosenblatt introduce el Perceptron.

1960. EL programa de ajedrez de Samuel gana juegos contra grandes jugadores.

1962. McCarthy se muda a Standford donde funda el laboratorio de IA en 1963.

1963. Se desarrollan los primeros robots industriales.

1964. ARPA da un fondo de investigación de dos millones de dólares al laboratorio de IA del MIT.

1963. Quillian desarrolla las redes semánticas como modelo de representación del conocimiento.

1964. Minsky escribe "Steps toward Artificial Intelligence".

1965. Bobrow desarrolla STUDENT.

1964. Se comienza el desarrollo de BBNLisp en BBN.

1965. Buchanan, Feigenbaum y Lederberg comienzan el proyecto DENDRAL, el primer Sistema Experto.

1965. Iva Sutherland hace demostración del primer monitor en forma de casco para realidad virtual.

1966. Simon predice "por 1985 los ordenadores serán capaces de realizar cualquier trabajo que un hombre pueda hacer".

1967. Dreyfus argumenta en contra de la IA.

1968. Donald Michie funda el laboratorio de IA en Edinburgo.

1968. Minsky publica "Semantic Information Processing".

1969. Minsky y Papert critican el Perceptron.

1970. Colmerauer desarrolla PROLOG quizás el lenguaje de IA más popular actualmente.

1970. Winograd crea SCHRDLU.

1972. Dreyfus publica "What Computers Can't Do".

1973. Se desarrolla el lenguaje SmallTalk en Xerox PARC.

1974. Shank y Abelson desarrollan los guiones, o scripts, base de muchas técnicas actuales de la IA y la Informática en general.

1975. Edward Shortliffe escribe su tesis con MYCIN, uno de los Sistemas Expertos más conocidos.

1974. Se desarrolla el primer robot controlado por ordenador.

1975. Minsky publica "A Framework for Representing Knowledge".

1976. Se establece la red SUMEX-AIM para aplicaciones de la IA en medicina.

1977. La DARPA lanza un programa de financiación para el priocesamiento y comprensión de imágenes.

1978. Greenblatt crea "CONS" el primer ordenador con arquitectura para LISP.

1976. Kurzweil introduce su máquina lectora.

1977. Lenat introduce su "Automated Matematician".

1978. Xerox comienza a desarrolla ordenadores LISP.

1979. Raj Reddy funda el Instituto de Robótica en la Universidad Carnegie Mellon.

1980. Primera conferencia de la AAAI (American Association on Artificial Intelligence) en Stanford, y primera Conferencia de Lisp y programación funcional de la ACM.

1981. Kazuhiro Fuchi anuncia el proycto japonés de quinta generación de ordenadores.

1981. El PSL (Portable Standard Lisp) se puede ejecutar sobre varias plataformas.

1982. Se construyen máquinas LISP por Xerox, LMI y Simbolics, las cuales soportan Programación Orientada a Objetos.

1983. Se sientan las bases del Common Lisp con aspectos comunes de las familias: Lisp machine Lisp, MacLisp, NIL, S-1 Lisp, Spice Lisp y Scheme.

1984. John Hopfield resucita las redes neuronales.

1985. Feigenbaum y McCorduck publican "The Fifth Generation".

1986. Steele publica "Common Lisp the Language".

1984. La comunidad europea comienza el programa ESPRIT.

1985. Gold Hill crea el Golden Commom Lisp.

1986. General Motors y Campbell's Soup dejan de usar Lisp para sus Sistemas Expertos.

1985. Un robot de la Kawasaki mata a un mecánico japonés en un mal funcionamiento.

1986. Se funda el Media Lab en el MIT.

1987. Minsky publica "The Society of Mind".

1988. Teknowledge, una compañía dedicada al desarrollo de sistemas en IA, abandona Lisp y Prolog por el lenguaje C.

1989. El robot jugador de tenis de mesa de Anderson le gana a un ser humano.

1986. La máquina de ajedrez HiTech de CMU compite en un torneo de nivel master.

1987. La policía de Dallas usa un robot para entrar en las casas.

1988. Primera conferencia de la OOPSLA sobre programación orientada a objetos, en la cual se presenta CLOS, Lisp Orientado a Objetos, como lenguaje independiente de la comunidad de Lisp e IA.

1989. IBM desarrolla shells para Lisp, Prolog y Sistemas expertos y entra a la AAAI.

1990. McClelland y Rumelhart's publican "Parallel Distributed Processing" (Redes Neuronales).

1991. Aparecen compañías dedicadas al desarrollo de Redes Neuronales.

1992. Existen alrededor de 1900 Sistemas Expertos en el mundo.

1987. Sistema experto XCON de DEC capaz de configurar ordenadores realizando el trabajo de 300 personas, basándose para esto en 10.000 reglas.

1988. Japón establece su sistema AFIS para la identificación automática de huellas digitales.

1989. El chip del 386 ofrece una velocidad a los PCs comparable a la de las máquinas Lisp.

1988. Minsky y Papert publican una revisión de "Perceptrons".

1989. Se establecen los lenguajes Orientados a Objetos.

1990. La compañía TI anuncia imcroExplorer una máquina Lisp con tecnología Macintosh.

1990. Steele publica la segunda edición de "Common lisp the Language".

1992. Apple Computer introduce el lenguaje Dylan, de la familia Lisp, como su visión del futuro en la programación.

1993. X3J13 crea una propuesta para la Sociedad Americana de Common Lisp.

1994. La versión para tiempo real del lenguaje CLOS, Lisp con Objetos, de Harlequin se utiliza en sistema de intercambio de AT&T.

AREAS DE LA INTELIGENCIA ARTIFICIAL

- Búsqueda de soluciones
- Teoría de JUEGOS
- Sistemas Expertos
- Procesamiento del Lenguaje Natural
- Robótica
- Visión de Maquina
- Redes Neuronales

LENGUAJES DE LA INTELIGENCIA ARTIFICIAL

- IPL II (para el procesamiento de listas), SAIL (derivado del ALGOL planeado para usarlo en sistemas de visión y procesamiento de lenguaje natural), BASIC para maquinas inteligentes, RAIL(lenguaje de robots), AL(para calcular los movimientos de un robot en el espacio), VAL(para programación de robots industriales) y AML(desarrollado por IBM para el control de robots).
- Actuales: LISP con sus variantes: MACLISP, INTERLISP, ZETALISP Y PSL.
- PROLOG. Es mas utilizado en Europa y Japón, de hecho logró que LISP cayera en desuso, aunque en la actualidad se encuentra en plena reactivación.

CAPITULO

2
SOLUCION DE PROBLEMAS LOGICOS

Mediante la creación de agentes se definen esquemas software o hardware - software que tratan de encontrar la solución de un problema de tipo lógico, a partir de definir los elementos del problema, su espacio de búsqueda (secuencia de solución) y su solución.

Para el espacio de búsqueda se definen secuencias de análisis a las cuales se les llama recorridos, que adoptan dos estrategias a saber:

A) Esquema exhaustivo, determinístico o algorítmico.
B) Esquema Heurístico o no determinístico.

TIPOS DE PROBLEMAS

A) Problemas de un estado
B) Problemas de estado múltiple
C) Problemas de contingencia

CONCEPTO DE PROBLEMA

Es un conjunto de información que el agente utiliza para decidir lo que se va a hacer.

ETAPAS

A) Estado inicial
B) Conjunto de acciones que el agente puede hacer
C) Ruta
D) Prueba de meta
E) Costo de ruta
F) Solución

Espacio de estados

PROBLEMAS TIPICOS

A) El agente viajero
B) Misioneros y caníbales
C) Las torres de Hanoi
D) El mico y los plátanos
E) Problemas de juegos :

1. **TIC-TAC-TOE**
2. **8 PUZZLE**
3. **LAS TAJUELAS**
4. **DAMAS**
5. **AJEDREZ**

Problema del Agente Viajero.

Optimización es uno de los temas con mayor debate dentro del área de cómputo.

Descripción del problema del Agente Viajero (PAV).

Indiscutiblemente, el PAV es quizá el problema de optimización combinatoria más popular de todos (Reyes, 1996). De la manera más simple, el Problema del Agente Viajero (PAV) consiste, en que dadas un conjunto de ciudades, un vendedor debe visitar cada una de ellas y regresar a su ciudad de partida, de tal forma que su recorrido sea minimo (figura 1).

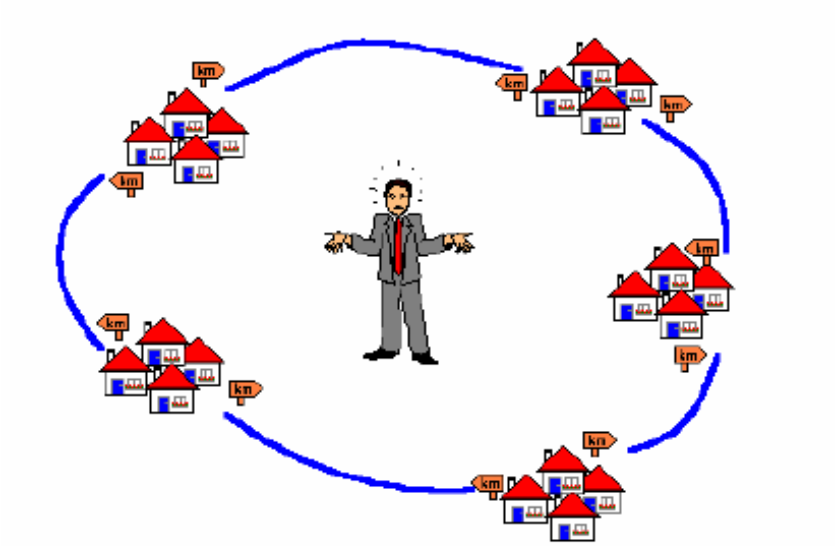
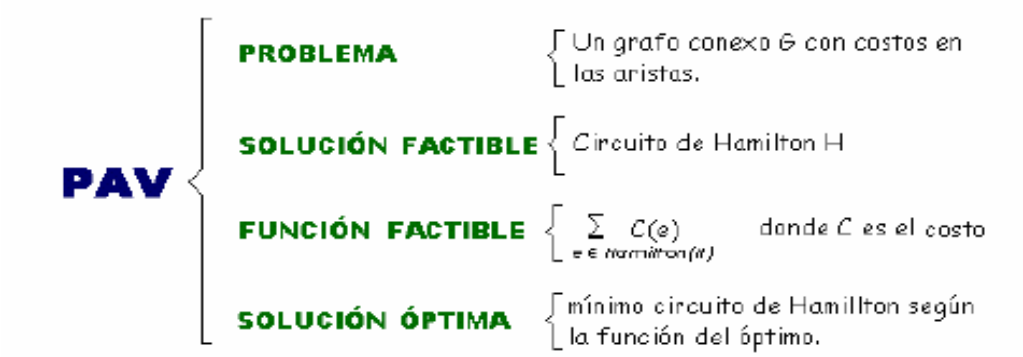


Figura 1.

Definición formal del PAV

Dado un grafo conexo *Kn* con pesos en sus aristas *Cuv*, encontrar el *Circuito de Hamilton* más corto en *Kn*.

Podemos esquematizar el problema del Agente Viajero en la siguiente



El PAV gráfico.

El PAVG (Problema del Agente Viajero Gráfico), emplea un grafo conectado arbitrario *G*. En este tipo de agente está permitido visitar una ciudad en más de una ocasión. Su objetivo es encontrar el camino cerrado en *G* más corto al visitar todas las ciudades requiriendo de la menor distancia posible.

La trayectoria más corta en un Circuito de Hamilton.

Dado un grafo *G* con pesos en sus aristas c_{ij}. Existen dos nodos especiales llamados v_s y v_t que pertenecen a V. La tarea es encontrar el circuito de Hamilton en G de v_s a v_t.

Una forma de solucionar este problema es escogiendo un valor M lo eficientemente grande y asignarle un peso -M a la arista que hay entre v_s y v_t. Entonces calcular el PAV óptimo.

El Circuito de Hamilton y los problemas cíclicos.

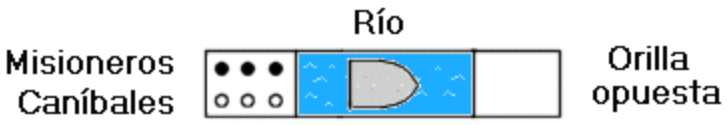
Algunas veces es necesario verificar si un grafo *G = (V,E)* contiene un c*ircuito de Hamilton*. Esta pregunta puede responderse utilizando un PAV simétrico en un grafo completo K_n (con $n = V$).

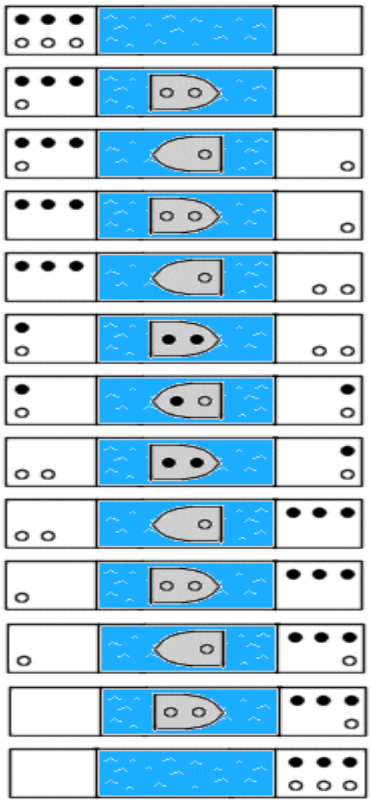
Construimos el grafo *K,* asignándole un peso de 1 a todas las aristas del grafo original y un peso de 2 a las demás. *G* contiene un *circuito de Hamilton* si el circuito más corto en K_n tiene una longitud de *n*. Si la longitud es de *n+1*, entonces *G* no es Hamiltoniano pero contiene una trayectoria Hamiltoniana.

Misioneros y caníbales

Tres misioneros se perdieron explorando una jungla. Separados de sus compañeros, sin alimento y sin radio, sólo sabían que para llegar a su destino debían ir siempre hacia adelante. Los tres misioneros se detuvieron frente a un río que les bloqueaba el paso, preguntándose que podían hacer. De repente, aparecieron tres caníbales llevando un bote, pues también ellos querían cruzar el río. Ya anteriormente se habían encontrado grupos de misioneros y caníbales, y cada uno respetaba a los otros, pero sin confiar en ellos. Los misioneros se aprovechaban de los caníbales cuando les superaban en número, bautizándoles quisieran o no antes de que pudieran escapar.

Los tres caníbales deseaban ayudar a los misioneros a cruzar el río, pero su bote no podía llevar más de dos personas a la vez y no querían que los misioneros les aventajaran en número. ¿Cómo puede resolverse el problema, sin que en ningún momento hayan más misioneros que caníbales en cualquier orilla del río? recuerda que un misionero y un caníbal en una orilla del río más uno o dos misioneros en el bote al mismo lado, significa que tendrás problemas.





Torre de Hanoi

La torre del rompecabezas de Hanoi fue inventada por el matemático francés Edouard Lucas en 1883. Nos dan una torre de ocho discos , empilada inicialmente en talla que disminuye en una de tres clavijas. El objetivo es transferir la torre entera a una de las otras clavijas , al mismo tiempo y nunca moviendo solamente un disco más grande sobre un más pequeño.

Sus tactos de la solución en dos asuntos importantes discutidos más adelante encendido:

- funciones recurrentes y pilas
- relaciones de repetición

Solución recurrente

Llamada dejada las tres clavijas Src (fuente), aux. (auxiliar) y Dst (destinación). Entender y apreciar mejor la solución siguiente que usted debe intentar solucionar el rompecabezas para el número pequeño de los discos por ejemplo, 2.3, y, quizás, 4. No obstante uno soluciona el problema, más pronto o más adelante los discos inferiores tendrán que ser movidos desde Src a Dst. A este punto en tiempo todos los discos restantes tendrán que ser empilados en orden de la talla que

disminuye en aux.. Después de mover el disco inferior desde Src a Dst que estos discos tendrán que ser movidos desde aux. a Dst. Therefore, para un número dado N de discos, el problema aparece ser solucionado si sabemos lograr las tareas siguientes:

1. Mueva los discos de la tapa N-1 desde Src a aux. (con Dst como clavija intermediaria)
2. Mueva los discos inferiores desde Src a Dst
3. Mueva los discos N-1 desde aux. a Dst (que usa Src como clavija intermediaria)

Mueva los discos N-1 desde aux. a Dst (que usa Src como clavija intermediaria) Asuma que hay una función soluciona con para argumentos - número de los discos y de tres clavijas (fuente, intermediario y destinación - en esta orden). Entonces el cuerpo de la función pudo mirar gusto

Solve(N, Src, aux., Dst)
si N es 0 salidas
Solve(N-1, Src, Dst) movimiento
aux. de Src a Dst
Solve(N-1, aux., Src, Dst)

Esto sirve realmente mientras que la definición de la función soluciona. La función es recurrente en que se llama en varias ocasiones con valores que disminuyen de N hasta que se ha resuelto una condición que terminaba (en nuestro caso N=0).

Relaciones de repetición

Deje T_N ser el número mínimo de los movimientos necesitados para solucionar el rompecabezas con los discos de N. De la sección anterior $T_3 = 7$ y $T_4 = 15$. Uno puede convencerse fácilmente ese $T_2 = 3$ y $T_1 = 1$. La solución recurrente arriba implica el mover dos veces de los discos (N-1) a partir de una clavija a otra y el hacer de un movimiento adicional mientras tanto. Entonces sigue eso

$T_N \leq T_{N-1} + 1 + T_{N-1} = 2T_{N-1} + 1$

La desigualdad sugiere que puede ser que sea posible mover discos de N con menos que movimientos $_{de\ 2T}$ N-1 +1. Cuál no es realmente el caso. De hecho, cuando el tiempo viene mover el disco inferior (N-1) los discos habrán sido movidos desde Src a aux. en por lo menos movimientos $_{de\ T}$ N-1. Puesto que estamos intentando utilizar como pocos pasos de progresión como sea posible, podemos asumir que esa porción de la tarea tomó exactamente movimientos $_{de\ T}$ N-1. Lleva apenas un movimiento de mover el disco más grande desde Src Dst. Uno entonces necesita exactamente T_{N-1} más pasos de progresión acabar la tarea. Por lo tanto el número mínimo de movimientos necesitó solucionar el rompecabezas con los iguales T de los discos de $N_{N-1} + 1 + T_{N-1} = 2T_{N-1} + 1$ movimientos.

Es decir

$T_N = 2T_{N-1} + 1$

Así podemos definir la cantidad T_N como

$T_o = 0$

$T_N = 2T_{N-1} + 1$ para $N > 0$

Así podemos computar $T_1 = 2T_0 + 1 = 1$, $T_2 = 2T_1 + 1 = 3$, $T_3 = 2T_2 + 1 = 7$ etcétera secuencialmente.

La expresión antedicha se conoce como una relación de repetición que, pues usted puede ser que haya notado, sea solamente función recurrente.

ARBOLES

DEFINICIONES:

Grafo: Es un conjunto compuesto por puntos, llamados nodos o vértices. Asociado a un conjunto compuesto por líneas llamadas arcos. Cada arco une a un nodo de origen con un nodo extremo.

Arbol: un árbol es grafo siempre que:

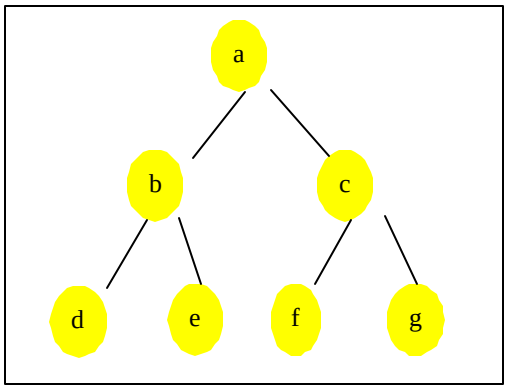
haya un nodo llamado raíz o padre.

Los demás nodos estén agrupados en n conjuntos desunidos x1, x2,...,xn que a su vez sean arboles que tengan conexión con la raíz, que se denominan sub-arboles de x. Los nodos x1,..,xn se denominan hijos de x. Cuando n<=2 el árbol se denomina binario.

ESTRATEGIAS DE BÚSQUEDA

Métodos Exhaustivos.

1. Preorden
2. Posorden
3. De anchura
4. Inorden



Considere el siguiente problema en el cual se pretende determinar las distintas maneras de que un joven puede vestirse.

ESPACIO DE ESTADOS: define las opciones de solución del problema integrado por las prendas de vestir.
BOXERS
CAMISETA
CAMISA
PANTALÓN
CALCETINES
ZAPATOS
CINTURÓN

ACTIVIDADES:
A: colocar BOXERS
B: ponerse CAMISETA
C: poner CAMISA
D: colocar CALCETINES
E: ponerse el PANTALON
F: poner en el pantalón el CINTURON
G: calzar ZAPATOS

PREORDEN: ABDECFG, lo cual indica colocar boxers, ponerse la camiseta, colocar calcetines, poner pantalón, poner camisa, poner el cinturón y calzar zapatos.
Su secuencia indica iniciar en la raíz; avanzar en preorden por el primer sub-arbol de la raiz (es decir, por el lado derecho), por ultimo, avanzar en preorden por el segundo sub-arbol.

POSORDEN: DEBFGCA. Este indica poner camisa, colocar calcetines, ponerse camiseta, poner el cinturón, calzar zapatos, ponerse el pantalón y por ultimo colocar boxers.
Inicia en el hijo izquierdo del primer sub-arbol de la raiz (el izquierdo), se recorre en posorden el primer sub-árbol de la raiz, se recorre en posorden el ultimo sub-arbol de la raíz, por ultimo se examina la raíz.

ANCHURA: ABCDEFG, colocar boxers, ponerse camiseta, ponerse el pantalón, poner camisa, colocar calcetines, poner el cinturón y por ultimo calzar zapatos.
Es decir, el recorrido en el árbol inicia en la raíz, se examinan los nodos nivel a nivel y de izquierda a derecha.

INORDEN: CBDAFEG, poner camisa, poner camiseta, colocar calcetines, colocar boxers, poner el cinturón, ponerse el pantalón y por ultimo calzar zapatos.
Se recorre en forma intermedia el primer sub-arbol de la raiz, se examina la raíz, se recorre en orden intermedio el segundo sub-arbol y por ultimo se examina la raíz.

Métodos Heuristicos.

- mejoran la eficiencia de la búsqueda a través de una función de evaluación que ayuda a ordenar las selecciones de modo que las más prometedoras se exploren primero. Denotando la importancia de la definición de una buena función heurística : “un mayor conocimiento suele reducir el tiempo de búsqueda”.

    - **Búsqueda de ascenso en colina :** es una búsqueda en profundidad con una medición heurística que ordena las alternativas conforme los nodos se extienden. La ordenación de las selecciones se hace conforme con alguna medición heurística de la distancia que queda por recorrer a la

meta. Conforme mejor sea la medición heurística, mejor será el ascenso de colina con relación a la búsqueda en profundidad normal. El ascenso en colina presenta los siguientes problemas :

- El **problema de la falda de colina** ocurre siempre que hay picos secundarios. Se encuentra un punto óptimo, pero se trata de un máximo local, no de un máximo global, y el usuario queda con una falsa impresión de seguridad o de haber terminado.
- El **problema de la meseta** se presenta cuando hay un área casi plana que separa los picos. La operación de mejoramiento local fracasa por completo, ya que para todas las posiciones - excepto un número muy pequeño - todas las pruebas de paso normal dejan intacta la medición de calidad.
- El **problema del pico** es más sutil y, en consecuencia, más frustrante. Un mapa de contorno muestra que cada paso normal le lleva hacia abajo, aun cuando no se encuentra en ninguna especie de máximo local o global. Puede ser de ayuda aumentar el número de direcciones que se utilizan en los pasos de prueba.

◊ forme una cola de un solo elemento consistente en una trayectoria de longitud cero que contenga sólo al nodo raíz ;
◊ hasta que la primera trayectoria de la cola concluya en el nodo meta o se vacíe la cola,
  ◊ elimine la primera trayectoria de la cola ; cree nuevas trayectorias extendiendo el primer paso a todos los vecinos del nodo terminal ;
  ◊ rechace todas las trayectorias nuevas con ciclos ;
  ◊ *ordene las trayectorias nuevas*, si las hay, según *las distancias estimadas* entre sus nodos terminales y la meta ;
  ◊ agregue las nuevas trayectorias, si las hay, al *frente* de la fila.
◊ si el nodo meta se encuentra, mencione que hubo éxito ; de otro modo, notifique el fracaso.

- **Búsqueda primero el mejor :** cuando el movimiento hacia adelante se ve impedido, el ascenso de colina exige un movimiento más desde el nodo abierto de más reciente creación ; en este tipo de búsqueda, el movimiento adicional se realiza a partir del mejor nodo abierto que se tiene hasta ese punto, sin importar donde éste ese nodo en el árbol parcialmente desarrollado.

- **Búsqueda por recorrido simulado :** el bucle más interno de la fusión simulada es bastante similar al del ascenso de la colina cuando se ha

quedado atorado en un máximo local; en vez de optar por la mejor acción, se escoge una al azar. Si mediante la acción se logra mejorar la situación, se ejecuta. De lo contrario, el algoritmo realizará la acción con cierta probabilidad inferior a 1. La probabilidad disminuye exponencialmente con lo "malo" de la acción. Para determinar la probabilidad se utiliza un segundo parámetro, temperatura T. Cuando los valores de T son elevados, hay más probabilidad de que se acepten las acciones "malas". Conforme T tiende a cero, es menos probable su aceptación, hasta que el algoritmo se comporte más o menos como el ascenso de la colina.

- **Métodos de búsqueda óptima :** tratan con situaciones de búsqueda en las que el costo de seguir una trayectoria es de principal importancia. A diferencia de los anteriores métodos, estos métodos no se conforman con una buena solución sino siempre seleccionan la mejor solución.
    - **Búsqueda del museo británico :** este procedimiento para encontrar la trayectoria más corta consiste en encontrar todas las trayectorias posibles y seleccionar la mejor de ellas. Por desgracia, el tamaño de los árboles de búsqueda suele ser grande y cualquier procedimiento para hallar todas las trayectorias posibles se vuelve en extremo fastidioso.

    - **Búsqueda de ramificación y poda :** funciona extendiendo la trayectoria parcial de menor costo hasta que tal trayectoria alcanza la meta. El esquema de ramificación y poda siempre se mantiene al tanto de todas las trayectorias parciales que compiten para su consideración posterior. La más corta de ellas se extiende un nivel, creándose tantas trayectorias parciales nuevas como ramas existan. Enseguida, se consideran estas nuevas trayectorias junto con las anteriores restantes : de nuevo, se extiende la más corta. Este proceso se repite hasta llegar a la meta a través de una trayectoria. Dado que la trayectoria más corta es la que siempre se escoge para su extensión, la trayectoria que primero encuentra la meta es probable que sea la óptima.

◊ forme una cola de un solo elemento consistente en una trayectoria de longitud cero que contenga sólo al nodo raíz ;
◊ hasta que la primera trayectoria de la cola concluya en el nodo meta o se vacíe la cola,
    ◊ elimine la primera trayectoria de la cola ; cree nuevas trayectorias extendiendo el primer paso a todos los vecinos del nodo terminal ;
    ◊ rechace todas las trayectorias nuevas con ciclos ;
    ◊ agregue las trayectorias nuevas restantes, si las hay, a la cola;
    ◊ *ordene la cola completa por longitud de trayectoria* con *las trayectorias de menor costo* al frente.

◊ si el nodo meta se encuentra, mencione que hubo éxito ; de otro modo, notifique el fracaso.

* Este tipo de búsqueda se puede mejorar en gran medida (extendiendo muchos menos nodos) mediante el uso de **subestimaciones** acerca de las distancias restantes, así como de hechos sobre las distancias ya obtenidos. Después de todo, si la conjetura sobre una distancia restante es buena, entonces esa distancia supuesta, al agregarse a la distancia que se conoce definitivamente y que se ha recorrido, deberá ser un buen cálculo de la longitud total de la trayectoria, *e* (longitud total de la trayectoria) :

***e* (longitud total trayectoria) = *d* (ya recorrida) + *u* (distancia restante),**

donde *d* (ya recorrida) es la distancia que ya se recorrió y *u* (distancia restante) es un cálculo de la distancia que falta.

- **Búsqueda A*:** es una búsqueda de ramificación y poda con subestimaciones, en combinación con el principio de programación dinámica. La idea de la programación dinámica es suprimir las trayectorias parciales redundantes que acaban con la eficiencia de la búsqueda bajo el principio de que “el mejor camino a través de un lugar intermedio específico es el mejor camino hacia esté desde el lugar de inicio, seguido por el mejor camino desde éste a la meta. No hay necesidad de buscar cualesquiera otras trayectorias hacia, o desde el punto intermedio”.

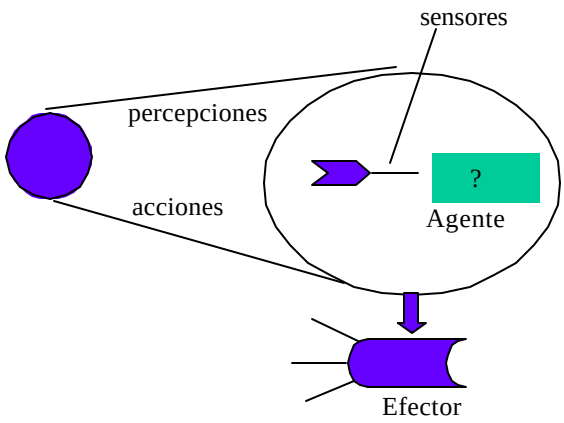
◊ forme una cola de un solo elemento consistente en una trayectoria de longitud cero que contenga sólo al nodo raíz ;
◊ hasta que la primera trayectoria de la cola concluya en el nodo meta o se vacíe la cola,
  ◊ elimine la primera trayectoria de la cola ; cree nuevas trayectorias extendiendo el primer paso a todos los vecinos del nodo terminal ;
  ◊ rechace todas las trayectorias nuevas con ciclos ;
  ◊ si dos o más trayectorias llegan a un mismo nodo, elimine todas aquellas que no alcancen el nodo común con el costo mínimo;
  ◊ *ordene la fila completa según la suma de la longitud de trayectoria* y una *estimación de límite inferior del costo redundante*, con las trayectorias de menor costo al frente..

◊ si el nodo meta se encuentra, mencione que hubo éxito ; de otro modo, notifique el fracaso.

AGENTES INTELIGENTES

Un agente es todo aquello que puede considerarse que percibe su ambiente mediante sensores y que responde y actúa en tal ambiente por medio de efectores.

```
Function agente_para_soluc_de_prob(p) return
accion
        entradas: p, una percepcion
        estatico: s, secuencia de acciones
        estado, descripcion del estado actual del
mundo
        g, meta
        problema, formulacion
  estado    ACTUALIZAR ESTADO(estado,p)
  si s vacio, entonces
      g    formular meta(estado)
      problema    formular(estado,g)
      s   buscar(problema)
  acción    Recomendación(s, estado)
      s    Resto(s, estado)
  responder con acción
```



Los elementos que conforman un agente inteligente son: su arquitectura, la cual delimita su actuación y un programa de computadora el cual le dará al agente la capacidad de interactuar con el medio ambiente para llevar a cabo las acciones que determinaran su desempeño.

En lo que respecta al software utilizado por el agente, invariablemente deberá tener integrada una base de conocimiento que contenga la secuencia de actividades para llevar a cabo las distintas acciones motivadas por cada una de las percepciones recibidas del medio ambiente; esto indica claramente la necesidad de dotar al agente de un nivel de autonomía derivada del uso de sus percepciones para “monitorear” constantemente al medio ambiente y comparar las condiciones imperantes con las condiciones almacenadas en su base de conocimiento y hacer los ajustes correspondientes, es decir se puede hablar de adquirir experiencias en la misma forma en la que lo hace un ser humano.

Considere los siguientes ejemplos:

Tipo de agente	Percepciones	Acciones	Metas	Ambiente
Sistema experto para diagnostico medico	Síntomas y enfermedades	Preguntas, pruebas, tratamientos	Paciente saludable	Paciente, hospital
Robot clasificador de partes	Pixels de intensidad y colores diversos	Recoger partes y clasificarlas poniéndolas en botes	Ponerlas en el bote que les corresponda	Banda transportadora

CAPITULO

3
TEORIA DE JUEGOS

Primeros pasos:

- Charles Babbage en el siglo XIX pensó adaptar su famosa maquina analítica para que jugara ajedrez y mas tarde en desarrollar una maquina que jugara al gato.
- 1950. Claude Shannon y Alan Turing desarrollan los primeros programas para jugar al ajedrez
- 1953. Arthur Samuel desarrolló un programa para jugar a las damas

Elementos

- Sistema de producción:

  En este modulo se definen las reglas del juego, movimientos validos y no validos, así como las posiciones aceptadas para cada una de las piezas o elementos que conforman el juego.
- Mecanismo de búsqueda.

  En este modulo se define una estrategia de búsqueda para poder encontrar en el menor tiempo posible las soluciones disponibles para cada una de las jugadas de cada uno de los contrincantes, generalmente se utiliza la técnica minimax y estrategias de poda de árbol para evitar recorrer todo el árbol, o evitar volver a jugar una jugada ya hecha, además se pueden definir estrategias de búsqueda heurística.
- Función de evaluación. Contiene la definición de los valores asignados a cada jugada a efecto de poder registrar la posición actual de cada uno de los jugadores en cuanto a situaciones ganadoras o perdedoras.
- Mecanismo de aprendizaje.

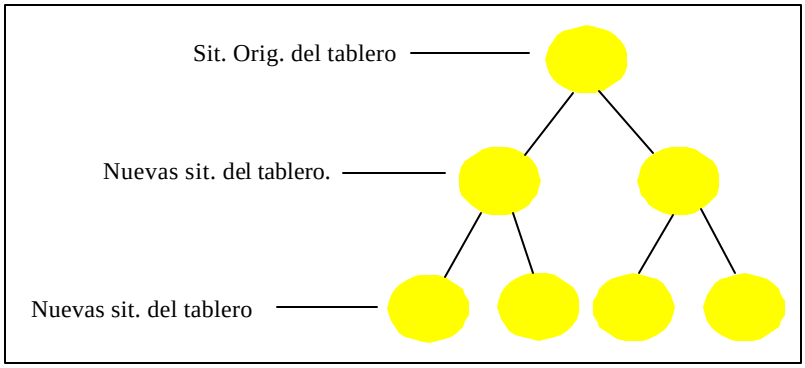
  Permite generar opciones de reprogramación de la computadora con el fin de almacenar en memoria las jugadas ya generadas, su función de evaluación y en caso de que el contrincante realice un retroceso o repita una jugada, decidir en forma anticipada incluso varios movimientos antes todas las posibles jugadas a realizar, y elegir la mejor opción.

Es necesario recalcar la importancia que tiene la heurística en el desarrollo de un juego y en general en la solución de cualquier problema relacionado con la inteligencia artificial, la heurística define una estrategia de solución para un espacio de búsqueda, el utilizar la heurística evita realizar búsquedas exhaustivas

dentro de un árbol de solución y por consiguiente reduce el tiempo de recorrido en el árbol de soluciones.

Métodos algorítmicos.

Permiten representar situaciones de juego en árboles definiendo el recorrido de acuerdo a la situación mas conveniente.

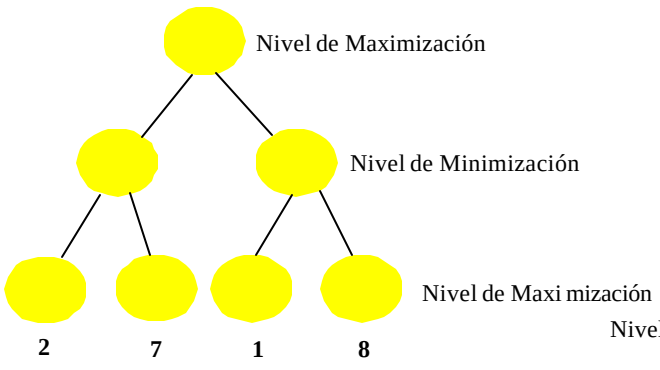


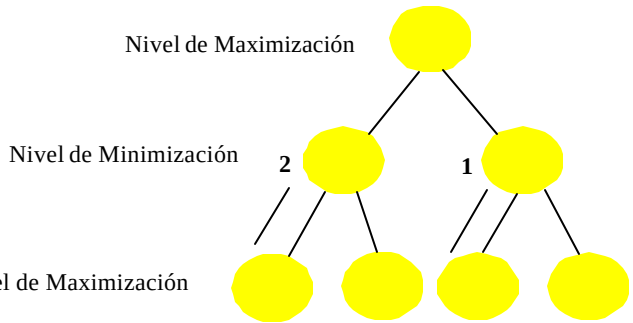
- **El método MiniMax.** Suponga que tenemos un analizador de situaciones que convierte todos acerca de las situaciones de tablero en numero simple de calidad global. Suponga tambien que los números positivos, por convención, indican la ventaja de un jugador y los negativos la desventaja del otro. El grado de ventaja aumenta con el valor absoluto del número.
- El proceso de cálculo de un número que refleje la calidad de tablero se conoce como evaluación estática. El procedimiento que hace el cálculo es un evaluador estático y el número que calcula se conoce como resultado de la evaluación estática.
- El jugador que espera obtener números positivos se conoce como jugador de maximización o maximizador. El otro jugador recibe el nombre de jugador de minimización o minimizador.
- El procedimiento mediante el cual la información acerca de los resultados pasa hacia arriba del árbol de juegos se conoce como procedimiento Minimax, debido a que el resultado en cada nodo es o bien el mínimo o el máximo de los resultados de los nodos que están inmediatamente abajo.

Si el limite de búsqueda se ha alcanzado, calcule el valor estático de la posición actual en relación con el jugador apropiado. De a conocer el resultado.

De otro modo, si el nivel es de minimización, use Minimax en bs hijos de la posición actual. Dé a conocer el resultado.

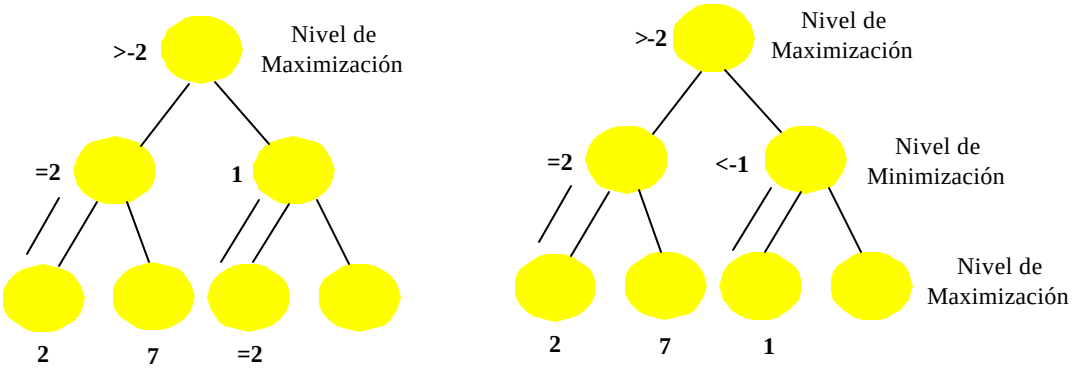
De lo contrario, el nivel es de Maximización. Use Minimax en los hijos de la posición actual. Notifique el máximo de los resultado.



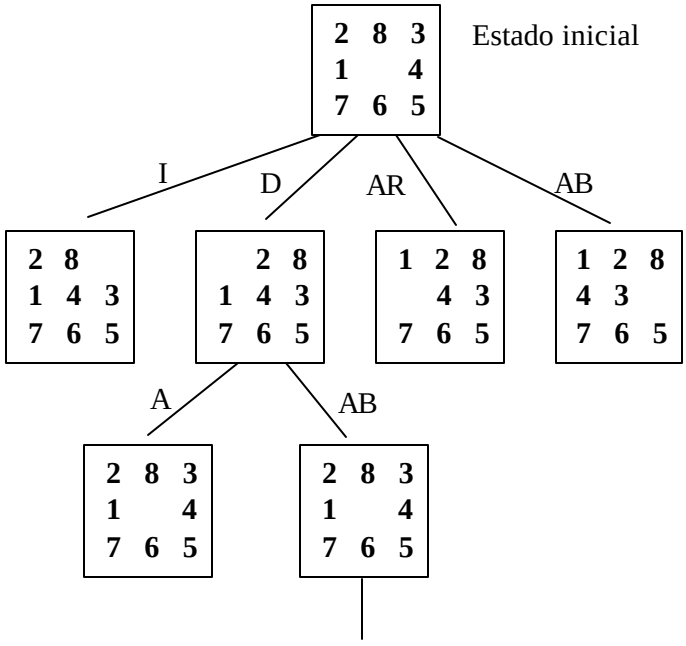


- El método Alfa-Beta. Reduce tanto el número de ramas que deben generarse como de evaluaciones estáticas que deben hacerse, acortando el así el trabajo global que se debe realizar.

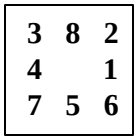
Si tiene una idea que indudablemente es mala, no se tome el tiempo para constatar que tan mala es.



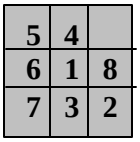
- El problema del 8 Puzzle. Consiste de una matriz de 3x3 con 8 fichas numeradas móviles y un espacio vacío que permite el movimiento en las de las fichas adyacentes.
- El problema planteado por el 8 Puzzle es el de pasar de una configuración inicial dada de las fichas, a una configuración final deseada.
- La database global describe el estado del problema, que es sencillamente la configuración actual de las fichas representada por una matriz numérica. El conjunto de configuraciones posibles constituye el espacio de estados del problema o el espacio de búsqueda. Para este problema el espacio de estados tiene 9! O 362,880 estados o configuraciones distintas de fichas, lo cual conlleva a emplear un mecanismo de poda de árbol con el fin de evitar una búsqueda exhaustiva..



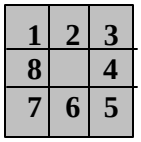
Caso Practico



Estado final



Estado de Partida

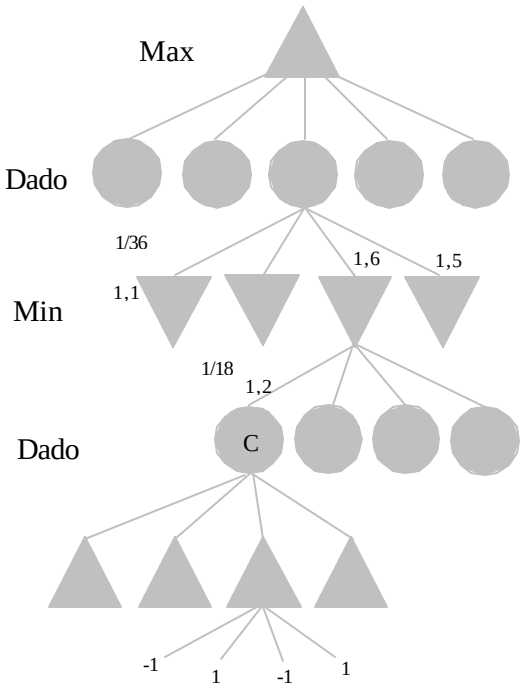


Estado meta

Evaluación de juegos con nodos aleatorios

Backgamón

- **Reglas:** existen fichas de dos diferentes colores y su posición inicial es encontrada, el tablero tiene posiciones de la 0 a la 25,
- el contrincante avanza en sentido de las manecillas del reloj hasta sacar sus fichas por la posición 0 y el contrario avanza en sentido opuesto hasta sacar sus fichas por la posición 25, cuando una ficha cae en una posición en donde hay una ficha del contrincante, se toma esta y tiene que volver a iniciar su desplazamiento desde el principio.
- El avance lo determina el lanzamiento de dos dados, lo que le agrega un toque de aleatoriedad al juego.
- Si bien cada jugador sabe cuales son las jugadas permitidas, no sabe cual será la siguiente jugada del contrincante, por lo que no puede construir un árbol de búsqueda completo y deberá agregar nodos aleatorios, además de los nodos Max y Min. cada una de las ramas que sale de un nodo aleatorio denota el posible resultado del lanzamiento de un dado, a cada uno lo acompaña el resultado y la posibilidad de que ocurra.
- Existen 36 posibles resultados del lanzamiento de dos dados, todas igualmente posibles; pero puesto que 6-5 es lo mismo que 5-6 solo hay 21 posibles lanzamientos diferentes. Los 6 resultados dobles (del 1-1 al 6-6) tienen 1/36 de posibilidad de resultar los otros 15 tienen 1/18 de posibilidad.
- El calculo de los valores esperados es directo. Para los nodos terminales se utiliza una función de utilidad como en los juegos deterministas. Avanzando un paso en el árbol de búsqueda se llega a un nodo aleatorio.



El árbol anterior se puede trazar utilizando los siguientes criterios:

Cada circulo es un nodo aleatorio, ubiquemos al marcado con C, sea di, el posible resultado del lanzamiento de un dado y P(di) la probabilidad de que se obtenga dicho resultado, se calcula la utilidad de la mejor jugada para Min correspondiente a cada lanzamiento de dados y se añaden las utilidades, ponderadas con la probabilidad de obtención de un determinado lanzamiento de dados. Si **S(C, di)** denota el conjunto de posiciones generadas por la aplicación de jugadas permitidas al lanzamiento de dados **P(di)** a la posición de C, se podrá calcular el valor esperado Max de C, mediante la formula: esperadoMax= $\sum$P(di) Max $s \in$ S(C,di) (utilidad(s))

Funciones Heurísticas

Conceptos:

- Newell, Shaw y Simon declararon al respecto:"cuando un proceso afirma poder resolver un problema determinado pero no ofrece ninguna garantía de ello, se dice que es la heurística del problema"
- También se dice que son reglas practicas para generar resultados sin tener que utilizar búsquedas exhaustivas.

Heurística

- H1= la cantidad de placas que están en lugar incorrecto. Como ninguna lo está H1=8.
- H2= la suma de las distancias que separa a las placas de sus posiciones meta. Puesto que las placas no se desplazan a través de diagonales, la distancia que se considera, es la suma de las distancias horizontales y verticales. (a esto se le llama distancia de Manhattan). Por lo que las 8 placas al momento de partida tienen una distancia de Manhattan de: H2=2+3+2+1+2+2+1+2=15.
- Por lo que: debe considerar lo siguiente:
- una placa puede pasar del cuadro A al B si A está junto a B y B está vació
- condiciones:
- Se puede mover una placa del A al cuadro B si A está junto a B
- Se puede mover una placa del A al cuadro B si B está vacío
- Se puede mover una placa del A al cuadro B

EL JUEGO DE LA CAZA

heurística del juego:

- Se desarrolla en un tablero que es una matriz 8x8 con dos jugadores, un jugador tiene una ficha de un color y el otro tiene cuatro.
- Cada jugador ocupa la primera línea del tablero en una posición encontrada y avanzan por turno en diagonal.
- El jugador con cuatro fichas, solo puede avanzar hacia adelante, mientras el que tiene una ficha puede hacerlo hacia adelante y hacia atrás.
- Si el jugador con una ficha logra llegar hasta la primera línea del contrario, gana el juego.
- Si el jugador con cuatro fichas logra acorralar al contrario gana el juego.

En el apéndice A, se agrega el código fuente en lenguaje Prolog de este juego

PROBLEMAS DE JUEGOS

A) TIC-TAC-TOE : Formar una línea horizontal , vertical u diagonal de tal manera que queden tres figuras iguales en alguna de las formas antes ya mencionadas.

B) 8 PUZZLE : El puzzle 8 consiste en una pequeña caja con la forma de una matriz de 3x3 con 8 piezas enumeradas de 1 a 8 y un espacio libre que permite que las piezas se desplacen cambiando su posición. Dada una configuración inicial se desea obtener una secuencia de movimientos que lleve hasta una configuración final.

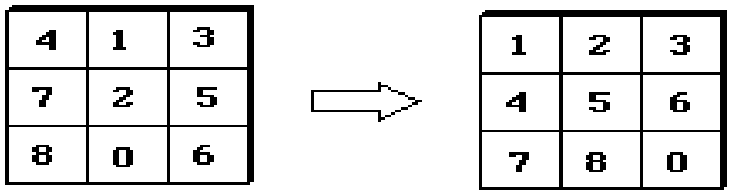


Fig.3.2.- Estados inicial y final del juego de los 8´s

Solución :

- Base de datos : Matriz de 3x3

- Conjunto de reglas :

R1 : Mueva el 0 hacia arriba

R2 : Mueva el 0 hacia la izquierda

R3 : Mueva el 0 hacia abajo

R4 : Mueva el 0 hacia la derecha

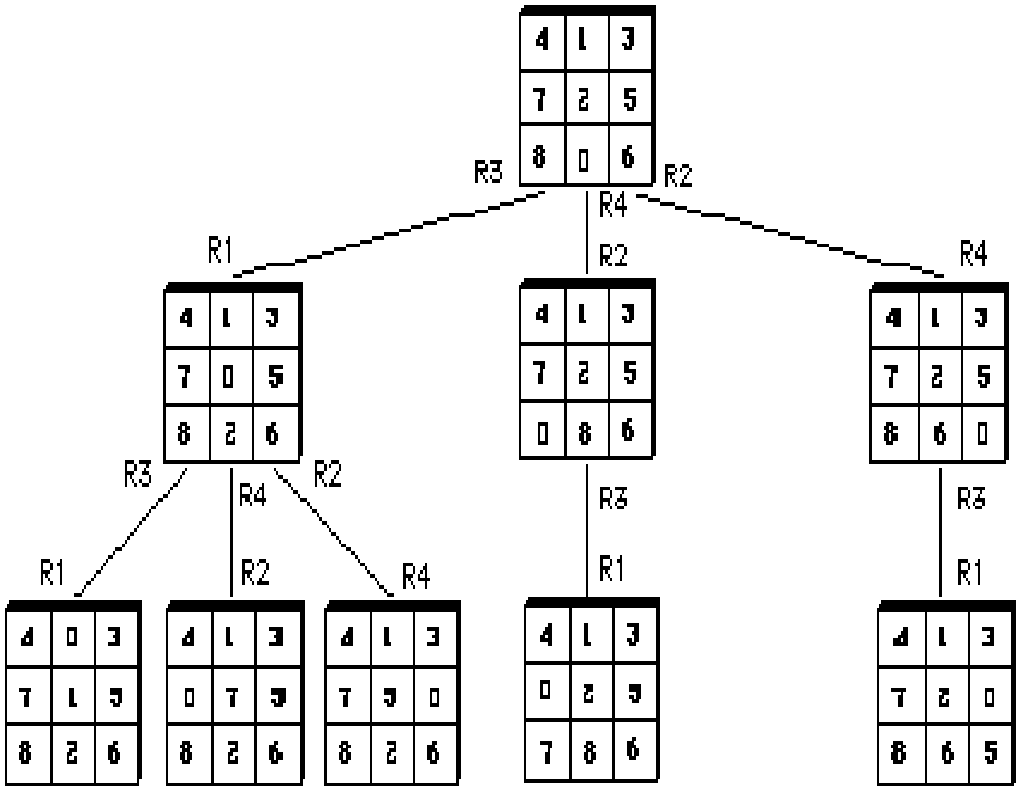


Fig.3.3.- Parte del grafo de estado del juego de los 8´s

C) DAMAS : Juegas con las negras.
Haz clic a la ficha que deseas mover e inm1ediatamente después haz clic a la casilla a la que deseas moverla.
Puedes cambiar de opinión respecto a la ficha que moverás.
Al llegar al final del tablero las fichas se coronan y se pueden mover de **uno en uno** hacia atrás.

C) AJEDREZ : **REGLAS DE JUEGO**

Artículo 1: Naturaleza y objetivos de la partida de ajedrez

1.1. La partida de ajedrez se juega entre dos adversarios que mueven alternativamente sus propias piezas sobre un tablero cuadrado, llamado "tablero de ajedrez". El jugador con las piezas blancas comienza la partida. Se dice que un jugador "está en juego" cuando se ha completado la jugada de su adversario.

1.2. El objetivo de cada jugador es situar al rey de su adversario "bajo ataque", de tal forma que el adversario no disponga de ninguna "jugada" (movimiento de pieza) legal que evite la "captura" del rey en la siguiente jugada. El jugador que consigue esto ha dado " mate" a su adversario y ha ganado la partida. El adversario, que ha recibido el mate, pierde la partida.

1.3. Si la posición es tal que ninguno de los jugadores puede dar mate, la partida es tablas.

Artículo 2: La posición inicial de las piezas sobre el tablero

2.1. El tablero de ajedrez es un cuadrado dividido en 64 casillas cuadradas del mismo tamaño, con distribución 8 x 8, alternativamente claras (las casillas "blancas") y oscuras (las casillas "negras").

El tablero se coloca entre los jugadores de tal forma que la casilla de la esquina derecha más cercana a cada jugador sea blanca.

2.2. Al comienzo de la partida, un jugador dispone de 16 piezas de color claro (las piezas "blancas"); el otro tiene 16 piezas de color oscuro (las piezas "negras").

Estas piezas son las siguientes:

Un rey blanco, usualmente indicado por el símbolo: 

Una reina blanca, usualmente indicado por el símbolo:

Dos torres blancas, indicados usualmente por el símbolo: 

Dos alfiles blancos, usualmente indicados por el símbolo: 

Dos caballos blancos, usualmente indicados por el símbolo:

Ocho peones blancos, usualmente indicados por el símbolo:

Un rey negro, usualmente indicado por el símbolo: 

Una reina negra, usualmente indicada por el símbolo: 

Dos torres negras, usualmente indicadas por el símbolo: 

Dos alfiles negros, usualmente indicados por el símbolo: 

Dos caballos negros, usualmente indicados por el símbolo: 

Ocho peones negros, usualmente indicados por el símbolo: 

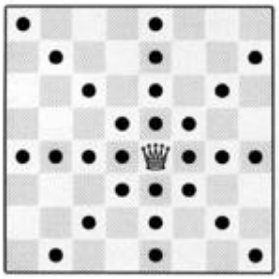
2.3. La posición inicial de las piezas sobre el tablero es la siguiente:

2.4. Las ocho hileras verticales de casillas se denominan ''columnas". Las ocho hileras horizontales de casillas se denominan ''filas ". Una sucesión de casillas del mismo color en línea recta, tocándose por sus vértices, se denomina "diagonal".

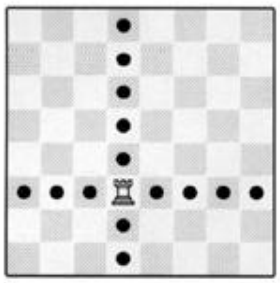
Artículo 3: El movimiento de las piezas

3.1. Ninguna pieza puede ser movida a una casilla ocupada por una pieza del mismo color. Si una pieza se mueve a una casilla ocupada por una pieza de su adversario, ésta es capturada y retirada del tablero como parte del mismo movimiento. Se dice que una pieza ataca una casilla si puede efectuar una captura en la misma conforme a los Artículos 3.2 a 3.6 (excepto en el Artículo 3.5.(a).(ii).(2).[a] , en que se aplicarán los Artículos 3.2 a 3.5).

3.2.(a) La dama se mueve a cualquier casilla a lo largo de la fila, columna o diagonal en las que se encuentra.

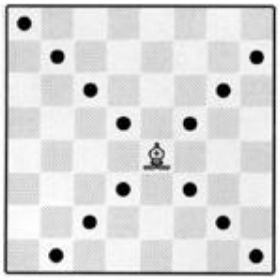


(b) La torre se mueve a cualquier casilla a lo largo de la fila o columna en las que se encuentra



.

(c) El alfil se mueve a cualquier casilla a lo largo de una de las diagonales sobre las que se encuentra.



Al realizar estos movimientos, la dama, la torre o el alfil no pueden pasar sobre ninguna otra pieza.

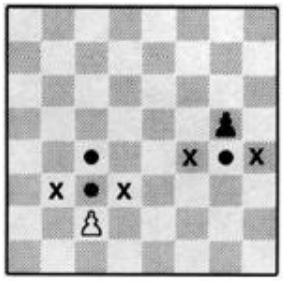
3.3. El caballo se mueve a una de las casillas más próximas a la que se encuentra, sin ser de la misma fila, columna o diagonal. No pasa directamente sobre ninguna casilla intermedia.

3.4.

(a) El peón se mueve hacia adelante a la casilla inmediatamente delante suyo en la misma columna, siempre que dicha casilla esté desocupada; o

(b) en su primer movimiento, el peón puede avanzar dos casillas a lo largo de la misma columna, siempre que ambas casillas estén desocupadas; o

(c) el peón se mueve a una casilla ocupada por una pieza del adversario que esté en diagonal delante suyo, sobre una columna adyacente, capturando dicha pieza.

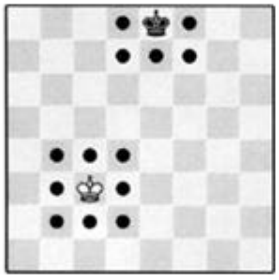


(d) Un peón que ataca una casilla atravesada por un peón del adversario que ha avanzado dos casillas en un movimiento desde su casilla original, puede capturarlo como si sólo hubiera avanzado una casilla. Esta captura sólo puede efectuarse en el movimiento inmediatamente siguiente al citado avance y se denomina captura "al paso".

(e) Cuando un peón alcanza la fila más alejada desde su posición inicial debe ser cambiado, como parte del mismo movimiento, por una dama, torre, alfil o caballo del mismo color. La elección del jugador no está limitada a piezas que hayan sido capturadas anteriormente. Este cambio de un peón por otra pieza se denomina "promoción", siendo inmediato el efecto de la nueva pieza.

3.5. (a) El rey puede moverse de dos formas diferentes:

(i) desplazándolo a cualquier casilla adyacente que no esté atacada por una o más piezas del adversario,

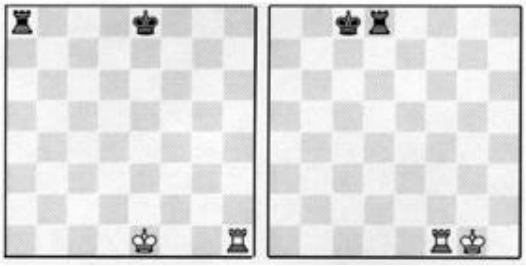


o bien

(ii) "enrocando". El enroque es un movimiento del rey y de una de las torres del mismo color y que esté en la misma fila, que cuenta como una simple jugada del rey y que se realiza como sigue: el rey es trasladado dos casillas desde su casilla original hacia la torre y luego dicha torre es trasladada sobre el rey, a la casilla que éste acaba de cruzar.

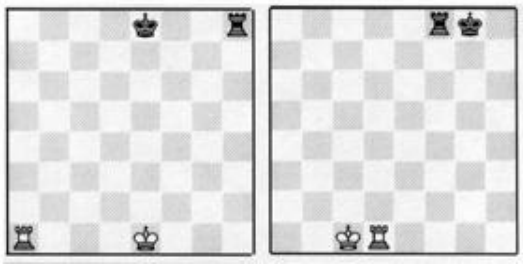
Antes del enroque en el flanco del Rey

Después del enroque en el flanco del Rey



Antes del enroque en el flanco de Dama

Después del enroque en el flanco de Dama



(1) El enroque es ilegal:

[a] si el rey ya ha sido movido, o

[b] con una torre que ya ha sido movida.

(2) El enroque está temporalmente impedido:

[a] si la casilla en la que se encuentra el rey, o la que debe cruzar, o la que finalmente va a ocupar, está atacada por una o más piezas del adversario,

[b] si hay alguna pieza entre el rey y la torre con la que se va a efectuar el enroque.

(b) Se dice que el rey está "en jaque" si se encuentra bajo ataque por una o más piezas del adversario, incluso aunque dichas piezas no pudieran ser movidas.

No es obligatorio declarar un jaque.

3.6. Un jugador no puede hacer una jugada que ponga o deje a su propio rey en jaque.

Artículo 4: La acción de mover las piezas

4.1. Cada jugada debe efectuarse con una sola mano.

4.2. El jugador que está en juego puede ajustar una o más piezas en sus casillas, siempre que previamente exprese su intención de hacerlo (por ejemplo, diciendo "compongo").

4.3. Exceptuando lo previsto en el Artículo 4.2, si el jugador que está en juego toca deliberadamente sobre el tablero:

(a) una o más piezas del mismo color, debe mover o capturar la primera pieza tocada que se pueda mover o capturar; o

(b) una pieza de cada color, debe capturar la pieza del adversario con la suya o, si ello es ilegal, mover o capturar la primera pieza tocada que se pueda mover o capturar. Si resulta imposible establecer qué pieza se tocó en primer lugar, será la pieza propia la que se considere como pieza tocada.

4.4.
(a) Si un jugador toca deliberadamente su rey y torre, debe enrocar por ese lado si fuera legal.

(b) Si un jugador toca deliberadamente una torre y luego su rey, no podrá enrocar con dicha torre en esa jugada y se procederá según lo establecido en el Artículo 4.3.

(c) Si un jugador, con la intención de enrocar, toca el rey o rey y torre a la vez, siendo ilegal el enroque por ese lado, el jugador podrá elegir entre enrocar por el otro lado, supuesto que sea legal, o mover su rey. Si el rey no tiene ningún movimiento legal, el jugador es libre de realizar cualquier jugada legal.

4.5. Si ninguna de las piezas tocadas puede ser movida o capturada, el jugador puede realizar cualquier jugada legal.

4.6. Un jugador no puede reclamar que su adversario ha violado el Artículo 4.3 o el 4.4 después de que él mismo haya tocado deliberadamente una pieza.

4.7. Cuando se ha soltado deliberadamente una pieza sobre una casilla, como jugada legal o parte de una jugada legal, ya no puede ser movida a otra casilla. El movimiento se considera realizado cuando se han cumplido todos los requisitos pertinentes del Artículo 3.

Artículo 5: La partida completada

5.1.

a) La partida es ganada por el jugador que ha dado mate al rey de su adversario con una jugada legal. Esto finaliza inmediatamente la partida.

b) La partida es ganada por el jugador cuyo adversario declara que abandona. Esto finaliza inmediatamente la partida.

5.2. La partida es tablas cuando el jugador que está en juego no puede hacer ninguna jugada legal y su rey no está en jaque. Se dice entonces que el rey está "ahogado". Esto finaliza inmediatamente la partida.

5.3. La partida es tablas por acuerdo entre los dos jugadores durante el desarrollo de la misma. Esto finaliza inmediatamente la partida. (Ver Artículo 9.1).

5.4. La partida puede ser tablas si se va a dar o ya se ha dado una posición idéntica tres veces sobre el tablero. (Ver Artículo 9.2).

5.5. La partida puede ser tablas si se han hecho los últimos 50 movimientos consecutivos de cada jugador sin que haya habido ningún movimiento de peón ni captura de pieza. (Ver Artículo 9.3).

Artículo 6: El reloj de ajedrez

6.1. Se entiende por "reloj de ajedrez" un dispositivo con dos indicadores de tiempo, conectados entre sí de tal forma que sólo uno de ellos pueda funcionar en cada momento.

En las presentes Leyes, el término "reloj" hace referencia a uno de los dos citados indicadores de tiempo.

Se entiende por "caída de bandera" la conclusión del tiempo asignado a un jugador.

6.2. Al utilizar un reloj de ajedrez, cada jugador debe realizar un cierto número de jugadas, o todas, en un período de tiempo prefijado y/o puede recibir un determinado tiempo adicional por cada jugada realizada. Todo ello debe ser especificado con antelación. El tiempo no consumido por un jugador durante un

período se añade a su tiempo disponible para el siguiente período, excepto en la modalidad de "tiempo añadido".

En la modalidad de tiempo añadido, ambos jugadores reciben un determinado "tiempo principal de reflexión". Además, por cada jugada reciben un "tiempo extra fijo". El descuento del tiempo principal comienza sólo cuando el tiempo extra se ha agotado. El tiempo principal de reflexión no cambia siempre que el jugador detenga su reloj antes de que se agote el tiempo extra, independientemente de la proporción de tiempo extra utilizado.

6.3. Cada reloj dispone de una "bandera". Inmediatamente después de una caída de bandera, deben comprobarse los requisitos del Artículo 6.2 (primera frase).

6.4. El Árbitro decide la ubicación del reloj de ajedrez.

6.5. A la hora especificada para el comienzo de la partida, se pone en marcha el reloj del jugador que tiene las piezas blancas.

6.6. Perderá la partida el jugador que se presente ante el tablero con más de una hora de retraso sobre la hora programada para el comienzo de la sesión (salvo que las bases de la competición o el Árbitro decidan otra cosa).

6.7.
(a) Durante la partida, cada jugador, una vez realizada su jugada sobre el tablero, detendrá su reloj y pondrá en marcha el de su adversario. A un jugador siempre se le debe permitir detener su reloj. Hasta que lo haya hecho así, no se considera que su jugada esté completada, salvo que la jugada realizada finalice la partida (ver Artículos 5.1, 5.2 y 5.3).

El tiempo transcurrido entre realizar la jugada sobre el tablero y detener su reloj, poniendo en marcha el del adversario, se considera como parte del tiempo asignado al jugador.

(b) Un jugador debe detener su reloj con la misma mano con la que realizó su jugada. Está prohibido mantener el dedo sobre el pulsador o rondando por encima del mismo.

(c) Los jugadores deben manejar el reloj de ajedrez correctamente. Está prohibido golpearlo violentamente, cogerlo o tirarlo. Un manejo incorrecto del reloj será penalizado de acuerdo con el Artículo 13.4.

6.8. Se considera que una bandera ha caído cuando el Árbitro lo observa o cuando uno de los dos jugadores ha efectuado una reclamación válida en ese sentido.

6.9. Excepto los casos donde se apliquen los Artículos 5.1, 5.2 ó 5.3, si un jugador no completa el número prescrito de jugadas en el tiempo asignado, pierde la partida. Sin embargo, la partida es tablas si la posición es tal que el adversario no puede dar mate al jugador mediante cualquier posible combinación de jugadas legales, incluso jugando de la forma más torpe.

6.10. Toda indicación proporcionada por los relojes se considera concluyente en ausencia de algún defecto evidente. Un reloj de ajedrez con un defecto evidente debe ser reemplazado. El Árbitro hará uso de su mejor criterio al determinar los tiempos que deben aparecer en el reloj de ajedrez reemplazado.

6.11. Si han caído ambas banderas y es imposible establecer cuál de ellas lo hizo en primer lugar, continuará la partida.

6.12.
(a) Si la partida debe ser interrumpida, el Árbitro detendrá los relojes.

(b) Un jugador puede detener los relojes para solicitar la asistencia del Árbitro.

(c) El Árbitro decidirá cuándo se reanudará la partida.

6.13. Si se produce una irregularidad y/o las piezas deben ser reubicadas a una posición anterior, el Árbitro hará uso de su mejor criterio para determinar los tiempos que deben aparecer en los relojes.

6.14. En la sala de juego se permite el uso de pantallas, monitores o tableros de demostración que muestren la posición actual sobre el tablero, los movimientos y el número de jugadas realizadas, así como relojes que muestren incluso el número de jugadas realizadas. Sin embargo, el jugador no puede realizar una reclamación basándose en lo mostrado en este tipo de dispositivos.

Artículo 7: Posiciones ilegales

7.1.
(a) Si en el curso de una partida se comprueba que la posición inicial de las piezas era incorrecta, la partida será anulada y se jugará una nueva.

(b) Si en el curso de una partida se comprueba que el único error ha sido que el tablero no se colocó de acuerdo con el Articulo 2.1, la partida continuará pero la posición deberá transferirse a un tablero colocado correctamente.

7.2. Si una partida ha comenzado con los colores invertidos, la misma continuará salvo que el Árbitro decida otra cosa.

7.3. Si un jugador desplaza una o más piezas, restablecerá la posición correcta en su propio tiempo. Si fuera necesario, el adversario tiene derecho a volver a poner en marcha el reloj del jugador sin realizar ninguna jugada, a fin de asegurarse que éste restablece la posición correcta en su propio tiempo.

7.4. Si en el curso de una partida se comprueba que se ha realizado una jugada ilegal, o que algunas piezas se han desplazado de sus casillas, se restablecerá la posición previa a producirse la irregularidad. Si no puede identificarse la posición inmediata anterior a producirse la irregularidad, la partida continuará a partir de la última posición identificable previa a la irregularidad. Los relojes se ajustarán de acuerdo con el Artículo 6.13 y, en caso de haberse producido una jugada ilegal, a la jugada que la reemplace se le aplica el Artículo 4.3. Después continuará la partida.

Artículo 8: La anotación de las jugadas

8.1. En el transcurso de la partida cada jugador está obligado a anotar sus propias jugadas y las de su adversario, jugada tras jugada, de forma tan clara y legible como sea posible, en anotación algebraica (ver Apéndice E), en la planilla prescrita para la competición.

Si así lo desea, un jugador puede replicar a una jugada de su adversario antes de anotarla. Debe anotar su jugada previa antes de realizar otra. La oferta de tablas debe ser inmediatamente anotada en la planilla por ambos jugadores. (Ver Apéndice E.12).

Si, por razones físicas o religiosas, un jugador está imposibilitado para anotar la partida, al comienzo de la misma se le deducirá de su tiempo asignado lo que el Árbitro decida.

8.2. La planilla estará en todo momento a la vista del Árbitro.

8.3. Las planillas son propiedad de los organizadores del torneo.

8.4. Si un jugador dispone de menos de cinco minutos en su reloj y no tiene un tiempo adicional de 30 segundos o más añadidos con cada jugada, no está obligado a cumplir los requisitos del Artículo 8.1. Inmediatamente después de que haya caído una bandera, el jugador debe actualizar completamente su planilla.

8.5.
(a) Si ningún jugador está obligado a anotar la partida según el Artículo 8.4, el Árbitro o un asistente procurará estar presente y anotar el resto de las jugadas. En tal caso, inmediatamente después de que haya caído una bandera, el Árbitro parará los relojes. Después ambos jugadores actualizarán sus planillas, utilizando la planilla del Árbitro o la del adversario.

(b) Si sólo un jugador no está obligado a anotar la partida según el Artículo 8.4, debe actualizar completamente su planilla en cuanto haya caído una bandera. Supuesto que el jugador esté en juego, puede utilizar la planilla de su adversario. El jugador no está autorizado a realizar una jugada hasta que haya completado su planilla y devuelto la de su adversario.

(c) Si no hay ninguna planilla completa disponible, los jugadores deben reconstruir la partida sobre un segundo tablero bajo el control del Árbitro o un asistente, quien previamente anotará la posición actual de la partida antes de llevar a cabo la reconstrucción.

8.6. Si no se pueden actualizar las planillas de forma que muestren que un jugador ha sobrepasado el tiempo asignado, la siguiente jugada realizada será considerada como la primera del siguiente periodo de tiempo, salvo que sea evidente que se han realizado más jugadas.

Artículo 9: La partida tablas

9.1. Un jugador puede proponer tablas después de realizar una jugada sobre el tablero. Debe hacerlo antes de parar su reloj y poner en marcha el de su adversario. Una oferta en cualquier otro momento de la partida será válida, pero se tendrá en consideración el Artículo 12.5. La oferta no puede ir ligada a ninguna condición. En ambos casos, no se puede retirar la oferta y mantiene su validez hasta que el adversario la acepte, la rechace verbalmente o bien realizando una jugada, o la partida concluya de alguna otra forma.

La oferta de tablas será anotada por cada jugador en su planilla con el símbolo (=).

9.2. La partida es tablas, bajo una correcta reclamación del jugador que está en juego, cuando la misma posición, al menos por tercera vez (no necesariamente por repetición de jugadas):

> (a) va a producirse, si el jugador primero anota su jugada en su planilla y declara al Árbitro su intención de realizarla; o
>
> (b) acaba de producirse.

Se consideran que las posiciones como las de (a) y (b) son la misma si es el mismo jugador quien está en juego, las piezas del mismo tipo y color ocupan las mismas casillas, y los movimientos posibles de todas las piezas de ambos jugadores son los mismos.

Las posiciones no se consideran la misma si un peón pudiera haber sido capturado al paso o si ha cambiado el derecho a enrocar, inmediatamente o en el futuro.

9.3. La partida es tablas, bajo una correcta reclamación del jugador que está en juego, si:

(a) escribe en su planilla y declara al Árbitro su intención de realizar una jugada que dará lugar a que, en los últimos 50 movimientos consecutivos por cada jugador, no se haya movido ningún peón ni se haya capturado alguna pieza; o

(b) se hayan producido los últimas 50 movimientos consecutivos de cada jugador sin moverse algún peón y sin capturar pieza alguna.

9.4. Si un jugador realiza una jugada sin haber reclamado tablas, pierde el derecho a reclamarlas en esa jugada, en base a los Artículos 9.2 y 9.3.

9.5. Si un jugador reclama tablas en base a los Artículos 9.2 ó 9.3, parará inmediatamente ambos relojes. No se le permite retirar su reclamación.

(a) Si se comprueba que la reclamación es correcta, la partida es tablas de forma inmediata.

(b) Si se comprueba que la reclamación es incorrecta, el Árbitro deducirá la mitad del tiempo restante del reclamante, hasta un máximo de tres minutos, y añadirá tres minutos al tiempo restante de su adversario. Luego la partida continuará y debe hacerse la jugada anunciada.

9.6. La partida es tablas cuando se llega a una posición a partir de la cual no puede producirse un mate mediante ninguna posible combinación de jugadas legales, incluso jugando de la forma más torpe. Esto finaliza inmediata mente la partida.

Artículo 10: Final a caída de bandera

10.1. Un "final a caída de bandera" es la última fase de una partida cuando todas las jugadas restantes deben hacerse en un tiempo limitado.

10.2. Si al jugador le quedan menos de dos minutos en su reloj, puede reclamar tablas antes de que caiga su bandera. Parará los relojes y requerirá la presencia del Árbitro.

(a) Si el Árbitro comprueba que el adversario no está haciendo ningún esfuerzo para ganar la partida por procedimientos normales, o que no es posible ganar por procedimientos normales, declarará la partida tablas. En otro caso aplazará su decisión.

(b) Si el Árbitro aplaza su decisión, puede otorgar al adversario dos minutos de tiempo extra de reflexión y la partida continuará en presencia del Árbitro.

(c) Habiendo aplazado su decisión, el Árbitro puede declarar posteriormente la partida tablas, incluso después de que haya caído una bandera.

10.3. Las jugadas ilegales no pierden necesariamente una partida. Una vez aplicado el Artículo 7.4, por la primera jugada ilegal de un jugador el Árbitro dará dos minutos de tiempo extra a su adversario; por una segunda jugada ilegal del mismo jugador, el Árbitro concederá otros dos minutos de tiempo extra al adversario; por una tercera jugada ilegal del mismo jugador, el Árbitro declarará la partida perdida por este jugador.

10.4. La partida es tablas si ambas banderas han caído y es imposible establecer cuál cayó en primer lugar.

Artículo 11: Puntuación

11.1. Un jugador que gana su partida recibe un punto (1), un jugador que pierde su partida recibe cero puntos (0) y un jugador que entabla su partida recibe medio punto (½).

Artículo 12: La conducta de los jugadores

12.1. Se espera de los jugadores un alto nivel de comportamiento correcto y deportivo.

12.2. Durante la partida está prohibido que los jugadores hagan uso de cualquier tipo de notas, fuentes de información, consejos, o que analicen en otro tablero.

La planilla sólo se utilizará para anotar las jugadas, el tiempo indicado en los relojes, la oferta de tablas y cuestiones relativas con una reclamación.

12.3. No está permitido analizar en la sala de juego en el transcurso de las partidas, ni a jugadores ni a espectadores. Los jugadores que hayan finalizado sus partidas se considerarán espectadores.

12.4. No está permitido a los jugadores abandonar el "local de juego" sin permiso del Árbitro. Se entiende por local de juego el recinto constituido por la propia sala de juego, salas de descanso, sala de refrigerio, área reservada a fumadores y cualesquiera otras áreas designadas por el Árbitro.

Al jugador que está en juego no le está permitido abandonar la sala de juego sin permiso del Árbitro.

12.5. Está prohibido distraer o molestar al adversario de alguna manera, cualquiera que sea ésta. Esto incluye la persistente oferta de tablas.

12.6. Las infracciones a las reglas establecidas en los Artículos 12.2 a 12.5 darán lugar a sanciones conforme al Artículo 13.4.

12.7. Pierde la partida el jugador que de forma persistente rehusa cumplir con las Leyes del Ajedrez. El Árbitro decidirá la puntuación del adversario.

12.8. Si ambos jugadores son culpables de infringir el Artículo 12.7, la partida se declarará perdida para ambos.

Artículo 13: El papel del Árbitro (ver el Prólogo)

13.1. El Árbitro se cuidará de que se cumplan estrictamente las Leyes del Ajedrez.

13.2. El Árbitro actuará en el mejor provecho de la competición. Se asegurará de que se mantengan unas buenas condiciones de juego y de que los jugadores no sean molestados. Supervisará el desarrollo de la competición.

13.3. El Árbitro observará las partidas, especialmente cuando los jugadores estén apurados de tiempo, exigirá el cumplimiento de las decisiones que haya adoptado y, cuando corresponda, impondrá sanciones a los jugadores.

13.4. Las sanciones que puede imponer el Árbitro incluyen:

(a) una advertencia,

(b) añadir tiempo en el reloj del adversario,

(c) restar tiempo en el reloj del infractor,

(d) declarar la partida perdida,

(e) la expulsión de la competición.

13.5. El Árbitro puede conceder tiempo adicional a uno o ambos jugadores en caso de desorden ajeno a la partida.

13.6. El Árbitro no puede intervenir en una partida para indicar el número de jugadas realizadas, excepto en aplicación del Artículo 8.5, cuando al menos uno de los jugadores ha empleado todo su tiempo. El Árbitro se abstendrá de informar a un jugador de que su adversario ha hecho una jugada, o de que ha olvidado pulsar su reloj.

13.7. Los espectadores y jugadores de otras partidas no pueden hablar o interferir de cualquier otro modo en una partida. Si fuera necesario, el Árbitro puede expulsar del local de juego al/a los infractores.

Artículo 14: FIDE

14.1. Las federaciones afiliadas pueden solicitar a la FIDE que adopte una decisión oficial en problemas relacionados con las Leyes del Ajedrez.

APÉNDICES

A. Partidas aplazadas

A1.

(a) Si una partida no ha terminado al final del tiempo establecido para la sesión, el Árbitro exigirá al jugador que esté en juego que "selle" su jugada. El jugador debe anotar su jugada de forma clara e inequívoca en su planilla, poner su planilla y la de su adversario en un sobre, cerrar el sobre y, sólo entonces, parar su reloj sin poner en marcha el de su adversario. Hasta que haya parado los relojes, el jugador mantiene el derecho a cambiar su jugada secreta. Si, tras haber sido advertido por el Árbitro para que selle su jugada, el jugador realiza una jugada sobre el tablero, debe anotar dicha jugada en su planilla como su jugada secreta.

(b) Un jugador que, estando en juego, aplaza la partida antes del fin de la sesión de juego, se considerará que ha realizado la jugada secreta a la hora establecida para el fin de la sesión.

A2. En el sobre se escribirán los siguientes datos:

(a) los nombres de los jugadores;

(b) la posición inmediatamente anterior a la jugada secreta;

(c) el tiempo empleado por cada jugador;

(d) el nombre del jugador que ha realizado la jugada secreta;

(e) el número de la jugada secreta;

(f) la oferta de tablas, si la propuesta se hizo inmediatamente antes del aplazamiento de la partida;

(g) la fecha, hora y lugar de reanudación de la partida.

A3. El Árbitro comprobará la exactitud de la información escrita en el sobre y es el responsable de su custodia.

A4. Si un jugador propone tablas después de que su adversario haya sellado su jugada, la oferta permanece válida hasta que el adversario la haya aceptado o rechazado, como en el Artículo 9.1.

A5. Antes de reanudarse la partida, se pondrá sobre el tablero la posición inmediatamente anterior a la jugada secreta y se indicará en los relojes el tiempo empleado por cada jugador en el momento del aplazamiento.

A6. Si se acuerdan tablas antes de la reanudación de la partida, o si uno de los jugadores notifica al Árbitro que abandona, finaliza la partida.

A7. El sobre se abrirá sólo cuando esté presente el jugador que debe responder a la jugada secreta.

A8. Excepto en los casos mencionados en los Artículos 6.9 y 9.6, pierde la partida un jugador cuya anotación de la jugada secreta:

(a) sea ambigua; o

(b) sea falsa, de tal forma que sea imposible establecer su significado real; o bien

(c) resulte ser una jugada ilegal.

A9. Si a la hora establecida para la reanudación:

(a) está presente el jugador que tiene que responder a la jugada secreta, se abre el sobre, se ejecuta la jugada secreta sobre el tablero y se pone en marcha su reloj.

(b) no está presente el jugador que tiene que responder a la jugada secreta, se pondrá en marcha su reloj. Al llegar, el jugador puede detener su reloj y requerir la presencia del Árbitro. Es entonces cuando se abre el sobre y se ejecuta la jugada secreta sobre el tablero, volviéndose a poner en marcha su reloj.

(c) no está presente el jugador que selló su jugada, su adversario tiene derecho a anotar su respuesta en la planilla, sellar su jugada en un nuevo sobre, detener su reloj y poner en marcha el del jugador ausente, en lugar de ejecutar su jugada en la forma normal. Si así fuere, el sobre se lo pasará al Árbitro para su custodia y se abrirá a la llegada del jugador ausente.

A10. Perderá la partida el jugador que se presente ante el tablero con más de una hora de retraso sobre la hora programada para el comienzo de la sesión (salvo que las bases de la competición o el Árbitro decidan otra cosa).

No obstante, si el ausente es el jugador que hizo la jugada secreta, la partida se decide de otra forma, si:

(a) el jugador ausente ha ganado la partida porque la jugada secreta da mate; o

(b) el jugador ausente ha entablado la partida porque la jugada secreta genera una posición de rey ahogado o como la descrita en el Artículo 9.6; o bien

(c) el jugador presente ante el tablero ha perdido la partida conforme al Artículo 6.9.

A11.

(a) Si se pierde el sobre que contiene la jugada secreta, la partida continuará a partir de la posición y con los tiempos anotados en el momento del aplazamiento. Si no se pudiera restablecer el tiempo consumido por cada jugador, los relojes serán ajustados por el Árbitro. El jugador que hizo la jugada secreta, ejecuta sobre el tablero la jugada que declara ser la que selló.

(b) Si es imposible restablecer la posición, se anula la partida y deberá jugarse una nueva.

A12. Si, en la reanudación de la partida y antes de hacer su primera jugada, uno de los jugadores muestra que el tiempo consumido ha sido puesto incorrectamente en alguno de los relojes, el error debe ser corregido. Si no se prueba el error en ese momento, la partida continúa sin hacer correcciones, salvo que el Árbitro considere que las consecuencias serían demasiado graves.

A13. La duración de cada sesión de reanudación estará controlada por el reloj del Árbitro. Las horas de comienzo y finalización serán anunciadas con antelación.

B. Ajedrez rápido

B1. Una "partida de ajedrez rápido" es aquella en la que todas las jugadas deben efectuarse en un tiempo fijo entre 15 y 60 minutos por jugador.

B2. El juego se regirá por las Leyes del Ajedrez de la FIDE, excepto en lo que quede anulado por las reglas de este Apéndice B.

B3. Los jugadores no están obligados a anotar las jugadas.

B4. Una vez efectuadas tres jugadas por cada jugador, no pueden hacerse reclamaciones relativas a una incorrecta ubicación de las piezas, la colocación del tablero o los tiempos que figuran en los relojes.

B5. El Árbitro adoptará una decisión con respecto a los Artículos 4 y 10 sólo a requerimiento de uno o de ambos jugadores.

B6. Se considera que la bandera ha caído cuando un jugador ha hecho una reclamación válida al efecto. El Árbitro se abstendrá de señalar una caída de bandera.

B7. Para reclamar una victoria por tiempo, el reclamante debe detener ambos relojes y notificárselo al Árbitro. Para que la reclamación prospere, la bandera del reclamante debe permanecer en alto, y la de su adversario caída, después de que se hayan parado los relojes.

B8. Si ambas banderas han caído, la partida es tablas.

C. Ajedrez relámpago

C1. Una "partida de ajedrez relámpago" es aquella en la que todas las jugadas deben efectuarse en un tiempo fijo menor a 15 minutos por jugador.

C2. El juego se regirá por las Leyes del Ajedrez rápido, como en el Apéndice B, excepto en lo que quede anulado por las siguientes reglas de este Apéndice C.

C3. Una jugada ilegal se ha completado en cuanto se haya puesto en marcha el reloj del adversario. Es entonces, antes de hacer su propia jugada, cuando el adversario está autorizado a reclamar la victoria. Una vez que el adversario ha hecho su jugada, no puede corregirse una jugada ilegal.

C4. Para poder ganar, un jugador debe tener "potencial de mate". Esto se define como la disposición adecuada de fuerzas capaz de producir eventualmente una posición legal, incluyéndose el "mate ayudado", en la que un adversario, estando en juego, no pudiera evitar que le den mate en la jugada siguiente. Así, dos caballos y rey contra sólo el rey es insuficiente, pero una torre y rey contra caballo y rey es suficiente.

C5. El artículo 10.2 no es de aplicación.

D. Finales a caída de bandera cuando no hay un Árbitro en el local de juego

D1. Cuando se disputan partidas de acuerdo con el Artículo 10, un jugador puede reclamar tablas cuando le queden menos de dos minutos en su reloj y antes de que caiga su bandera. Esto finaliza la partida.

Puede reclamar en base a que:

(a) su adversario no puede ganar por procedimientos normales, o

(b) que su adversario no ha estado haciendo ningún esfuerzo para ganar por procedimientos normales.

En (a) el jugador debe anotar la posición final y su adversario verificarla.

En (b) el jugador debe anotar la posición final y presentar como prueba una planilla actualizada, que debe ser completada antes de la suspensión de la partida. El adversario verificará tanto la planilla como la posición final.

La reclamación será remitida a un Árbitro cuya decisión será definitiva.

E. Notación Algebraica

La FIDE sólo reconoce para sus propios torneos y matches un sistema de anotación, el Sistema Algebraico, y recomienda también el uso de este sistema uniforme de anotación ajedrecística para literatura y revistas de ajedrez. Las planillas en las que se utilice un sistema de anotación distinto al algebraico, no podrán ser utilizadas como prueba en casos en los que la planilla de un jugador se usa para tal fin. Un Árbitro que observe que un jugador utiliza un sistema de anotación distinto al algebraico, advertirá al jugador en cuestión de este requisito.

Descripción del Sistema Algebraico

E1. Cada pieza es indicada por la primera letra, en mayúscula, de su nombre. Por ejemplo: R = Rey, D = Dama, T = Torre, A = Alfil, C = Caballo.

E2. Para la letra inicial del nombre de una pieza, cada jugador es libre de usar la primera letra del nombre utilizado normalmente en su país. Por ejemplo: A = Alfil (en español); F = Fou (en Francés); L = Loper (en Alemán). En publicaciones impresas se recomienda el uso de figuras.

E3. Los peones no son indicados por su primera letra, sino que se les reconoce precisamente por la ausencia de la misma. Por ejemplo: e5, d4, a5.

E4. Las ocho columnas (de izquierda a derecha para las Blancas y de derecha a izquierda para las Negras) son indicadas por las letras minúsculas: a, b, c, d, e, f, g, h, respectivamente.

E5. Las ocho filas (de abajo arriba para las Blancas y de arriba abajo para las Negras) están numeradas: 1, 2, 3, 4, 5, 6, 7, 8, respectivamente. Consecuentemente, en ha posición inicial las piezas y peones blancos se colocan en la primera y segunda filas; las piezas y peones negros en la octava y séptima filas.

E6. Como consecuencia de las reglas anteriores, cada una de las 64 casillas está indicada invariablemente por una sola combinación de una letra y un número.

E7. Cada movimiento de una pieza se indica por (a) la letra inicial del nombre de la pieza en cuestión y (b) la casilla de llegada. No hay guión entre (a) y (b). Por ejemplo: Ae5, Cf3, Td1. En el caso de peones, sólo se indica la casilla de llegada. Por ejemplo: e5, d4, a5.

E8. Cuando una pieza realiza una captura, se inserta una x entre (a) la letra inicial del nombre de la pieza en cuestión y (b) la casilla de llegada. Por ejemplo: Axe5, Cxf3, Txd1.

Cuando un peón realiza una captura, no se indica sólo la casilla de llegada, sino que debe indicarse también la columna de partida seguida de una x. Por ejemplo: dxe5, gxf3, axb5. En el caso de una captura "al paso", se pone como casilla de llegada la que finalmente ocupa el peón que realiza la captura, añadiéndose a la anotación "a.p.".

E9. Si dos piezas idénticas pueden moverse a la misma casilla, la pieza que se mueve se indica como sigue:

> (1) Si ambas piezas están en la misma fila: por (a) la letra inicial del nombre de la pieza, (b) la columna de la casilla de salida y (c) la casilla de llegada.
>
> (2) Si ambas piezas están en la misma columna: por (a) la letra inicial del nombre de la pieza, (b) la fila de la casilla de salida y (c) la casilla de llegada.
>
> (3) Si las piezas están en filas y columnas distintas, es preferible el método (1).

En el caso de una captura, debe insertarse una x entre (b) y (c).

Ejemplos:

> (1) Hay dos caballos, en las casillas g1 y e1, y uno de ellos mueve a la casilla f3: según el caso, puede ser Cgf3 o Cef3.
>
> (2) Hay dos caballos, en las casillas g5 y g1, y uno de ellos mueve a la casilla f3: según el caso, puede ser C5f3 o C1f3.
>
> (3) Hay dos caballos, en las casillas h2 y d4, y uno de ellos mueve a la casilla f3: según el caso, puede ser Chf3 o Cdf3.

Si se da una captura en la casilla f3, se modifican los ejemplos anteriores insertando una x: según el caso, puede ser (1) Cgxf3 o Cexf3, (2) C5xf3 o C1xf3, (3) Chxf3 o Cdxf3.

E10. Si dos peones pueden capturar la misma pieza o peón del adversario, el peón que se mueve se indica por (a) la letra de la columna de salida, (b) una x, (c) la casilla de llegada. Por ejemplo: si hay peones blancos en las casillas c4 y e4 y un peón o pieza negro en la casilla d5, la anotación para la jugada de Blancas puede ser, según el caso: cxd5 o exd5.

E11. En el caso de la promoción de un peón, se indica el movimiento efectivo del peón, seguido inmediatamente por la letra inicial de la nueva pieza. Por ejemplo: d8D, f8C, b1A, g1T.

E12. La oferta de tablas se anotará como (=).

Abreviaturas esenciales:

0-0 enroque con la torre de h1 o de h8 (enroque por el flanco de rey o enroque corto).

0-0-0 enroque con la torre de a1 o de a8 (enroque por el flanco de dama o enroque largo).

x captura.

+ jaque.

++ o # mate.

a.p. captura de peón al paso.

Ejemplo de partida: 1.d4 Cf6 2.c4 e6 3.Cc3 Ab4 4.Ad2 0-0 5.e4 d5 6.exd5 exd5 7.cxd5 Axc3 8.Axc3 Cxd5 9.Cf3 b6 10.Db3 Cxc3 11.bxc3 c5 12.Ae2 cxd4 13.Cxd4 Te8 14.0-0 Cd7 15.a4 Cc5 16.Db4 Ab7 17.a5 ... etc.

CAPITULO

4
Sistemas Expertos

Concepto: ”Son programas de computadora diseñados para resolver problemas que normalmente son solucionados por expertos humanos en una rama del conocimiento especifica, y mediante esquemas propios del cerebro humano tales como memorización, razonamiento y aprendizaje automático para apoyar a un usuario en la toma de decisiones”.

Sin embargo es de hacer notar la dificultad para entender este concepto sin antes tratar a grandes riesgos la forma como un experto humano puede trabajar, si habláramos de un experto en mecánica automotriz que trata de encontrar la falla en un coche, en principio realiza una revisión del vehículo, reúne información relativa al comportamiento del mismo, los sistemas de que consta, el patrón de comportamiento bajo ciertas fallas y si dicho patrón se encuentra presente realiza un diagnostico y repara el coche.

Si se analiza a un medico éste primero hace una revisión de ciertos parámetros de su paciente, si éstos no son los marcados por los estándares tales como temperatura, pulso el ritmo cardiaco, etc., puede inferir la presencia de una posible enfermedad que incluso deberá corroborar mediante análisis clínicos que tendrían que ser aplicados por otro experto.

Existen varias justificaciones para el diseño de un sistema experto:

1. Hacer que el conocimiento sea más accesible para una mayor cantidad de usuarios.
2. Que económicamente el conocimiento este al alcance de mas usuarios.
3. Lograr la trascendencia del conocimiento a través del tiempo sin importar la posible desaparición del experto humano.

Tipos:

1. Basados en reglas. 2. Basados en la probabilidad.

Elementos de un sistema experto:

1. Base del conocimiento.
   En ella se almacena el conocimiento proporcionado por el experto humano en una área de la ciencia o el conocimiento humano.
2. Motor de inferencia.
   Se encarga de realizar las inferencias en la base del conocimiento de acuerdo con los parámetros definidos en la heurística del sistema y que permitirá inferir la solución de un problema o llevar a cabo la toma de

decisiones la cual podría ser diagnosticar una posible enfermedad a partir de la relación síntoma - enfermedad.

Frente a una situación dada, detecta los conocimientos que interesan, los utiliza, los encadena y construye un esquema de resolución independientemente del dominio y especificidad del problema.

3. Subsistema de explicación.

   Permite que el usuario no experto pueda entender los parámetros de funcionamiento del sistema sin necesidad de tener que ver mas allá de la aplicación del mismo, es decir, un paciente lo que necesita saber es solo los resultados de un sistema experto medico y el porque de las resoluciones sin tener que adentrarse en situaciones complejas que solo se encuentran dentro del ámbito del experto humano.

4. Adquisición del conocimiento.

   Es la interfaz que permite al experto humano mantener actualizada la base del conocimiento. Es de hacer notar que este elemento contempla la interacción entre el experto humano y el diseñador del sistema quien se encarga de extraer la esencia del conocimiento humano para agregarlo a la base del conocimiento del SE.

5. Subsistema de aprendizaje.

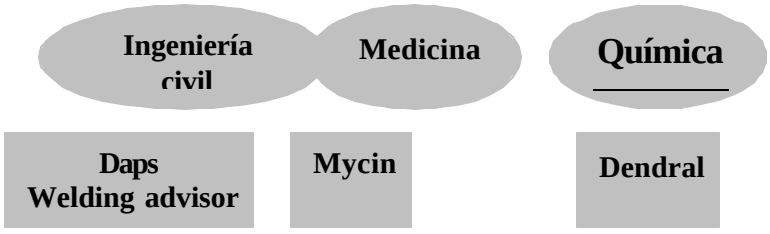
   Este es un elemento incluido en forma reciente gracias al avance que se ha tenido en lo que se ha denominado aprendizaje automático y permite al sistema experto definir mediante experiencias realizar la adquisición automática del conocimiento.

6. Interfaz de usuario. Comunica al motor de inferencia las consultas del usuario y a este ultimo los resultados de la consulta. Por lo regular su interacción se da mediante procesamiento de lenguaje natural

PRINCIPALES APLICACIONES

Medicina: Internist, Mycin, Casnet, Caduceus.
Matemáticas: Macsima, Excheck, Wumpus.
Química: Dendral, Congen, Crysalis.
Geología: Prospector, Expert.
Ingeniería: Sacon, Spear.
Lenguaje natural: Hersay, Sir, Harpy, Lunar.

Ejemplos de Sistemas Expertos



TIPOS DE SISTEMAS EXPERTOS

1. Probabilísticas. 2. Basados en reglas.

1. SISTEMAS DE TIPO PROBABILISTICO

ELEMENTOS:

a) Base del Conocimiento: es el centro del sistema conformado por el espacio probabilístico el cual forma lo que se denomina conocimiento abstracto o de aplicación general. En el caso de un sistema para el diagnostico medico lo conformarían los síntomas y las enfermedades las cuales son tratadas desde un enfoque binario es decir, sobre una población determinada se inferirá la posibilidad de tener una enfermedad sobre la base de su sintomatología.
b) El motor de inferencias: se basa en las probabilidades condicionales y consiste en actualizar las probabilidades de que el paciente en estudio tenga una determinada enfermedad a medida que van conociéndose los síntomas que padece o se van incorporando nuevos datos, como los análisis clínicos, por ejemplo.

Así cuando un paciente acude a la consulta, tendrá unas probabilidades iniciales de poseer cada una de las n enfermedades posibles E1, E2, .. , En, dadas simplemente por:

$P(Ei)=|Ei|/N$, i=1, 2, ,..,n

Donde N y |Ei| representan el numero total de pacientes que han asistido anteriormente a la consulta y el numero de ellos que tenían la enfermedad i-ésima respectivamente.

Tan pronto como un paciente aparece en el consultorio, nuevos datos son incorporados tales como: edad, sexo, aspecto, etc, lo cual modifica las probabilidades de las diferentes enfermedades. Por ello es necesario actualizar las probabilidades iniciales con los nuevos datos.

De esta forma, y a medida que se van incorporando nuevos síntomas o datos las probabilidades van cambiando hasta que una o varias de ellas alcanzan un valor suficientemente próximo a la unidad y las demás, valores suficientemente próximos a cero. En teoría los valores óptimos deberían ser 1, para una o varias de las enfermedades y cero para el resto.

La probabilidad Ei condicionada por el síntoma Aj, se designa por P(Ei/Aj).

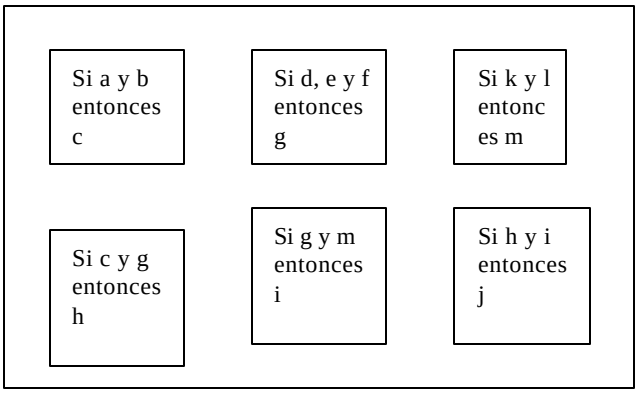
Una buena forma de calcular estas probabilidades es utilizar el teorema de Bayes

SISTEMAS BASADOS EN REGLAS

ELEMENTOS:

a) Base del Conocimiento: en ella se almacenan los datos a partir de su concepción como objetos, relaciones y niveles de jerarquía marcados por la herencia.

   Cuando las premisas de algunas reglas coinciden, en su totalidad o en parte, con las conclusiones de otras, se produce lo que se llama un encadenamiento de reglas. Por ejemplo, en la figura siguiente cuando se tienen dos reglas encadenadas, la conclusión de una coincide con la premisa de la otra.



b) El motor de inferencias: Las reglas sirven para obtener nuevos hechos o conclusiones a partir de verdades o hechos iniciales, ya que si el hecho representado por la premisa de una regla es cierto, el hecho representado por su conclusión también lo es. Se llamaran conclusiones simples a aquellas que resultan de la aplicación de solo una regla y conclusiones compuestas a las que resultan del encadenamiento de varias reglas. Tanto los hechos iniciales como los que resultan de la aplicación de las reglas forman el conocimiento concreto, que reside en la memoria de trabajo.

Para la obtención de conclusiones se utilizan diferentes tipos de estrategias de inferencia y control del razonamiento, así, para las simples existen dos estrategias: modus ponens y modus tollens.

El modus ponens afirma que si si se tiene la regla: "si A es cierto entonces B es cierto".

El modus tollens afirma que si se tiene la regla anterior y se sabe que B es falso, entonces se puede inferir que A también lo es.

Sin embargo el encadenamiento de dos o mas reglas no siempre conduce a sacar conclusiones, ya que puede o no conocerse la verdad o falsedad de las premisas, ante esta situación el SE podrá tomar las siguientes determinaciones:

a) abandonar la regla por no poder concluir nada, o

b) preguntar, a través del subsistema de demanda de información, sobre la verdad o falsedad de las premisas y continuar el proceso hasta producirse una conclusión.

De acuerdo con lo anterior se pueden determinar niveles de probabilidad de que una premisa sea cierta o falsa, y así podríamos decir que A implica a B con una probabilidad P(B/A) (probabilidad de B condicionada por A o probabilidad de B supuesto que A es cierta).

En la tabla siguiente se hace una comparación de las características de cada uno de los dos enfoques descritos anteriormente.

ELEMENTOS	MODELO PROBABILISTICO	MODELO BASADO EN REGLAS
Base de conocimiento	Abstracto: Estructura probabilística (sucesos dependientes). Concreto: hechos	Abstracto: Reglas Concreto: hechos
Motor de inferencia	Evaluación de probabilidades condicionales mediante el teorema de Bayes	Encadenamiento hacia atrás y hacia adelante
Subsistema de explicación	Basado en probabilidades condicionales	Basado en reglas activas
Adquisición del conocimiento	Espacio probabilístico Parámetros	Reglas factores de certeza
Subsistema de aprendizaje	Cambio en la estructura del espacio probabilistico Cambio en los parámetros	Nuevas reglas Cambio en los factores de certeza
Ventajas	Motor de inferencia rápido Aprendizaje paramétrico fácil Propagación de incertidumbre fácil	Explicación fácil Solo implicaciones deseadas
Desventajas	Elevado numero de parámetros Implicaciones superfluas	Motor de inferencia lento Dificultad de propagación de incertidumbre

FASES PARA EL DESARROLLO DE UN SISTEMA EXPERTO

Se utiliza un enfoque similar al método del ciclo de vida para el desarrollo de software descrito en los paradigmas de la Ingeniería de Software y abarca las siguientes etapas:

a) Planteamiento del problema.
b) Integración del equipo de desarrollo el cual debe incluir al menos un experto humano en la disciplina relativa al problema y un ingeniero de software.
c) Diseño del sistema mediante un lenguaje de alto nivel como Prolog, Lisp, C, o Delphi, herramientas de diseño de sistemas asistido por computadora acorde con el tipo de arquitectura decidida por el equipo de desarrollo.
   Es de hacer notar que se da una mayor importancia a la funcionalidad del sistema con respecto a las interfaces de usuario las cuales pueden carecer de una presentación estética.
   En esta etapa se incluyen las estructuras de almacenamiento para la base del conocimiento, el motor de inferencia, el subsistema de explicación y la interfaz de usuario.
d) Creación y prueba de un prototipo.
e) Depuración del prototipo. Esta etapa se da a partir de la interacción del usuario con el sistema y los parámetros de funcionamiento del mismo.
f) Mantenimiento y actualización. Esta etapa podría realizarse con el apoyo del experto humano o mediante aprendizaje automático.

ARQUITECTURA

Existen cuatro categorías en las cuales se engloban los elementos de un SE:

a) arquitecturas de pizarrón. Su nombre proviene del hecho de que contiene un repositorio o memoria común (el pizarrón) donde se escriben datos iniciales, hechos reducidos, conclusiones o cualquier otra información relevante para la solución del problema, el cual es accesible a diferentes fuentes de conocimiento que resuelven el problema en forma cooperativa. Cada fuente del conocimiento que puede ser una regla, un conjunto de reglas o cualquier otra estructura que representa conocimiento, tiene tres partes:
   1) una que le indica al motor de inferencia cuando puede ser activada,
   2) otra que le indica cuando es deseable que sea ejecutada y,
   3) una tercera donde se describen las acciones que deben realizarse al ser ejecutada.

   La comunicación entre las fuentes del conocimiento es por medio del pizarrón, en donde pueden leer los datos necesarios para resolver el subproblema de su especialidad y anotar los resultados de su trabajo.

b) arquitecturas de capas en ellas los operadores que resuelven el problema y la información utilizada por estos operadores se estructuran en capas (layers). La capa inferior utilizada por estos operadores que actúan directamente sobre el estado de solución del problema y las capas superiores tienen meta-operadores que actúan sobre los operadores de la capa inmediatamente inferior. La comunicación entre capas se realiza mediante una interfaz de mensajes, que solamente es accesible a las dos capas entre las cuales se encuentra el mensaje.

c) Arquitecturas para un tipo de problema. Están diseñadas para resolver un tipo de problema en particular. Los elementos que las conforman incluyen mecanismos de representación, inferencia y control que combinan algunos de los paradigmas de las arquitecturas de pizarrón y de capas.
d) Arquitecturas distribuidas. Consiste en combinar SE desarrollados de manera independiente para lograr una mayor capacidad para resolver problemas complejos. En una arquitectura de este tipo del concepto de fuentes del conocimiento se generaliza en el concepto de agentes. Un agente puede ser una fuente del conocimiento, un conjunto de fuentes del conocimiento, un SE completo, un objeto o cualquier otro elemento de la arquitectura. Adicionalmente a los agentes se tiene una red que permite interactuar y negociar a los distintos agentes para resolver el problema. Entre los elementos que se pueden enumerarse se encuentran el conocimiento, el control, la credibilidad de las inferencias, ejecución de acciones, la percepción de señales, los recursos usados por el SE, el almacenamiento de información, la evaluación de resultados, y la explicación de la solución o del proceso de inferencia.

REPRESENTACIÓN DEL CONOCIMIENTO.

Representar el conocimiento en una computadora, consiste en encontrar una correspondencia entre el mundo exterior un sistema simbólico que permite el razonamiento. Existen tres formas de representar el conocimiento a saber:

a) Procedimental. Que utilizan lenguajes de alto nivel para mediante autómatas finitos expresar las interrelaciones entre fragmentos de conocimiento que son difícilmente modificables.
b) Representación declarativa. Se utiliza como base el calculo de predicados, reglas de producción y redes semánticas para crear fragmentos de conocimiento independientes unos de otros y que, por consiguiente, son fácilmente modificables. Estos conocimientos se combinan, después, mediante un mecanismo de razonamiento y deducción.
c) Representación mixta. Utiliza objetos estructurados: esquemas, marcos, grafismos, etc., y utiliza una mezcla de los dos métodos anteriores.

CAPITULO

5
Procesamiento del Lenguaje Natural

Concepto: ”es el área de la IA que se encarga de crear condiciones similares a las seguidas por un humano para el procesamiento del lenguaje natural a nivel texto y fonético”.

Fases del P LN

- Síntesis automática del habla: en esta etapa se definen los dispositivos hardware mediante los cuales una señal fonética es almacenada, adaptada al medio de almacenamiento para su posterior aplicación.

- Comprensión automática a nivel texto: mediante esta etapa es posible definir una aplicación que pueda interpretar los símbolos que integran el alfabeto para con ello realizar una acción determinada, un ejemplo de esto son los analizadores de léxico que se pueden utilizar en un lenguaje de programación en su etapa de interpretación o compilación de código fuente.

- Reconocimiento automático del habla: esta etapa es la mas compleja pues involucra reconocer las palabras de un mensaje oral, sus contenidos sintácticos y semánticas así como sus contenidos pragmáticos.

AREAS QUE LO CONFORMAN:
PROSODIA: ritmo y entonación.
FONOLOGIA: examina los sonidos que conforman el lenguaje
MORFOLOGIA: los morfemas o los componentes para formar las palabras de un lenguaje tales como los prefijos((un, no, anti, etc), los sufijos(ando, mente, etc).
SINTAXIS: estudia las reglas para combinar las palabras dentro de frases bien estructuradas.
SEMANTICA: considera el significado de las palabras, frases y oraciones y el modo en el cual ese significado es convertido en lenguaje natural.
PRAGMATICA: es el estudio de la manera en la cual el lenguaje es utilizado y sus efectos al utilizar una determinada palabra.
CONOCIMIENTO DE LAS PALABRAS: incluye el conocimiento del mundo físico, la interacción de los humanos en sociedad, el rol de los objetivos e intenciones en la comunicación de las personas.

CAPITULO

6
ROBÓTICA

Antecedentes

Desde épocas inmemoriales el hombre ha tenido la ilusión de crear algún tipo de vida a imagen y semejanza de si mismo y entre los que destacan ejemplos como la vieja leyenda judía del golem, un homínido fabricado con barro y animado por la sabia repetición de palabras y signos cabalísticos del rabino Loew en el siglo XVI para defender el ghetto judío de Praga.

Otro ejemplo lo da el gran medico del siglo XVI Paracelso, quien describió una receta para fabricar un ser humano distinta al proceso natural: " se deja pudrir el esperma de un hombre en un recipiente durante cuatro días o hasta que, al final, comience a vivir, moverse y fijarse. Pasado ese tiempo, se parece hasta cierto punto, a una criatura humana pero aun es translucida y carente de cuerpo. Tras este tiempo, se nutre a diario y se alimenta cautelosa y prudentemente con el arcano de la sangre humana y se mantiene durante 400 semanas con el calor continuo e igual de un vientre equino; entonces se transformará en un bebe verdadero y vivo, con todos los miembros de que esta provisto el nacido de una mujer; pero mucho mas pequeño.

Se trata aquí del denominado homúnculo, que después debe criarse con el mayor cuidado y celo, hasta que se desarrolle y comience a adquirir inteligencia".

En la edad media se fabricaron innumerables autómatas mecánicos para adornar las catedrales de la época como parte de los relojes y en pequeñas cajas musicales.

Otro ejemplo de lo anterior lo representa Frankenstein una obra de ciencia ficción escrita por Mary Wollstonecraft Godwin Selley en 1818, en la cual narra la historia del doctor Víctor Frankenstein un moderno prometeo, el cual en un laboratorio con sofisticados artilugios, crea vida artificial en un monstruo que llega a ser temido y acaba destruyendo a su propio creador.

Etimológicamente la palabra robot proviene del idioma checo de la palabra *robota* que significa esclavitud/servidumbre aunque más asociada al trabajo, habiéndose acuñado en una obra escrita en 1921 por Karel Capeck titulada "Rossum´s Universal Robots".

Según el diccionario Webster robot es un dispositivo automático que efectúa funciones ordinariamente asociadas a los seres humanos. Sin embargo la definición aportada por the Robot Institute of America y que se aplica a los robots industriales es la mas completa: "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

De las anteriores definiciones se deriva la definición de robótica como la ciencia que estudia los robots aportada por Isaac Asimov en sus obras de ciencia ficción sobre robots escritas en la década de 1940 a 1950, sobresaliendo las tres leyes de la robótica enunciadas en su libro "yo robot":

1. Un robot no debe dañar a un ser humano o, por su inacción, dejar que un ser humano sufra daño.
2. Un robot debe obedecer las ordenes que le son dadas por un ser humano, excepto cuando estas ordenes se oponen a la primera ley.
3. Un robot debe proteger su propia existencia hasta donde esa protección no entre en conflicto con la primera o la segunda ley.

TIPOS:

Según su forma

- 1. Poliarticulados: brazos mecánicos y automatismos.
- 2. Móviles: robots rodantes y robots multipodos (mecatrones).

Según su uso

- robots industriales

- militares
- animatrones o promocionales
- educativos
- médicos
- domésticos o personales

Arquitectura

A) Tipo Cartesiano
B) Articulado
C) Cilíndrico
D) Polar o Esférico

Ponderación de las arquitecturas.

Configuración	Ventajas	Desventajas
Tipo cartesiano (x, y z - base viajera, extensión y altura)	Tres ejes lineales fáciles de visualizar, estructura rígida, fácil de programar fuera de línea.	Solo puede realizar extensiones en frente de si mismo, el tamaño del área de trabajo es igual al robot por lo cual la estructura en la cual se monta el robot puede abarcar una gran extensión sin que esto sea necesario.
Articulado tipo SCARA (Base rotacional, extensión, ángulo y altura)	Ideal para una línea de montaje por su analogía con un brazo humano, ocupa poco espacio y es multitareas dado que puede tener como elemento terminal distintos efectores, Puede estar dotado de sensores para manipulación de objetos o evasión de obstáculos.	Debe de estar fijo sobre su base por lo que su área de trabajo no puede abarcar 360 grados, no puede tener mas de 6 grados de libertad, difícil de programar fuera de línea.
Cilíndrico (Base Rotacional, extensión, ángulo de elevación.)	Dos ejes lineares, dos ejes de rotación, puede extenderse alrededor de si mismo,	Eje de rotación de la base más rígido que el eje lineal, no puede esquivar obstáculos.
Polar o esférico (base rotacional, estiramiento y	Tres ejes rotacionales, largo estiramiento	No puede esquivar obstáculos, poco

ángulo de elevación)	horizontal	estiramiento vertical.

Según la arquitectura del robot se pueden deducir los grados de libertad de que consta, se le llama grado de libertad al tipo de movimiento que puede desarrollar una articulación; por analogía el brazo de una persona esta dotado por seis grados de libertad de acuerdo a lo siguiente, la primera articulación que es la base, es decir el hombro puede realizar movimientos de giro, el codo realiza movimientos de estiramiento y contracción, el antebrazo de giro y por ultimo, la muñeca realiza tres grados de libertad a saber: de giro, arriba - abajo e izquierda - derecha.

Dispositivos de potencia y control

A) Hidráulicos
B) Neumáticos
C) Motores paso a paso y servomotores
D) Sensorica
E) Autómatas programables.
F) Herramientas: manos, pinzas, pistolas para pintar, equipo de soldadura por puntos.

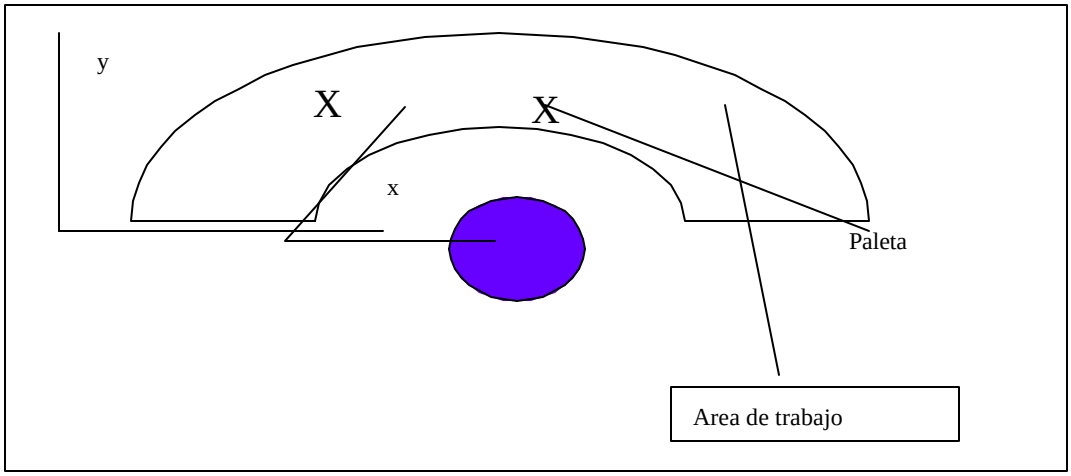
PLANIFICACION DE TAREAS

Se deben contemplar tres aspectos:

a) La planificación de los movimientos. En este punto se define en forma previa la secuencia de movimientos a efectuar por el robot para la realización de una tarea, esto implica la existencia de una rutina de trabajo. De acuerdo con lo anterior el robot será capaz de efectuar una serie de trayectorias las cuales combinadas dejaran al efector final en posición para la labor asignada, la cual puede ser soldadura por puntos, pintura, perforación o manipulación de objetos.

b) El control del robot. Es el esquema mediante el cual este recibe las secuencias de movimientos o rutinas a realizar las cuales pueden ser introducidas previamente mediante un dispositivo de enseñanza como un tablero o un joystick, rutinas previamente definidas por un microprocesador o la ejecución de un programa de computadora que se diseño en un lenguaje de alto nivel.

c) El área de trabajo. Ésta depende de la arquitectura del robot y consta de todos los puntos coordenados que el dispositivo robótico puede tocar con su efector final, tanto en forma contraída como extendida. A cada punto se le llama paleta, ocupa un lugar en el espacio y para poder encontrar su posición es necesario efectuar el calculo cinemático correspondiente, como se mostrará mas adelante.



LENGUAJES DE ROBOT

Esquemas:

a) Reconocimiento de palabras separadas. Mediante este esquema el robot reconoce ordenes que lo motiva a realizar movimientos, acciones o realizar tareas puede ser a nivel texto o a nivel fonético, aunque esto ultimo requiere de arreglos de procesamiento de lenguaje natural.

b) Enseñanza y repetición. También llamada programación gestual requiere conectar al robot un dispositivo de entrada de datos en tiempo real como puede ser un joystick, o un tablero de control que permita introducir las rutinas de movimientos al robot las cuales van quedando grabadas en una memoria intermedia que puede ser regrabada en cuanto cambie la rutina asignada al robot.

c) Lenguajes de programación de alto nivel. Son lenguajes diseñados específicamente para el control de robots realizados en lenguajes de programación de alto nivel como Pascal, C o incluso Basic, los cuales definen una estructura en la cual existen comandos como MOV, GRID, SPEED, etc., que programan efectivamente al robot para la realización de tareas de movimiento, velocidad o agarre.

LENGUAJES DE PROGRAMACION DE ALTO NIVEL

Este tipo de lenguajes se clasifica en dos categorías:

a) Programación orientada a robot. Se utiliza un lenguaje orientado a robot (AML por ejemplo) y cada instrucción del lenguaje define una acción a realizar por el robot, de tal manera que conforme se va ejecutando una línea de código del programa el robot va efectuando una tarea o movimiento.
b) Programación orientada a nivel de tarea.

Esta se describe como una secuencia de objetivos a lograr con el posicionamiento de los objetos y no como un conjunto de movimientos necesarios para que se logren estos objetivos.

PRINCIPALES LENGUAJES DE ROBOT UTILIZADOS

LENGUAJE	DISEÑADOR	APLICACIÓN	LENGUAJE BASE
AL	Stanford	Brazo	Pascal
AML	IBM	Brazo	Pascal, Lisp, APL
AUTOPASS	IBM		PL/1
MCL	Mc Donell	Brazo	APT
VAL	Unimate	Brazo tipo Puma	Basic
RAIL	Automatrix	Brazo cartesiano	Pascal

CINEMATICA

Mediante de esta disciplina se define la trayectoria de un brazo de robot a través de su área de trabajo con respecto a un sistema de coordenadas acordes con la arquitectura del mismo. La cinemática se encarga del estudio y la descripción analítica del desplazamiento del robot en el espacio y sobre una base de tiempo, tomando en cuenta la posición de las articulaciones y la posición y orientación del efector final del robot.

Para las variables de posición de las articulaciones se considera el problema cinemático directo y para la posición y orientación del efector se toma en cuenta el problema cinemático inverso.

ESTADO DEL ARTE

En la actualidad la robótica se encuentra en una etapa de avance como nunca antes lo había estado los desarrollos son muy variados y sus aplicaciones cada vez se acercan al concepto original mediante el cual fue concebida esta área de la inteligencia artificial es decir crear dispositivos que sustituyan al hombre en tareas para las cuales es preferible arriesgar el funcionamiento de una maquina y no perder una vida humana.

Las siguientes imágenes son solo algunas de las arquitecturas que forman parte de los prototipos de actualidad.

Imagen 1. Brazo de robot articulado tipo puma

Imagen 2. Brazo de robot tipo Scara

Imagen 3. Brazo articulado para labores de pintura.

Imagen 4. Robot tricept.

CASO PRACTICO1: PROYECTO MECANO

Este proyecto se desarrolló por el grupo de inteligencia artificial de la Facultad de Ingeniería Mecánica de la Universidad de Colima, y consiste en un brazo de robot que en su primer diseño contó con 5 grados de libertad, sin embargo en su modelo actual es de cuatro grados de libertad, su arquitectura

es de tipo articulado tomando como modelo el tipo puma mostrado en la figura 1.

Consta de elementos motores paso a paso como elementos motrices distribuidos de la siguiente forma: Base, con movimiento circular, Antebrazo, con movimiento de torsión, muñeca, en ella se colocaron dos motores que realizan movimientos de elevación y giro y un ultimo motor en la pinza que realiza tareas de agarre o sujeción.

Para la articulación del codo se utilizó un pistón neumático debido a que esta articulación tiene asignada la función de soportar todo el esfuerzo que implica la realización de una tarea o rutina de trabajo, su movimiento es lineal y requiere utilizar un circuito independiente para encender y apagar la compresora que alimenta de aire al pistón. Consulte las imágenes 5 y 6.

El software utilizado para el control de Mecano originalmente se desarrollo utilizando el lenguaje pascal, sin embargo actualmente se utiliza Prolog debido a que es un lenguaje utilizado en aplicaciones de inteligencia artificial, debido a lo anterior la programación de Mecano es de tipo textual, es decir, las rutinas de trabajo se generan mediante líneas de código de Prolog, que se ejecutan en forma recursiva, sin embargo, se esta diseñando el modulo de enseñanza asistida mediante un tablero de mando para dotar a Mecano de un modulo de programación gestual.

Imagen 5. Mecano, brazo de robot articulado desarrollado en la Universidad de Colima

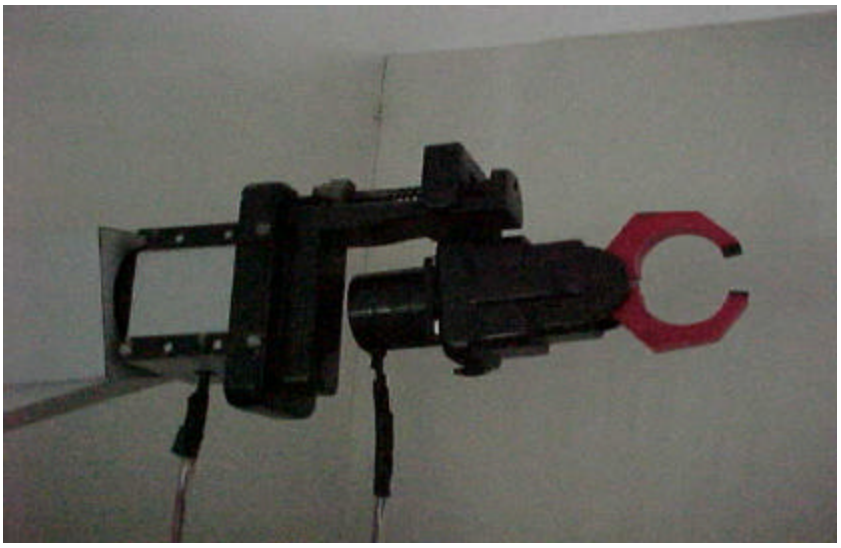


Imagen 6. Pinza o efector final de Mecano.

HARDWARE UTILIZADO

Se diseño una interfaz para el control de motores paso a paso, compuesta por una fuente de poder de DC, una señal de DC usualmente enviada por un generador de pulso, un motor de paso y un driver de potencia. Cada motor es activado por su propio driver, su esquema de funcionamiento se muestra en la figura numero 6, y su diagrama eléctrico en la figura 7.

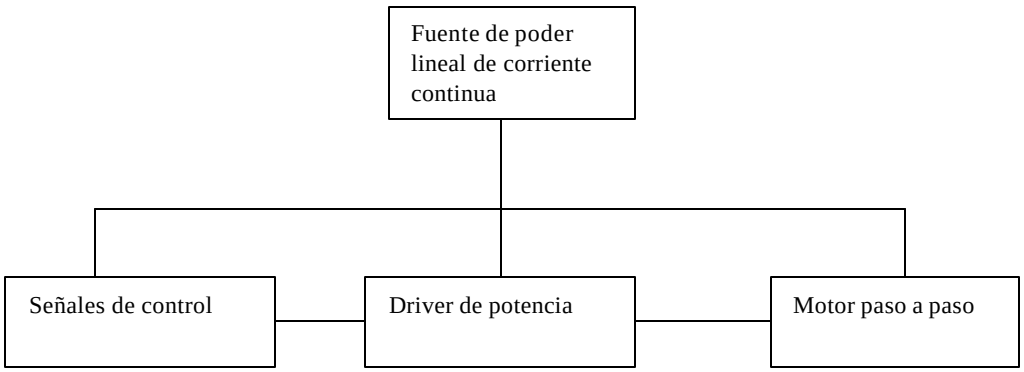


Figura 6. esquema de funcionamiento del driver de potencia.

El modo de excitación utilizado para los motores determina la velocidad de accionamiento de la articulación a mover, en este caso es posible trabajar a distintas velocidades programando el accionar de los motores a media, una o doble fase. La interfaz se diseño en base al 74LS08 (consultar figura 7), aunque también es posible utilizar el SAA1042a (consulte figura 8), que son Circuitos integrados que funcionan muy bien en funciones de control de motores de este tipo.

Cada uno de los drivers de potencia esta compuesto por un opto aislador a foto resistor como base para la excitación de transistores BJT da alta potencia, esta configuración permite el posible reemplazo de motor o transistor en forma totalmente independiente al generador de pulsos.

El generador de pulsos se implemento por medio de una rutina software, lo anterior permite controlar de manera eficaz la velocidad de ejecución de las secuencias de movimiento de los motores.

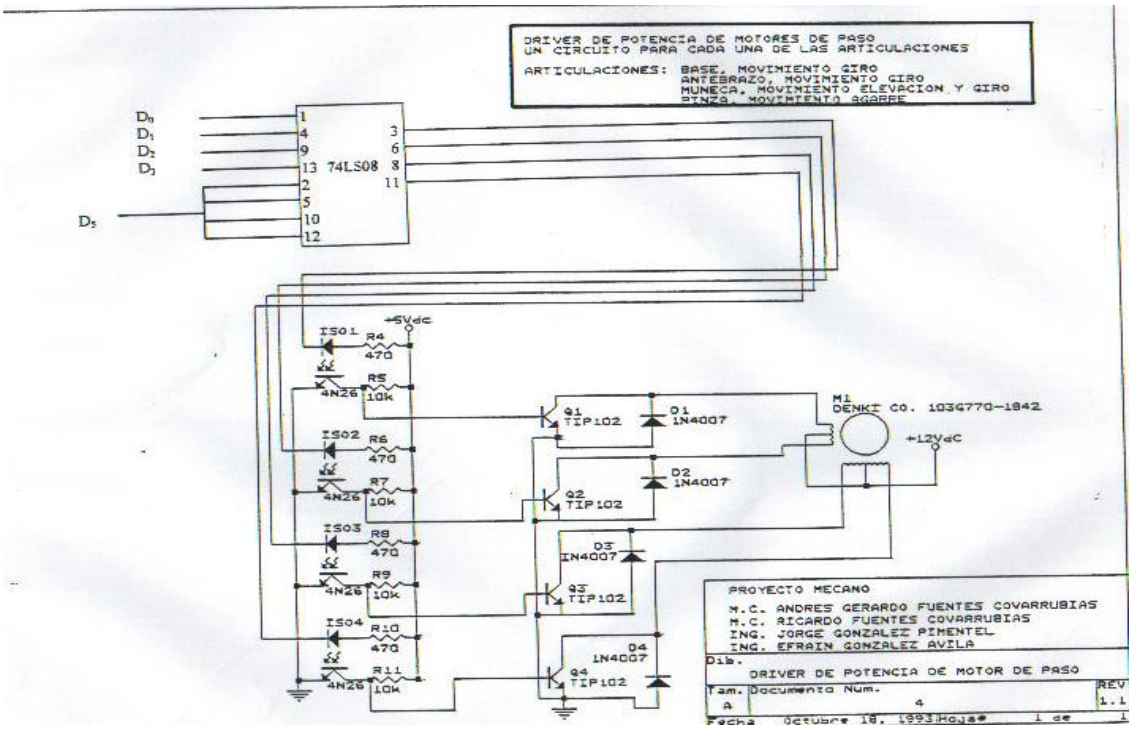


Figura 7. driver de potencia mediante el Chip 74LS08

Una variante podría ser:

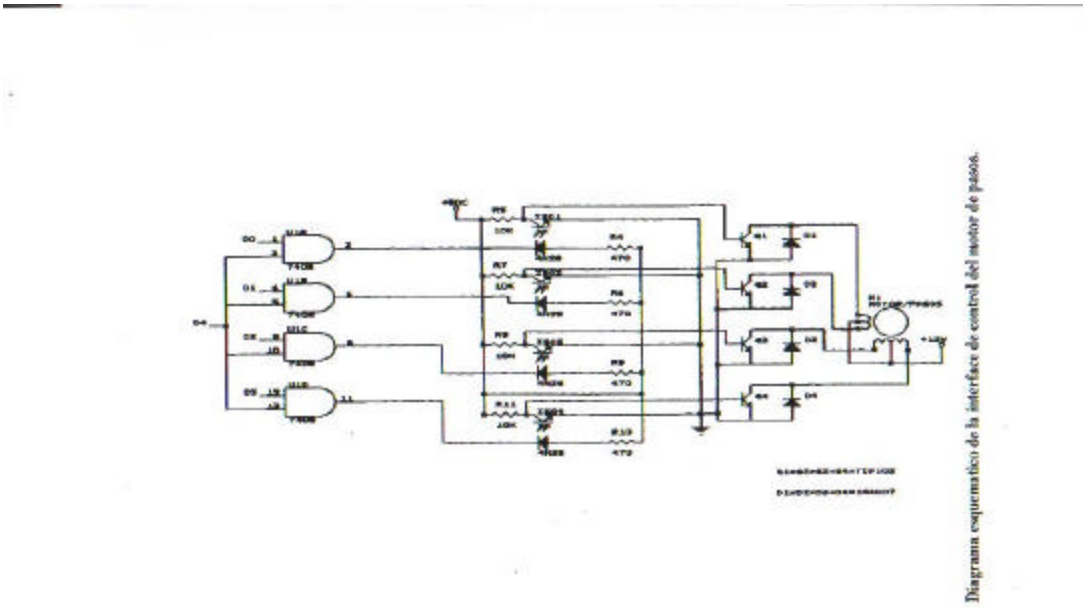


Figura 8. driver de potencia mediante el Chip SAA1042a

Un arreglo mas compacto es el mostrado en la figura 9, el cual utiliza el Chip ULM2003 en el control lógico y en la parte de potencia se utilizan los Chips 74LS194 y 74LS85.

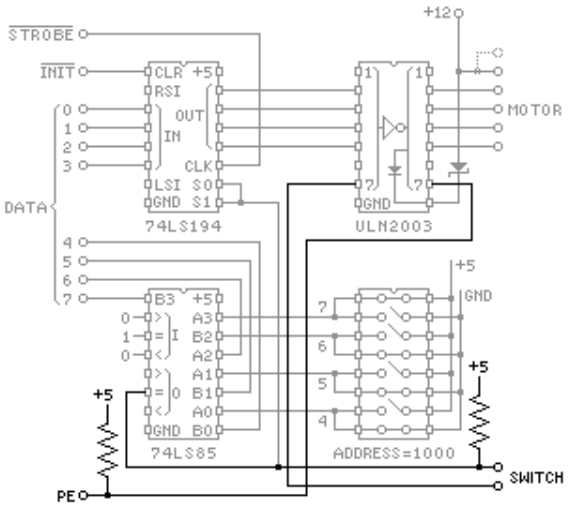


Figura 9. El circuito de control utilizando el Chip ULN2003.

El circuito impreso se muestra a continuación:

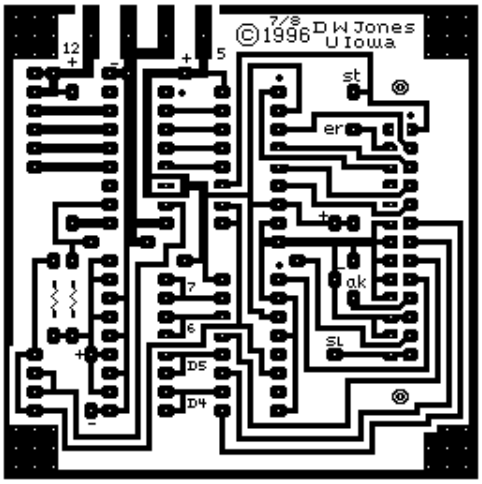


Figura 10. controlador de motores PAP

La instalación deberá coincidir con la figura 11.

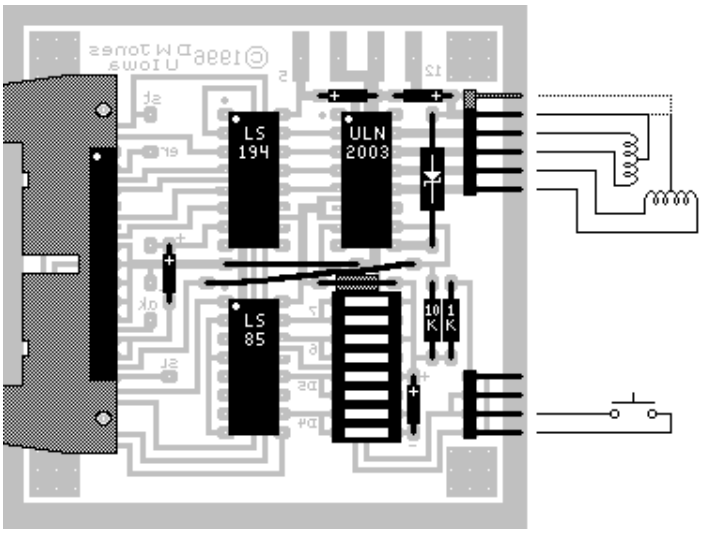


Figura 11. El impreso concluido.

CASO PRACTICO2: PROGRAMANDO UN BRAZO DE ROBOT TIPO PUMA

Este caso practico incluye la programación de un brazo de robot tipo puma marca Mitsubishi modelo RV-M1 el cual tiene las siguientes características:

Es un brazo articulado de 5 ejes, pinza, controlador, unidad de instrucción, caja de simulación así como los periféricos que complementan la unidad robótica. En la figura 12 se muestra la instalación general del sistema.

Figura 12. El brazo de robot y sus accesorios.

Funcionamiento.

1. La unidad básica de instrucción (Teaching box).

   Se utiliza para el manejo del robot en la modalidad de enseñanza y permite definir lo movimientos que realizara durante la ejecución de una o varias tareas. Con la unidad de instrucción puede hacerse mover el brazo del robot en modo PTP (punto a punto), XYZ, así como en modo herramienta.

   En modo PTP, se controla el movimiento el movimiento individual de los ejes del robot. En modo XYZ se puede desplazar el brazo de robot paralelamente a los ejes del sistema de coordenadas cartesianas básicas. En modo herramienta, se puede desplazar el efector a lo largo de un eje vertical el área de la brida de la herramienta. Consulte la figura 13 mostrada a continuación.

   Figura 13. unidad básica de instrucción

2. Primer ejercicio. El brazo de robot deberá ser colocado en su punto de referencia también denominado origen, para ello deberá colocar en modo de encendido el tablero de enseñanza y posteriormente pulsar la tecla NST seguida de la tecla ENT.

a. Trabajando en modo PTP. Seleccione el modo PTP pulsando las teclas

TPP

ENT

En la unidad de instrucción; a continuación se deberá mover los ejes del brazo del robot secuencialmente utilizando las teclas en la sección para funciones de movimientos.

Tecla para Movimiento en Sentido positivo (antohorario)	Tecla para Movimiento en Sentido negativo (horario)	Descripción de los ejes	Unión De ejes
B+	B-	Cuerpo	J 1
S+	S-	Hombro	J 2
E+	E-	Codo	J 3

b. Trabajando en modo XYZ. Seleccionar este modo pulsando las teclas:

XYZ

ENT

En la unidad de instrucción. Mover el brazo del robot utilizando las teclas en la sección de funciones de movimiento de la unidad de instrucción.

A continuación mover el brazo de robot de tal forma que la pinza apunte hacia abajo en ángulo, mientras que al mismo tiempo se verifica el manejo en los modos PTP y XYZ.

c. Seleccionar el modo herramienta. Pulsando las teclas

TOOL

ENT

En la unidad de instrucción. A continuación mover el brazo de robot de acuerdo a lo siguiente:

Tecla para movimiento en sentido positivo	Tecla para movimiento En sentido negativo
Z+	Z-

El modo herramienta es el modo de recorrido que desplaza la pinza a lo largo del eje que esta posicionado verticalmente respecto de la superficie de la brida de la herramienta.

Para abrir y cerrar la pinza pulse las teclas <O> y >C<.

2. Posicionamiento del brazo del robot utilizando la unidad de instrucción.

   Además de alcanzar posiciones, el almacenamiento y borrado de datos también forman parte de las funciones básicas de la unidad de instrucción.

   Una posición se almacena bajo un numero de posición y contiene las posiciones angulares de los cinco ejes del robot. Las teclas para lo números de posición se hayan en el teclado de las funciones de movimiento.

   Las posiciones dentro del espacio operativo pueden ser alcanzadas punto a punto en cualquier secuencia que se quiera por medio de los números de posición.

   a. Asignando una secuencia de movimientos:

      [P.S.] + [Numero] + [ENT]

      La secuencia anterior permite definir las coordenadas de la posición presente del robot en una posición con el numero especificado. Si un solo numero es asignado a dos posiciones diferentes, la definida al ultimo toma precedencia.

   b. Eliminando el contenido de una posición con el numero especificado.

      [P. C.] + [Numero] + [ENT]

c. Transferir el contenido del EEPROM del usuario a la RAM de la unidad motora.

TRN + ENT

3. Procedimiento de fijación de posición del brazo del robot utilizando la unidad de instrucción.

Este procedimiento se hace para efectuar los movimientos de los ejes con tanta precisión como sea posible y nos es necesaria si el robot se mueve solamente a través de una serie de puntos enseñados. Empero, si se van a usar comandos en el sistema cartesiano de coordenadas, tales como comandos de paletas, esta fijación debe ser hecha entes de la enseñanza. Una vez hecha y almacenada en la memoria EPROM se procede a trabajar en modo de programación.

a. Definición de posiciones.

Oprima las teclas de movimientos necesarias para mover el brazo a una posición apropiada.

Suponga que una posición adecuada seria la posición “1”. Ahora apriete las siguientes teclas sucesivamente:

P. S. 1 ENT

Esto fija la posición 1.

b. Verificación de posiciones.

Para verificar la posición 1, oprima las siguientes teclas sucesivamente:

MOV 1 ENT

Si la posición ha sido correctamente definida, el brazo se moverá al punto definido en el punto anterior.

c. cambio de posiciones.

Para cambiar o redefinir las posiciones previamente definidas:

c.1. Mueva el brazo a una posición distinta de 1 y oprima las teclas:

P. S. 1 ENT

Esto limpia los datos de la vieja posición y redefinirá la posición 1.

c.2. De la misma manera, redefina 1 y 2 (en el supuesto de que existieran).

d. Eliminación de posiciones.

d.1. Para eliminar la posición 1, oprima las siguientes teclas sucesivamente:

P. C.	1	ENT

Esto limpia la posición 1 haciéndola disponible para una nueva definición.

d.2. Para verificar que la posición 1 a sido eliminada apropiadamente, oprima las siguientes teclas sucesivamente:

MOV	1	ENT

Si los datos de la posición han sido correctamente eliminados, el LED indicador del status de la caja de enseñanza muestra " " lo que indica que la función invocada no puede ser realizada.

4. Generación y ejecución de programas.

En este apartado se describen los procedimientos requeridos para generar un programa en el modo computadora, usando las posiciones previamente definidas y ejecutarlo. Es importante mencionar que al momento de la ejecución el interruptor ON/OFF de la caja de enseñanza está en la posición OFF.

4.1 Generación y transferencia de un programa.

4.1.1 Mediante la caja de enseñanza genere al menos tres posiciones, lo siguiente lista la secuencia del programa, en que los números al principio representan loa números de línea del programa movemaster ubicado en

el archivo COSIPROG ubicado en el escritorio o dentro del menú de opciones programa de Windows.

```
10 NT       ; fijación del origen (anidación)
12 SP 7     ; Fija velocidad en 7.
14 MO 1,O   ; muévete a la posición 1 con la mano abierta.
16 MO 2, C  ; muévete a la posicion2 con la mano cerrada.
18 MO 3, C  ; muévete a la posición 3 con la mano cerrada.
20 TI 30    ; detente durante 3 segundos.
22 GT 14    ; salta a la línea 14.
```

4.1.2 Ejecucion por pasos:

El programa generado puede ser ejecutado, línea por línea, operando las teclas de la caja de enseñanza para verificación.

a. ponga ON el interruptor ON/OFF de la caja de enseñanza.
b. Para ejecutar el programa comenzando con el numero de línea 10, oprima las siguientes teclas sucesivamente en ese orden.

STEP	10	ENT

Esto ejecuta el comando “NT” de la línea numero 10.

c. Después de que ha sido ejecutado el comando “NT”, el LED de la caja de enseñanza muestra el numero de la línea de programa subsecuentente. Para ejecutar la línea 12, teclee:

STEP	ENT

CAPITULO

7
Visión de maquina

Esta disciplina de la IA, nació en la década de los 60´s con la idea básica de conectar una cámara de video a una computadora.

Los primeros trabajos supusieron que el trabajar con una imagen en 3D era un problema sencillo dada la “facilidad” con que los humanos realizan esa tarea, sin embargo después de los primeros ensayos se concluyo que reproducir el SVH, es una tarea ardua como se describe en el transcurso del capitulo.

Se define como los procesos de obtención, caracterización e interpretación de información de imágenes tomadas de un mundo tridimensional.
Dichos procesos se dividen en seis etapas:
Captación, Preprocesamiento, segmentación, descripción, reconocimiento e interpretación.

También se le llama visión por computadora o visión artificial, se encarga del procesamiento de imágenes fijas o en tiempo real es decir en movimiento a una velocidad no menor de 30 cuadros por segundo, que es la resolución que tiene el ojo humano para las siguientes tareas:

- Reconocimiento de objetos, fijos o en movimiento.
- Ubicación del lugar en el cual se encuentra un objeto.
- Definición morfológica o física de un objeto.
- Identificación de los puntos o forma geométrica que tiene un objeto para su posterior manipulación.
- Realizar mediciones para poder identificar la distancia que guarda un objeto con respecto de otro.

Visión de bajo nivel.

Comprende las etapas de capitación y preprocesamiento, la primera de ellas se apoya en dispositivos de adquisición de imágenes los cuales pueden ser a nivel percepción mediante sensórica o mediante cámaras que pueden ser vidicón o CCD (estado sólido).
En cuanto al preprocesamiento, este comprende distintas técnicas de filtrado con el fin de generar una imagen lo mas nítido posible. Se utilizan dos métodos: a) dominio del espacio: se refiere al conjunto de pixeles que conforman una imagen.
b) Dominio de frecuencias: trabaja con conjuntos de pixels complejos resultado de aplicar la transformada de Fourier.

Visión intermedia.
Comprende las etapas de segmentación, descripción y reconocimiento, la primera de ellas se encarga de definir los algoritmos computacionales que

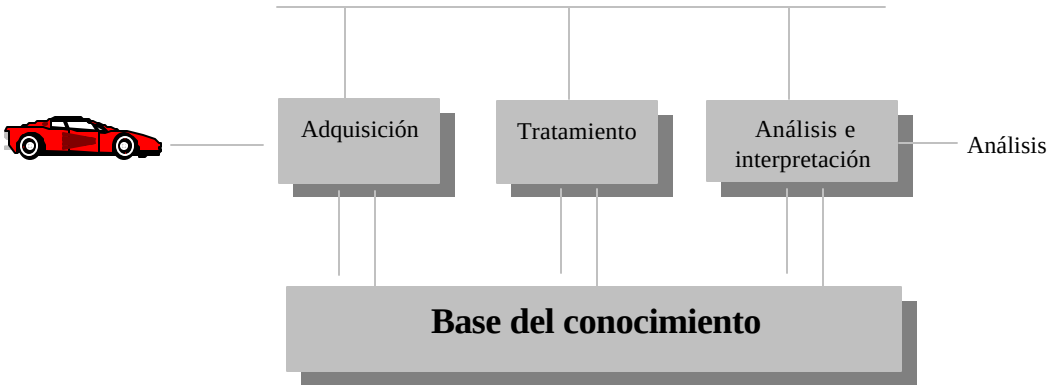
permitan resaltar características básicas de cada objeto con el fin de su clasificación para efectuar un proceso de reconocimiento posterior.

Visión de alto nivel.
En esta etapa se desarrollan los algoritmos computacionales orientados a la emulación del proceso de visión humana y que permiten reconocer objetos y tomar decisiones o efectuar tareas la etapa que la conforma es la interpretación.

Sistemas de visión artificial

El inicio de todo un proceso de visión artificial se da con la adquisición de la imagen la cual se puede efectuar utilizando una cámara de video de estado sólido o vidicón. Posteriormente la imagen es adquirida por la computadora a través de una tarjeta de procesamiento de imágenes, la tercera etapa es el procesamiento de la imagen definiendo niveles de grises o tonos de color, sus contornos o formas geométrica y por ultimo la imagen ya procesada es enviada a un dispositivo que definirá la aplicación a dar a la imagen.

Etapas:



Adquisición.

Cuando la imagen es captada por la cámara, es representada por una matriz numérica que representa la imagen por niveles de luminancia y los cuales son identificados como puntos gráficos o pixeles que en el caso de tratarse de una imagen monocroma tendrá una resolución de 512 filas por 512 columnas.

Tratamiento

Los primeros trabajos enfocados al procesamiento de imágenes se deben al matemático francés Joseph Fourier a quien se debe el teorema que lleva su nombre y que indica que cualquier función con significado físico se puede obtener sumando funciones sinusoidales.

Análisis e interpretación.

Los sistemas de análisis e interpretación se encargan del estudio de escenas complejas, el análisis de los objetos que conforman las escenas implica el desarrollo de potentes técnicas de clasificación para interpretar parte de la información que conforma cada tipo de objeto, compararlo con una base de conocimiento y a partir de ello analizar si cumplen con las características definidas y el resultado seria un sistema que identifique un camino, el rostro de una persona o las placas de un vehículo.

Procesamiento digital de imágenes.

Se inicio en 1921 transmitiendo imágenes de Londres a NY por un cable submarino.
En 1963 IBM inicia el tratamiento digital de imágenes procedentes de satélites artificiales.
Se orienta en dos direcciones:
1.Extraer la información de una imagen y proceder a su análisis
2.Partir de un información y generar en base ella, imágenes
Sistemas para el proceso de imágenes:
a) Óptico
b) Digital

Etapas del procesamiento digital de imágenes.

1. Se capta una imagen con un elemento Sensible a la luz
2. Sobre la imagen contenida en el elemento sensible se proyecta una retícula que la divide en pequeños cuadros, en cada uno se mide la luminosidad.
3. Con los niveles luminosos se obtiene una matriz de valores que expresan la luminosidad de cada punto, los cuales representan las diferentes tonalidades de gris comprendidas entre el blanco y el negro.
4. La composición de la imagen se realiza implementando sobre su matriz de cuadrados los niveles de gris de la etapa anterior

Cuantos mas cuadrados integren la imagen y mas niveles de gris existan, mayores serán su resolución y definición.

Tratamiento de las señales.

Las cámaras captan una imagen de una escena y generan una señal de salida, en América el formato es de 525 líneas y en Europa es de 625.

La señal de una cámara se aplica a un Conversor A/D que la convierte en números binarios equivalentes a la intensidad luminosa de cada punto de la imagen. Cada numero producido representa un nivel de gris.

Si se dispone de una tarjeta de 8 bits la señal contendrá tantos niveles de gris como combinaciones se puedan hacer con ellos: 2_8=256.

Manejo de la memoria.

Una imagen B/N se compone de mas de 2 millones de bits ejemplo:

Si la imagen contiene 525 líneas y cada línea contiene 525 píxeles y si cada píxel admite 256 niveles de gris esto resulta: 525X525x8=2,205,000 bits.
En las aplicaciones en tiempo real esto es critico porque se trabaja con secuencias de imágenes no menores a 25 cuadros por segundo.

Algoritmos para el proceso de imágenes.

Los principales algoritmos son SRI (Standord Researsh Institute)

Lo que debe satisfacer una imagen para poder aplicar un algoritmo SRI es:
1.Debe ser binaria
2.Las partes del objeto no se tocan
3.La imagen es estable.

CAPITULO

8
Redes Neuronales

- **Concepto:** Sistema computacional que consta de un gran numero de elementos simples interconectados, que procesan la información respondiendo dinámicamente frente a estímulos externos..

Componentes: una red esta compuesta por nodos o neuronas, que están unidas mediante conexiones a las cuales se le asigna un peso numérico. Los pesos constituyen el principal recurso de memoria de la red, a largo plazo.

Estructuras de red. 1. Redes de alimentación progresiva (son unidireccionales y no hay ciclos) 2. Redes recurrentes (utilizan conexiones bidireccionales con pesos simétricos).

NEURONAS

Es un dispositivo que transforma (en su SOMA, o cuerpo celular) varias señales de entrada (DENDRITAS) en una única salida (AXON). Las entradas pueden proceder de otras neuronas o bien ser entradas a la red desde el exterior. La salida puede transmitirse a otras neuronas o funcionar como señal de salida a la red, en cuyo caso el comportamiento es ligeramente diferente en cuanto a las funciones que se le aplican o el uso final que se hace de ella.

Una neurona es un microprocesador simple, con una capacidad limitada de computo, restringida a un conjunto elemental de instrucciones(sumas y productos) y una memoria pequeña para almacenar pesos y activaciones.

La función computada en cada elemento de procesamiento se divide en varios pasos:

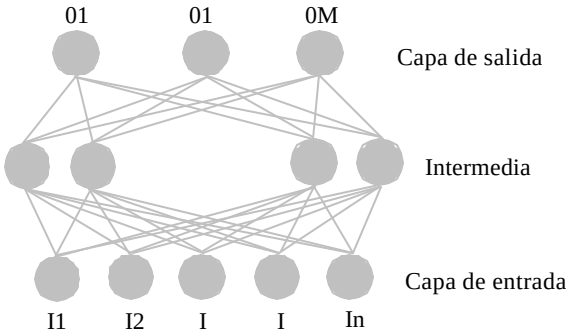
1) Suma ponderada de sus entradas. El lugar de entrada entre cada una de las señales de entrada y la neurona se llama sinapsis. Dichas sinapsis regulan la cantidad de información útil recibida desde cada una de las conexiones.

La expresión matemática que describe la conversión de patrones de entrada en señales de respuesta o salida se llama función de transferencia de la neurona.

2.) Función de activación. Es la función aplicada a la entrada neta de información y consiste en convertir dicha entrada neta en un nivel de activación para la neurona, el cual es equivalente al nivel de excitación de una neurona biológica.

Arquitectura de una red

1) capa de entrada
2) capas intermedias
3) capa de salida



APLICACIONES

- Sistemas de decisión.
- Procesamiento de señales.
- Lenguaje natural.
- Visión artifi cial
- Reconocimiento de texto escrito a mano e impreso.
- Control en tiempo real.
- Problemas de búsqueda de soluciones.
- Predicción de series temporales.
- Clasificación de patrones.

PROYECTOS SUGERIDOS

1. Programa ejemplo de solución de problemas lógicos caso practico: caníbales y misioneros. se recomienda lenguaje Prolog.
2. Programas ejemplo de teoría de juegos: la caza
3. Driver de potencia para control de motores paso a paso .
4. Control de dispositivos con arreglos de sensorica caso practico coche prototipo que se mueve con detección de luz mediante fotoresistencias.
5. Una cabeza de androide con quijada móvil activada por servomotores.
6. Procesador fonético para una cabeza de androide.
7. Control por computadora de una grúa prototipo con tres grados de libertad.

BIBLIOGRAFIA

Angulo José Ma., Curso de robótica, Paraninfo, 1989.
Angulo José Ma., Robótica practica, Paraninfo, 1992.
Angulo José Ma., Iñigo Madrigal R., Visión artificial por computador, Paraninfo, 1986.
Asimov Isaac, Yo robot, Editorial sudamericana, 2002.
L. Fuller James, Robotics introduction, programming, and projects, Mc Millan, 1991.
Fu K. S., González R. C., Lee C. S. G., Robótica control, detección, visión e inteligencia, Mc Graw Hill 1990.
Varios autores, Inteligencia artificial conceptos, técnicas y aplicaciones, Marcombo, 1987.
Stuart Russell, Peter Norvig, Inteligencia artificial un enfoque moderno, Prentice may, 1996.
Castillo Ron Enrique, Alvarez Sainz Elena, sistemas expertos aprendizaje e incertidumbre, Paraninfo, 1989.
Winston Patrick Henry, Inteligencia artificial, Adisson Wesley Iberoamericana, 1992.
Castillo Enrique, Alvarez elena, sistemas expertos aprendizaje e incertidumbre, Paraninfo, 1989.
Dieter Niebendhal, Sistemas expertos parte 2 experiencia de la practica, Marcombo, 1991.
James A. Freeman, M. Skapura David, Redes neuronales algoritmos, aplicaciones y técnicas de programación, 1993.
F. Luger George, A. Stubblefield William, Artificial Intelligence Structures and strategies for complex problem solving, second edition, Addison Wesley, 1993.
Patrick Henry Winston, Inteligencia Artificial, Addison Wesley Iberoamericana, 1994.

CORREO ELECTRÓNICO

Si el lector desea hacer algún comentario, sugerencias o consultas puede contactar al autor mediante los siguientes correos electrónicos:

fuentesr@.ucol.mx
fuentesr@hotmail.com
fuentesr_99@yahoo.com
fuentesr@palmera.colimanet.com

O bien consultar a las siguientes paginas de internet:

http://ciam.ucol.mx
www.geocities.com/fuentesr_99
http://www.members.tripod.com.mx/fuentesr
www.icq.com/51813267

APENDICE

A
EL JUEGO DE LA CAZA

```
/*JUEGO DE TABLERO DESARROLLADO EN TURBO PROLOG VER.2.0 */
/* DISEÑADO POR Ricardo fuentes Covarrubias */
/*Abril de 1999 */


code=4000
trail=1000
heap=6000
errorlevel=1
include "c:\\prolog\\grapdecl.pro"
constants
    bgi_driver ="c:\\prolog\\bgi"

database nume
   posficha(integer,integer,integer,integer,integer,integer,integer,integer)
   posmaquina(integer,integer)
   poscasilla(integer,integer,integer,integer,integer)
predicates
      presenta
      barda
      inicial
      inicialmaquina(integer)
      retarda(integer)
      muevemaquina
      casilla(integer,integer,integer)
      borra
      mover(char,char)
      poshumano
      repetir
      tablero
      rellenar
      fichas
      fin

goal
    initgraph(vga,1,_,_,bgi_driver),
    assert(posficha(112,85,208,85,304,85,400,85)),inicial,
    casilla(112,85,A),casilla(208,85,B),casilla(304,85,C),casilla(400,85,D),casilla(256,316,E),
    assert(poscasilla(A,B,C,D,E)),
    presenta,
    tablero,repetir,closegraph.
```

```
clauses

presenta:-
     setfillstyle(1,12),
     setcolor (1),
     rectangle (1,63,639,349),
     floodfill (2,67,1),
     setcolor(6),
     setfillstyle(1,3),
     rectangle(40,68,424,332),
     floodfill (42,70,6),
     /* Lineas verticales */
     line(88,68,88,332),line(136,68,136,332),
     line(184,68,184,332),line(232,68,232,332),line(280,68,280,332),
     line(328,68,328,332),line(376,68,376,332),
     /* Lineas horizontales */
     line(40,101,424,101),line(40,134,424,134),
     line(40,167,424,167),line(40,200,424,200),line(40,233,424,233),
     line(40,266,424,266),line(40,299,424,299),rellenar,
     setfillstyle(1,3),
     setcolor(11),
     rectangle(0,0,639,60),floodfill(2,2,11),
     setcolor(15),settextstyle(2,0,6),
      outtextxy(218,0,"UNIVERSIDAD DE COLIMA"),
     settextstyle(2,0,5),
     outtextxy(205,15," INTELIGENCIA ARTIFICIAL"),
     setcolor(15),
     /*setfillstyle(1,0),
     rectangle(440,240,630,330),*/
     setcolor(4),
     outtextxy(70,160," LA  C A Z A "),
     readchar(_).
barda:-
    write(' ').
borra:-
    write(' ').

/* Procedimiento que dibuja el tablero y toda la pantalla de presentacion */
 tablero:-
     setfillstyle(1,12),
     setcolor (1),
     rectangle (1,63,639,349),
     floodfill (2,67,1),
     setcolor(6),
     setfillstyle(1,3),
     rectangle(40,68,424,332),
     floodfill (42,70,6),
     /* Lineas verticales */
     line(88,68,88,332),line(136,68,136,332),
     line(184,68,184,332),line(232,68,232,332),line(280,68,280,332),
     line(328,68,328,332),line(376,68,376,332),
     /* Lineas horizontales */
     line(40,101,424,101),line(40,134,424,134),
     line(40,167,424,167),line(40,200,424,200),line(40,233,424,233),
     line(40,266,424,266),line(40,299,424,299),rellenar,
     setfillstyle(1,3),
```

```
    setcolor(11),
    rectangle(0,0,639,60),floodfill(2,2,11),
    setcolor(15),settextstyle(2,0,5),
    outtextxy(218,0,"UNIVERSIDAD DE COLIMA"),
    settextstyle(2,0,4),
    /*outtextxy(4,15,"Materia  : Inteligencia Artificial"),     */
    outtextxy(4,26,"Dise¤o   : Ricardo Fuentes Covarrubias"),
    outtextxy(4,37,"Area     : Sistemas Computacionales"),
    outtextxy(4,48,"Semestre : Noveno"),
    setcolor(15),
    setfillstyle(1,0),
    rectangle(440,240,630,330),
    floodfill(442,282,15),fichas.


/* Procedimiento que rellena cad cuadro del tablero */
 rellenar:-
  setfillstyle(1,15),

  floodfill(42,70,6),floodfill(180,70,6),floodfill(270,70,6),floodfill(370,70,6),
  floodfill(42,158,6),floodfill(180,158,6),floodfill(270,158,6),floodfill(370,158,6),
  floodfill(42,230,6),floodfill(180,230,6),floodfill(270,230,6),floodfill(370,230,6),
  floodfill(42,270,6),floodfill(180,270,6),floodfill(270,270,6),floodfill(370,270,6),

  floodfill(130,130,6),floodfill(190,130,6),floodfill(320,130,6),floodfill(380,130,6),
  floodfill(130,180,6),floodfill(190,180,6),floodfill(320,180,6),floodfill(380,180,6),
  floodfill(130,260,6),floodfill(190,260,6),floodfill(320,260,6),floodfill(380,260,6),
  floodfill(130,300,6),floodfill(190,300,6),floodfill(320,300,6),floodfill(380,300,6).

/* Procedimiento que dibuja las fichas del juego */
fichas:-
        setcolor(15),setfillstyle(9,2),
        posficha(A,B,C,D,E,F,G,H),posmaquina(J,K),
        A1=A-2,B1=B-5,
        C1=C-2,D1=D-5,
        E1=E-2,F1=F-5,
        G1=G-2,H1=H-5,
        J1=J-2,K1=K-5,
        circle(A,B,18),floodfill(A,B,15),outtextxy(A1,B1,"1"),
        circle(C,D,18),floodfill(C,D,15),outtextxy(C1,D1,"2"),
        circle(E,F,18),floodfill(E,F,15),outtextxy(E1,F1,"3"),
        circle(G,H,18),floodfill(G,H,15),outtextxy(G1,H1,"4"),
        setfillstyle(1,7),circle(J,K,18),floodfill(J1,K1,15).

repetir:-poshumano,muevemaquina,repetir,!.
repetir.

/* Procedimiento que lee el movimiento que da el jugador humano */
poshumano:-outtextxy(510,245,"TU TURNO"),attribute(3),
          outtextxy(460,267,"Numero de ficha :"),cursor(19,71),readchar(X),
          write(X),outtextxy(460,290,"Mover a (I/D) :"),readchar(P),upper_lower(D,P),
          borra,mover(X,D),!.

/* Procedimiento que mueve la ficha 1 a la izquierda */
mover('1','I'):- posficha(X,Y,A,B,C,D,E,F),X<>64,Y<>316,
                 X1=X-48,Y1=Y+33,X2=X1-2,Y2=Y1-5,
```

```
                casilla(X1,Y1,Pos),
                poscasilla(M,N,L,O,P),Pos<>N,Pos<>L,Pos<>O,Pos<>P,
                retractall(poscasilla(M,N,L,O,P)),
                assert(poscasilla(Pos,N,L,O,P)),
                retractall(posficha(X,Y,A,B,C,D,E,F)),
                assert(posficha(X1,Y1,A,B,C,D,E,F)),
                setfillstyle(1,3),floodfill(X,Y,6),
                setfillstyle(9,2),
                circle(X1,Y1,18),floodfill(X1,Y1,15),outtextxy(X2,Y2,"1"),!.

/*Procedimiento que mueve la ficha 1 a la derecha */
mover('1','D'):- posficha(X,Y,A,B,C,D,E,F),X<>400,Y<>316,
                X1=X+48,Y1=Y+33,X2=X1-2,Y2=Y1-5,
                casilla(X1,Y1,Pos),
                poscasilla(M,N,L,O,P),Pos<>N,Pos<>L,Pos<>O,Pos<>P,
                retractall(poscasilla(M,N,L,O,P)),
                assert(poscasilla(Pos,N,L,O,P)),
                retractall(posficha(X,Y,A,B,C,D,E,F)),
                assert(posficha(X1,Y1,A,B,C,D,E,F)),
                setfillstyle(1,3),floodfill(X,Y,6),
                setfillstyle(9,2),
                circle(X1,Y1,18),floodfill(X1,Y1,15),outtextxy(X2,Y2,"1"),!.

/* Procedimiento que mueve la ficha 2 a la izquierda */
mover('2','I'):- posficha(A,B,X,Y,C,D,E,F),X<>64,Y<>316,
                X1=X-48,Y1=Y+33,X2=X1-2,Y2=Y1-5,
                casilla(X1,Y1,Pos),
                poscasilla(M,N,L,O,P),Pos<>M,Pos<>L,Pos<>O,Pos<>P,
                retractall(poscasilla(M,N,L,O,P)),
                assert(poscasilla(M,Pos,L,O,P)),
                retractall(posficha(A,B,X,Y,C,D,E,F)),
                assert(posficha(A,B,X1,Y1,C,D,E,F)),
                setfillstyle(1,3),floodfill(X,Y,6),
                setfillstyle(9,2),
                circle(X1,Y1,18),floodfill(X1,Y1,15),outtextxy(X2,Y2,"2"),!.

/* Procedimiento que mueve la ficha 2 a la derecha */
mover('2','D'):- posficha(A,B,X,Y,C,D,E,F),X<>400,Y<>316,
                X1=X+48,Y1=Y+33,X2=X1-2,Y2=Y1-5,
                casilla(X1,Y1,Pos),
                poscasilla(M,N,L,O,P),Pos<>M,Pos<>L,Pos<>O,Pos<>P,
                retractall(poscasilla(M,N,L,O,P)),
                assert(poscasilla(M,Pos,L,O,P)),
                retractall(posficha(A,B,X,Y,C,D,E,F)),
                assert(posficha(A,B,X1,Y1,C,D,E,F)),
                setfillstyle(1,3),floodfill(X,Y,6),
                setfillstyle(9,2),
                circle(X1,Y1,18),floodfill(X1,Y1,15),outtextxy(X2,Y2,"2"),!.

/* Procedimiento que mueve la ficha 3 a la izquierda */
mover('3','I'):- posficha(A,B,C,D,X,Y,E,F),X<>64,Y<>316,
                X1=X-48,Y1=Y+33,X2=X1-2,Y2=Y1-5,
                casilla(X1,Y1,Pos),
                poscasilla(M,N,L,O,P),Pos<>M,Pos<>N,Pos<>O,Pos<>P,
                retractall(poscasilla(M,N,L,O,P)),
                assert(poscasilla(M,N,Pos,O,P)),
```

```
            retractall(posficha(A,B,C,D,X,Y,E,F)),
            assert(posficha(A,B,C,D,X1,Y1,E,F)),
            setfillstyle(1,3),floodfill(X,Y,6),
            setfillstyle(9,2),
            circle(X1,Y1,18),floodfill(X1,Y1,15),outtextxy(X2,Y2,"3"),!.

/* Procedimiento que mueve la ficha 3 a la derecha */
mover('3','D'):- posficha(A,B,C,D,X,Y,E,F),X<>400,Y<>316,
            X1=X+48,Y1=Y+33,X2=X1-2,Y2=Y1-5,
            casilla(X1,Y1,Pos),
            poscasilla(M,N,L,O,P),Pos<>M,Pos<>N,Pos<>O,Pos<>P,
            retractall(poscasilla(M,N,L,O,P)),
            assert(poscasilla(M,N,Pos,O,P)),
            retractall(posficha(A,B,C,D,X,Y,E,F)),
            assert(posficha(A,B,C,D,X1,Y1,E,F)),
            setfillstyle(1,3),floodfill(X,Y,6),
            setfillstyle(9,2),
            circle(X1,Y1,18),floodfill(X1,Y1,15),outtextxy(X2,Y2,"3"),!.

/* Procedimiento que mueve la ficha 4 a la izquierda */
mover('4','I'):- posficha(A,B,C,D,E,F,X,Y),X<>64,Y<>316,
            X1=X-48,Y1=Y+33,X2=X1-2,Y2=Y1-5,
            casilla(X1,Y1,Pos),
            poscasilla(M,N,L,O,P),Pos<>M,Pos<>N,Pos<>L,Pos<>P,
            retractall(poscasilla(M,N,L,O,P)),
            assert(poscasilla(M,N,L,Pos,P)),
            retractall(posficha(A,B,C,D,E,F,X,Y)),
            assert(posficha(A,B,C,D,E,F,X1,Y1)),
            setfillstyle(1,3),floodfill(X,Y,6),
            setfillstyle(9,2),
            circle(X1,Y1,18),floodfill(X1,Y1,15),outtextxy(X2,Y2,"4"),!.

/* Procedimiento que mueve la ficha 4 a la derecha */
mover('4','D'):- posficha(A,B,C,D,E,F,X,Y),X<>400,Y<>316,
            X1=X+48,Y1=Y+33,X2=X1-2,Y2=Y1-5,
            casilla(X1,Y1,Pos),
            poscasilla(M,N,L,O,P),Pos<>M,Pos<>N,Pos<>L,Pos<>P,
            retractall(poscasilla(M,N,L,O,P)),
            assert(poscasilla(M,N,L,Pos,P)),
            retractall(posficha(A,B,C,D,E,F,X,Y)),
            assert(posficha(A,B,C,D,E,F,X1,Y1)),
            setfillstyle(1,3),floodfill(X,Y,6),
            setfillstyle(9,2),
            circle(X1,Y1,18),floodfill(X1,Y1,15),outtextxy(X2,Y2,"4"),!.


mover('1','I'):-borra,beep,
             outtextxy(470,280,"Direccion no valida"),readchar(_),borra,poshumano,!.
mover('2','I'):-borra,beep,
             outtextxy(470,280,"Direccion no valida"),readchar(_),borra,poshumano,!.
mover('3','I'):-borra,beep,
             outtextxy(470,280,"Direccion no valida"),readchar(_),borra,poshumano,!.
mover('4','I'):-borra,beep,
             outtextxy(470,280,"Direccion no valida"),readchar(_),borra,poshumano,!.
mover('1','D'):-borra,beep,
             outtextxy(470,280,"Direccion no valida"),readchar(_),borra,poshumano,!.
```

```
mover('2','D'):-borra,beep,
                outtextxy(470,280,"Direccion no valida"),readchar(_),borra,poshumano,!.
mover('3','D'):-borra,beep,
                outtextxy(470,280,"Direccion no valida"),readchar(_),borra,poshumano,!.
mover('4','D'):-borra,beep,
                outtextxy(470,280,"Direccion no valida"),readchar(_),borra,poshumano,!.

/* Procedimiento que controla en caso de parametros erroneos */
mover(_,_):-borra,beep,
                outtextxy(470,280,"Parametros no validos"),readchar(_),borra,poshumano,!.


muevemaquina:-   posmaquina(A,B),
                A<>64,B<>85,
                X1=A-48,Y1=B-33,
                casilla(X1,Y1,Pos),
                poscasilla(M,N,L,O,P),Pos<>M,Pos<>N,Pos<>L,Pos<>O,
                retractall(poscasilla(M,N,L,O,P)),
                assert(poscasilla(M,N,L,O,Pos)),
                setfillstyle(1,3),floodfill(A,B,6),
                retractall(posmaquina(A,B)),
                assert(posmaquina(X1,Y1)),
                setfillstyle(1,7),
                circle(X1,Y1,18),floodfill(X1,Y1,15),Y1<>85,!.


/* Procedimiento que mueve la ficha de la maquina arriba a la derecha */
muevemaquina:-   posmaquina(A,B),A<>400,B<>85,
                X1=A+48,Y1=B-33,
                casilla(X1,Y1,Pos),
                poscasilla(M,N,L,O,P),Pos<>M,Pos<>N,Pos<>L,Pos<>O,
                retractall(poscasilla(M,N,L,O,P)),
                assert(poscasilla(M,N,L,O,Pos)),
                setfillstyle(1,3),floodfill(A,B,6),
                retractall(posmaquina(A,B)),
                assert(posmaquina(X1,Y1)),
                setfillstyle(1,7),
                circle(X1,Y1,18),floodfill(X1,Y1,15),Y1<>85,!.

/* Procedimiento que mueve la ficha de la maquina abajo a la izquierda */
muevemaquina:- posmaquina(A,B),A<>64,B<>316,B<>85,
               X1=A-48,Y1=B+33, Y2=B-33,
                casilla(X1,Y1,Pos),
                poscasilla(M,N,L,O,P),Pos<>M,Pos<>N,Pos<>L,Pos<>O,Y2<>85,
                retractall(poscasilla(M,N,L,O,P)),
                assert(poscasilla(M,N,L,O,Pos)),
                setfillstyle(1,3),floodfill(A,B,6),
                retractall(posmaquina(A,B)),
                assert(posmaquina(X1,Y1)),
                setfillstyle(1,7),
                circle(X1,Y1,18),floodfill(X1,Y1,15),!.

/* Procedimiento que mueve la ficha de la maquina abajo a la derecha */
muevemaquina:- posmaquina(A,B),A<>400,B<>316,B<>85,
                X1=A+48,Y1=B+33,Y2=B-33,
                casilla(X1,Y1,Pos),
```

```
            poscasilla(M,N,L,O,P),Pos<>M,Pos<>N,Pos<>L,Pos<>O,Y2<>85,
            retractall(poscasilla(M,N,L,O,P)),
            assert(poscasilla(M,N,L,O,Pos)),
            setfillstyle(1,3),floodfill(A,B,6),
            retractall(posmaquina(A,B)),
            assert(posmaquina(X1,Y1)),
            setfillstyle(1,7),
            circle(X1,Y1,18),floodfill(X1,Y1,15),!.

/* Si no se puede mover a ningun lado el juego termina */
muevemaquina:-fin,exit,!.

/* Procedimiento de finalizaci¢n del juego */
fin:-
     borra,outtextxy(495,250,"FIN DEL JUEGO"),
     settextstyle(1,0,6),
     setcolor(4),
     outtextxy(70,160,"G A M E   O V E R"),
     sound(70,600),retarda(300),sound(80,700),retarda(200),sound(90,800),
     retarda(32000),
     setcolor(15),
     settextstyle(2,0,5),
     outtextxy(460,310,"Pulse una tecla . . ."),readchar(_),borra,
     posficha(A,B,C,D,E,F,G,H),posmaquina(I,J),poscasilla(K,L,M,N,O),
     retractall(posficha(A,B,C,D,E,F,G,H)),retractall(posmaquina(I,J)),
     retractall(poscasilla(K,L,M,N,O)),exit,!.

/* Procedimientos que genera la casilla donde se encuentra cada ficha */
casilla(Columna,Fila,PosCasilla):-Col1=(Columna-16)/48,Fil1=(Fila-52)/33,
                          PosCasilla=Col1+(Fil1-1)*8,!.


/* Procedimiento que realiza un retardo */
    retarda(N):-N>0,!,N1=N-1,retarda(N1).
    retarda(0).

/*    Procedimiento que genera un n£mero aleatorio para saber que en que
posicion va a comenzar a jugar la maquina */
inicial:-random(7,A),inicialmaquina(A),!.

inicialmaquina(0):-assert(posmaquina(256,316)),!.
inicialmaquina(2):-assert(posmaquina(64,316)),!.
inicialmaquina(1):-assert(posmaquina(352,316)),!.
inicialmaquina(3):-assert(posmaquina(160,316)),!.
inicialmaquina(5):-assert(posmaquina(256,316)),!.
```