

Unidad 1: Introducción a la inteligencia artificial

¿Qué es la inteligencia artificial?

La **Inteligencia Artificial** tiene por objeto el estudio del comportamiento inteligente en las máquinas. Este comportamiento supone: **percibir**, **razonar**, **aprender**, **comunicarse** y **actuar** en entornos complejos.

- Una de las metas a largo plazo de la IA es el desarrollo de máquinas que puedan hacer cosas igual o mejor que los humanos.

- Otra meta es llegar a comprender este tipo de comportamiento.

La IA persigue al mismo tiempo metas científicas y de ingeniería.

Pensadores:

John Searle, cree que la materia de la que estamos hechos es fundamental para la inteligencia. Las máquinas vivientes están hechas de proteínas.

Newell y Simon: un sistema físico de símbolos es una máquina. No importa de que está hecho el sistema físico.

Otros pensadores: creen que el comportamiento inteligente es el resultado de lo que ellos llaman *procesamiento subsimbólico*, es decir procesamiento de señales no de símbolos.

Test de Turing

En lugar de definir la palabra pensar, propuso un test, mediante el cual pudiera decidirse si una máquina en particular es o no inteligente.

En el juego participan 3 personas, un hombre (A), una mujer (B) y un interrogador (C), que pueden ser de cualquier sexo. El interrogador permanece en una sala, separado de los otros dos, pero pudiendo comunicarse con ellos mediante un teletipo. El objetivo del juego para el interrogador es determinar cual de los otros dos es el hombre y cual es la mujer. Para ello, el interrogador puede plantear preguntas a A y a B.

El objetivo de A es intentar que C haga una identificación errónea.

El objetivo del juego para B es ayudar al interrogador.

Ahora planteamos la siguiente cuestión: ¿Qué sucedería si una máquina interpretase el papel de A en el juego? ¿El interrogador hará tantas identificaciones erróneas como cuando el juego es interpretado por un hombre y una mujer? Estas cuestiones reemplazan a nuestra cuestión original "¿Pueden pensar las máquinas?"

Aproximaciones a la IA.

Los principales paradigmas se clasifican en dos grupos:

Basadas en procesamiento de símbolos

Se sustentan en la hipótesis del sistema físico de símbolos de Newell y Simon, la cual se basa mucho de lo que podríamos llamar la IA clásica. Un miembro de esta familia de aproximaciones es el que se basa en la *aplicación de operaciones lógicas sobre bases de conocimiento declarativo*. Este estilo de IA representa el conocimiento sobre un problema del dominio mediante sentencias declarativas, a menudo basadas en sentencias de la lógica de predicados o sustancialmente equivalentes a ellas. Para deducir consecuencias a partir de este conocimiento se aplican técnicas de inferencia lógica.

En muchas de las aproximaciones basadas en procesamiento de símbolos, el análisis de los comportamientos deseados, o la síntesis de máquinas para conseguirlos, se extienden a través de varios niveles:

- Nivel superior: es el **nivel del conocimiento**, en el cual se especifica el conocimiento necesario para que la máquina alcance sus objetivos.
- **Nivel simbólico**, se representa este conocimiento mediante estructuras simbólicas.
- **Niveles inferiores**, en los cuales, realmente se implementan las operaciones de procesamiento de símbolos. Muchas aproximaciones basadas en procesamiento de

símbolos utilizan una metodología de diseño "descendente", se comienza en el nivel de conocimiento y se procede por los niveles simbólicos y de implementación.

Aproximaciones subsimbólicas

Siguen un estilo de diseño "ascendente".

En los niveles más bajos, el concepto de símbolo no es tan apropiado como el concepto de señal. Entre las aproximaciones subsimbólicas, una es "vida artificial". Para conseguir máquinas inteligentes tendremos que seguir muchos pasos evolutivos. Esta estrategia no solo conducirá a la obtención a corto plazo de máquinas útiles, sino que desarrollará el substrato sobre el cual deben construirse necesariamente los niveles superiores de inteligencia.

Brooks introdujo la hipótesis de los fundamentos físicos. Según su hipótesis, se puede obtener un comportamiento complejo sin usar modelos centralizados; para ello, bastaría con dejar que los diversos módulos de comportamiento de un agente interactúen independientemente con el entorno.

Sistemas que resuelven problemas de la IA.

La IA estudia como lograr que las máquinas realicen tareas que, por el momento, son realizadas mejor por los seres humanos.

Los problemas de la IA.

Los primeros trabajos en este campo hicieron hincapié en *tareas formales* como juegos y demostración de teoremas. Estos comparten la propiedad de que son tareas en las que se considera que es necesaria la inteligencia para desarrollarlas.

Otra incursión son los *problemas que aparecen a diario*, denominados de "sentido común". Estos problemas incluyen el razonamiento sobre objetos físicos y sus relaciones, también sobre acciones y sus consecuencias.

Conforme las investigaciones de la IA progresaron, se realizaron avances en las tareas descritas y aparecieron nuevas áreas que incluyen *percepción, comprensión del lenguaje natural y resolución de problemas en campos especializados*.

Tareas de percepción son difíciles ya que incluyen señales analógicas que suelen contener bastante ruido.

Comprensión del Lenguaje hablado como es difícil de resolver, es posible restringir el problema al lenguaje escrito. No sólo se debe poseer un amplio conocimiento sobre el propio lenguaje sino también manejar el suficiente conocimiento para reconocer las suposiciones.

Tareas más especializadas en las cuales es necesario experiencia.

Las suposiciones subyacentes (VER!!!)

Sistema de símbolo físicos: conjunto de entidades llamadas símbolos, que pueden funcionar como componentes de otro tipo de entidad llamada expresión (o estructura de símbolos). Una estructura de símbolos está compuesta por un número de instancias (señales o tokens) de símbolos relacionados de alguna forma física. El sistema contiene también una colección de procesos que operan sobre expresiones para producir otras expresiones. Un sistema de símbolos físicos es una máquina que produce a lo largo del tiempo una colección evolutiva de estructuras de símbolos.

La hipótesis del sistema de símbolos físicos. **Un sistema de símbolos físicos posee los medios necesarios y suficientes para realizar una acción inteligente genérica.**

Esta hipótesis es sólo una hipótesis, el único camino para determinar su certeza es mediante la experimentación.

¿Qué es una técnica de IA?

La inteligencia necesita conocimiento. El conocimiento posee algunas propiedades poco deseables, como:

- es voluminoso
- es difícil caracterizarlo con exactitud
- cambia constantemente
- EL CONOCIMIENTO se organiza de tal forma que se corresponde con la forma en que va a ser usado. se distingue de los datos en eso.

Una técnica de IA es un método que utiliza conocimiento representado de tal forma que:

- El conocimiento representa las generalizaciones. Se agrupan situaciones que comparten propiedades importantes. Si no posee esta propiedad puede necesitarse demasiada memoria.
- Debe ser comprendido por las personas que lo proporcionan.
- Puede modificarse fácilmente para corregir errores y reflejar los cambios en el mundo y en nuestra visión del mundo.
- Puede usarse en gran cantidad de situaciones aun cuando no sea totalmente preciso o completo.
- Puede usarse para ayudar a superar su propio volumen.

Técnicas de IA:

- **Búsqueda:** proporciona una forma de resolver los problemas en los que no se dispone de un método más directo tan bueno como una estructura en la que empujar algunas técnicas directas existentes.
- **Uso del conocimiento:** proporciona una forma de resolver problemas complejos explotando las estructuras de los objetos involucrados.
- **Abstracción:** proporciona una forma de separar aspectos y variaciones importantes de aquellos otros sin importancia y que en caso contrario podrían colapsar un proceso.

El nivel del modelo

Los esfuerzos dedicados a construir programas que lleven a cabo tareas de la misma forma que el hombre, se dividen en dos clases:

- **Los programas de la primera clase:** problemas que una computadora puede resolver fácilmente, pero cuya resolución implica el uso de mecanismos de los que no dispone el hombre.
- **La segunda clase:** intentan modelar lo humano, son aquellas que realizan tareas que se adecuan claramente con nuestra definición de tareas de IA. Hacen cosas que no son triviales para una computadora.

Razones para querer modelar la forma de trabajar humana para llevar a cabo estas tareas:

- Verificar las teorías psicológicas de la actuación humana.
- Capacitar a las computadoras para comprender el razonamiento humano.
- Capacitar a la gente para comprender a las computadoras.
- Explotar el conocimiento que se puede buscar en el hombre.

Criterios de determinación del éxito

El método de Turing sirve para determinar si una máquina es capaz de pensar.

Se necesitan dos personas y la máquina que se desea evaluar. Una de las personas actúa de entrevistador y se encuentra en una habitación separado de la computadora y de la otra persona. El entrevistador hace preguntas tanto a la persona como a la computadora mecanografiando las cuestiones y recibe las respuestas de igual forma. El entrevistador sólo los conoce como A y B y debe intentar determinar quien es la persona y quien es la máquina. El objetivo de la máquina es hacer creer al investigador que es una persona, si lo consigue, se concluye que la máquina piensa. Se permite que ésta haga cualquier cosa para engañar al entrevistador.

Si lo que se quiere es escribir programas que simulen el comportamiento humano ante una tarea, la forma de medir el éxito está en que el comportamiento del programa se corresponda con el humano. Se busca un programa que falle donde la gente lo hace.

Acciones que debe llevar a cabo el sistema

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

1. Definir el problema con precisión.
2. Analizar el problema.
3. Aislar y representar el conocimiento necesario para resolver el problema.
4. Elegir la mejor técnica y aplicarla al problema particular.

Definición mediante búsqueda en espacio de estados.

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

- Definir un espacio de estados que contenga todas las configuraciones posibles de los objetos más relevantes.
- Identificar uno o más estados que describan situaciones en las que comience el proceso de resolución del problema. (estados iniciales)
- Especificar uno o más estados que pudieran ser soluciones aceptables del problema. (estados objetivos)
- Especificar un conjunto de reglas que describan las acciones disponibles.

Sistemas de producción

Consiste en:

- Un conjunto de reglas compuestas por una parte izquierda (un patrón) que determinar la aplicabilidad de la regla y una parte derecha, que describe la operación que se lleva a cabo si se aplica la regla.
- Una o más bases de datos/conocimiento. Partes de la BD pueden ser permanentes, mientras que otras pueden hacer referencia sólo a la solución del problema actual.
- Una estrategia de control que especifique el orden en el que las reglas se comparan con la BD, y la forma de resolver los conflictos que surjan cuando varias reglas puedan ser aplicadas a la vez.
- Un aplicador de reglas.

Análisis del problema

A fin de poder elegir el método más apropiado para un problema en particular, es necesario analizarlo con arreglo a varias dimensiones claves:

- ¿Puede descomponerse el problema?**- ¿Pueden ignorarse pasos dados o al menos deshacerse si se comprueba que no eran adecuados?**

Clases de problemas:

- **Ignorables:** pueden ignorarse pasos dados (ej.: demostración de teoremas)
- **Recuperables:** pueden deshacerse pasos dados (ej.: 8-puzzle)
- **No recuperables:** no pueden deshacerse pasos dados (ej.: ajedrez)

- ¿Es predecible el universo del problema?

Problemas:

- **de Consecuencia – cierta:** para resolverlo se usa un sistema de lazo abierto, ya que el resultado de una acción se puede predecir perfectamente. La planificación puede utilizarse para generar una secuencia de operadores que garantizan llegar a una solución.

- **de Consecuencia – incierta:** la planificación puede al menos generar una secuencia de operadores que tiene una buena probabilidad de conducir a una solución. La planificación no garantiza una solución, además suele ser muy cara ya que el número de rutas de solución que necesita explorar crece exponencialmente con el número de puntos en los cuales la consecuencia no puede predecirse.

- ¿Una solución es buena de manera evidente, sin necesidad de compararla con todas las demás posibles soluciones?

Los problemas de "el mejor camino" son por lo general más complicados de computar que los problemas "algún camino". Para los problemas de "el mejor camino" se requiere una búsqueda mucho más exhaustiva.

- La solución deseada, ¿es un estado del mundo o una ruta hacia algún estado?

Tenemos problemas en los que la solución es un estado y problemas en los que la solución es un ruta hacia un estado (ej.: jarras de agua).

- ¿Es necesario una gran cantidad de conocimiento para resolver el problema o solo es necesario para restringir la búsqueda?

Los problemas del ajedrez y la comprensión de artículos periodísticos, ilustran la diferencia existente entre aquellos problemas en los que mucho conocimiento solo es necesario para acotar la búsqueda y aquellos en los que el conocimiento se emplea para poder reconocer una solución.

- La computadora a la que simplemente se le da el problema ¿puede emitir una solución, o es necesario que ésta interactúe con una persona?

Tipos de problemas:

- **Solitarios:** a la computadora se le da una descripción del problema y ésta proporciona una respuesta sin ningún tipo de comunicación ni necesidad de explicaciones del proceso de razonamiento.
- **Conversacionales:** en los que existe una comunicación intermedia entre el hombre y la computadora, bien para proporcionar una ayuda adicional a la máquina o para que la computadora proporcione información al usuario.

Características de los sistemas de producción

Sistema de producción monótono: es aquel en el que la aplicación de una regla nunca prevé la posterior aplicación de otra regla que podría haberse aplicado cuando se seleccionó la primera.

Sistema de producción no monótono: es aquel en el que lo anterior no es cierto.

Sistema de producción parcialmente conmutativo: es aquel que tiene la propiedad de que si una determinada aplicación de una secuencia de reglas transforma el estado x en el estado y, entonces alguna permutación permitida de estas reglas, también transforma el estado x en el estado y.

Sistema de producción conmutativo: es aquel que es a la vez monótono y parcialmente conmutativo.

Desde un punto de vista formal, no existe relación alguna entre tipos de problemas y tipos de sistemas de producción debido a que todos los problemas pueden resolverse utilizando todos los tipos de sistemas de producción. Pero desde un punto de vista práctico, existe relación.

	Monótono	No monótono
Parcialmente conmutativo	Demostración de teoremas	Navegación de robots
No parcialmente conmutativo	Síntesis química	Bridge

Los sistemas parcialmente conmutativos y los monótonos son adecuados para resolver problemas ignorables.

Son importantes del punto de vista de la implementación porque no contemplan la característica de volver hacia estados pasados cuando se descubre que se ha seguido un camino incorrecto.

Estos sistemas son adecuados para problemas en los que las cosas no cambian, se crean.

Los sistemas no monótonos y parcialmente conmutativos, se adecuan a aquellos problemas en los que se realizan cambios pero éstos son reversibles y en los que el orden de las operaciones no es crítico.

Los sistemas de producción que no son parcialmente conmutativos son adecuados para muchos problemas en los que se producen cambios irreversibles.

El orden en que se realizan las operaciones es importante para determinar el resultado final.

UNIDAD 2: Búsqueda y planificación.

Técnicas de búsqueda a ciegas

El problema se reduce a buscar una ruta a través del espacio que una un estado inicial con un estado objetivo. El proceso de resolución del problema puede modelarse como un sistema de producción y entonces se debe elegir la estructura de control apropiada para el sistema de producción con el fin de que el proceso de búsqueda sea lo más eficiente posible.

Estrategia de control

Con frecuencia es posible aplicar más de una regla en el proceso de búsqueda de la solución de un problema.

Una buena estrategia de control debe cumplir:

Primer requisito: que cause algún cambio.

Segundo requisito: que sea sistemática.

Explosión combinatoria

El problema es que cada nodo añadido al espacio de búsqueda añadirá más de un camino: es decir, el número de caminos hacia el objetivo se incrementará más rápido con cada nuevo nodo.

El número de formas en que N objetos pueden combinarse es igual a $N!$.

Debido a que el número de posibilidades crece rápidamente, sólo los problemas más simples se dirigen a búsquedas exhaustivas. Una búsqueda exhaustiva, teóricamente funcionará siempre, no es práctica porque consume demasiado tiempo, recursos de computadora o ambas cosas.

Ramificación y acotación

Esta técnica comienza generando rutas completas, manteniéndose la ruta más corta encontrada hasta el momento. Deja de explorar una ruta tan pronto como su distancia total, hasta ese momento, sea mayor que la que se ha marcado como la más corta. Esta técnica garantiza hallar la ruta más corta. La cantidad exacta de tiempo utilizado en un problema en particular, depende del orden en que se exploren las rutas. Es inadecuada para problemas grandes.

Búsqueda primero en anchura

Se evalúa cada nodo del mismo nivel antes de proceder al siguiente nivel más profundo. Esto garantiza que encontrará una solución, si existe, porque eventualmente degenerará en una búsqueda exhaustiva.

Algoritmo:

1. Crear una variable llamada LISTA-NODOS y asignarle el estado inicial.
2. Hasta que se encuentre un estado objetivo o LISTA-NODOS esté vacía, hacer:
 - a. Eliminar el primer elemento de LISTA-NODOS y llamarlo E. Si LISTA-NODOS está vacía, terminar.
 - b. Para que cada regla se empareje con el estado descrito en E hacer:
 - i. Aplicar la regla para generar un nuevo estado
 - ii. Si el nuevo estado es un estado objetivo, terminar y devolver este estado
 - iii. En caso contrario, añadir el nuevo estado al final de LISTA-NODOS.

Búsqueda primero en profundidad

Se explora cada camino posible hacia el objetivo hasta su conclusión antes de intentar otro camino. En este tipo de recorrido, va por la izquierda hasta que o bien alcanza un nodo terminal o bien encuentra el objetivo. Si alcanza un nodo terminal, retrocede un nivel, va a la derecha y luego a la izquierda hasta encontrar el objetivo o nodo terminal. Repetirá este procedimiento hasta haber encontrado el objetivo o haber examinado el último nodo.

Algoritmo:

1. Si el estado inicial es un estado objetivo, terminar y devolver el éxito.
2. Caso contrario, hacer lo siguiente hasta que se marque un éxito o un fracaso.
 - a. Generar un sucesor, E, del estado inicial. Si no existen mas sucesores, marcar un fracaso.
 - b. Llamar a la búsqueda en profundidad con E como estado inicial.
 - c. Si se devuelve un éxito, marcar un éxito. En caso contrario, continuar con el ciclo.

Ventajas de la búsqueda primero en profundidad

PRIMERO EN PROFUNDIDAD	PRIMERO EN ANCHURA
Necesita menos memoria ya que solo se almacenan los nodos del camino que se sigue en ese instante.	Almacena todo el árbol que hay sido generado hasta ese momento.
Puede encontrar una solución sin tener que examinar gran parte del espacio de estados	Debe examinar todas las partes del árbol de nivel n antes de comenzar con los nodos de nivel $n+1$

Ventajas de la búsqueda primero en anchura

PRIMERO EN ANCHURA	PRIMERO EN PROFUNDIDAD
No queda atrapada explorando callejones sin salida	Puede seguir una ruta infraestructuosa durante mucho tiempo, y quizás antes de acabar en un estado sin sucesores
Si existe una solución, garantiza que se logre encontrarla. Si existen múltiples soluciones, encuentra la solución mínima	Es posible encontrar una solución larga en alguna parte del árbol, cuando puede existir otra más corta en alguna parte inexplorada del mismo.

Técnicas de búsqueda heurística

Una heurística es una técnica que aumenta la eficiencia de un proceso de búsqueda, posiblemente sacrificando demandas de completitud. Algunas heurísticas ayudan a guiar el proceso de búsqueda sin sacrificar ninguna demanda de completitud que el proceso haya podido tener previamente. Otras pueden ocasionalmente causar que una buena ruta sea pasada por alto. Pero, en promedio mejoran la calidad de las rutas que explotan.

Cuando se aplican a problemas específicos, la eficacia de las técnicas de búsqueda heurísticas depende en gran medida de la forma en que exploten el conocimiento del dominio particular, ya que, por sí solas, no son capaces de salvar la explosión combinatoria. Por esta razón, a estas técnicas se las denomina "métodos débiles".

Problema de la mochila

Consiste en elegir, de entre un conjunto de n elementos de un negocio, (cada uno con un valor v_i y un peso p_i), aquellos que puedan ser cargados en la mochila. Ésta resiste un peso máximo P y se pretende acumular el mayor valor posible.

Una clase de algoritmos heurísticos son los **métodos constructivos** que consisten en ir agregando componentes individuales a la solución hasta que se obtiene una solución factible.

En el problema de la mochila debemos ir escogiendo los elementos que aporten el mayor valor en proporción a su peso (v_i/p_i).

Generación y Prueba**Algoritmo: Generación y Prueba**

1. Generar una posible solución. Para algunos significa generar un objetivo particular en el espacio problema. Para otros, supone generar un camino a partir de un estado inicial.
2. Verificar si realmente el objetivo elegido es una solución comparándolo con el objetivo final.
3. Si se ha encontrado la solución, terminar. Si no, volver al paso 1.

Es un procedimiento primero en profundidad ya que las soluciones completas deben generarse antes de que se comprueben. Es simplemente una **búsqueda exhaustiva por el espacio problema**. Puede funcionar de forma que genere las soluciones en forma aleatoria, pero no garantiza que se pueda encontrar alguna vez la solución, esto se conoce como **algoritmo del Museo británico**. Entre estos dos extremos existe un punto intermedio en donde el proceso de búsqueda actúa de forma sistemática, a pesar de que algunos caminos no se consideren porque dan la impresión de que por ellos no se llega a la solución.

Esta técnica no es muy eficiente por sí misma. Pero cuando se combina con otras técnicas que restrinjan el espacio de búsqueda, puede llegar a ser muy eficaz.

Escalada o Remonte de colinas

Escalada

Es una variante del de generación y prueba. Existe realimentación a partir del procedimiento de prueba que se usa para ayudar al generador a decidirse por cual dirección debe moverse en el espacio de búsqueda. En un procedimiento de generación y prueba puro, la función de prueba responde un sí o un no. Pero si la función de prueba se amplía mediante una función heurística que proporcione una estimación de lo cercano que se encuentra un estado al estado objetivo, el procedimiento de prueba puede usar esta información.

La escalada se utiliza frecuentemente cuando se dispone de una buena función heurística para evaluar los estados, pero cuando no se dispone de otro tipo de conocimiento provechoso.

Escalada simple

Algoritmo: Escalada simple

1. Evaluar el estado inicial. Si es el estado objetivo, devolverlo y terminar. En caso contrario, continuar con el estado inicial como estado actual.
2. Repetir hasta que se encuentre una solución o hasta que no queden nuevos operadores que aplicar al estado actual:
 - a) Seleccionar un operador que no haya sido aplicado con anterioridad al estado actual y aplicarlo para generar un nuevo estado.
 - b) Evaluar el nuevo estado:
 - i. Si es un estado objetivo, devolverlo y terminar.
 - ii. Si no es un estado objetivo, pero es mejor que el estado actual, convertirlo en estado actual.
 - iii. Si no es mejor que el estado actual, continuar con el bucle.

La principal diferencia entre éste y el de generación y prueba consiste en el uso de una función de evaluación.

Escalada por la máxima pendiente

Consiste en considerar todos los posibles movimientos a partir del estado actual y elegir el mejor de ellos.

Algoritmo: Escalada por la máxima pendiente

1. Evaluar el estado inicial. Si es el estado objetivo, devolverlo y terminar. En caso contrario, continuar con el estado inicial como estado actual.
2. Repetir hasta que se encuentre una solución o hasta que una iteración completa no produzca un cambio en el estado actual:
 - a. Sea SUCC un estado tal que algún posible sucesor del estado actual sea mejora que este SUCC.
 - b. Para cada operador aplicado al estado actual hacer lo siguiente:
 - i. Aplicar el operador y generar un nuevo estado.
 - ii. Evaluar el nuevo estado. Si es un estado objetivo, devolverlo y terminar. Si no, compararlo con SUCC. Si es mejor, asignar a SUCC este nuevo estado. Si no es mejor, dejar SUCC como está.
 - c. Si SUCC es mejor que el estado actual, hacer que el estado actual sea SUCC.

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

- **Máximo local:** es un estado que es mejor que todos sus vecinos, pero que no es mejor que otros estados de otros lugares.
- **Meseta:** es un área en la que un conjunto de estados vecinos posee el mismo valor.
- **Cresta:** es un tipo especial de máximo local. Es un área más alta que las áreas circundantes y que además posee en ella misma una inclinación.

Existen algunas formas de evitar estos problemas, si bien estos métodos no dan garantías:

- Volver atrás hacia algún modo anterior e intentar seguir un camino diferente. Este método es adecuado para superar máximos locales.
- Realizar un gran salto en alguna dirección. Indicado para superar mesetas.
- Aplicar dos o más reglas antes de realizar la evaluación. Indicado para superar crestas.

Búsqueda El primero mejor

Se selecciona el nodo más prometedor que se haya generado hasta ese momento. A continuación se expande el nodo elegido aplicando las reglas para generar a sus sucesores. Si alguno de ellos es una solución, el proceso termina. Si no es así, estos nuevos nodos se añaden a la lista de nodos que se han generado hasta ese momento. De nuevo, se selecciona el más prometedor y el proceso continúa de la misma forma.

En esta búsqueda se selecciona el mejor estado disponible, aún si este estado tiene un valor menor que el del que se estaba explorando.

Se puede utilizar un grafo para ilustrar el proceso de búsqueda. Cada nodo representa un punto en el espacio de estado y contiene una descripción de lo que representa en el espacio de estados, una indicación de lo prometedor que es, un enlace paterno que apunta al mejor nodo desde el que se ha generado y una lista de los nodos que se generan a partir de él. El enlace paterno nos posibilita restablecer el camino hacia el objetivo y la lista de sucesores hará posible propagar la mejora a sus sucesores. A los grafos de este tipo se los denomina **grafos O**.

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

- ABIERTOS: nodos que se han generado y a los que se les ha aplicado la función heurística, pero que aún no han sido examinados.
- CERRADOS: nodos que ya se han examinado. Es necesario mantener estos nodos en memoria si se desea hacer una búsqueda sobre un grafo y no sobre un árbol, debido a que cuando se genera un nuevo nodo, se debe verificar si ese nodo se había generado con anterioridad.

También se necesita una función heurística que vaya haciendo una estimación de los méritos de cada nodo.

f: función que es la suma de dos componentes g y h. Estimación del coste necesario para alcanzar un estado objetivo por el camino que se ha seguido para generar el nodo actual.

g: es una medida del coste para ir desde el estado inicial hasta el nodo actual.

h: es una estimación del coste adicional para alcanzar un nodo objetivo a partir del nodo actual.

Algoritmo: Búsqueda El primero mejor

1. Comenzar con ABIERTOS conteniendo solo el estado inicial.
2. Hasta que se llegue a un objetivo o no queden nodos en ABIERTOS hacer:
 - a. Tomar el mejor nodo de ABIERTOS.
 - b. Generar sus sucesores.
 - c. Para cada sucesor hacer:
 - i. Si no se ha generado con anterioridad, evaluarlo, añadirlo a ABIERTOS y almacenar a su padre.
 - ii. Si ya se ha generado antes, cambiar al padre si el nuevo camino es mejor que el anterior. En este caso, se actualiza el coste empleado para alcanzar el nodo y a los sucesores que pudiera tener.

Transformación en el espacio de configuraciones.

La planificación de trayectoria de robot ilustra la búsqueda.

Para evitar obstáculos, la representación original implica un objeto en movimiento y obstáculos estacionarios, y la nueva representación implica un punto en movimiento y obstáculos virtuales más grandes llamados **obstáculos del espacio de configuraciones**.

Se desliza el objeto alrededor del obstáculo, manteniendo el contacto entre ellos todo el tiempo, y llevando el registro de un punto de rastreo arbitrario en el objeto en movimiento conforme avanza.

No queda del todo claro si la trayectoria más corta es a través de la abertura. Para asegurarse que de veras lo es, usted tiene que efectuar una búsqueda.

Hasta ahora no existe una red en donde buscar. Desde donde usted está, puede ver la posición deseada o bien no verla. Si puede, entonces no requerirá hacer más, ya que la trayectoria más corta es la línea recta entre usted y la posición deseada.

Si no puede ver la posición deseada desde donde está, entonces el único movimiento que tiene sentido es hacia uno de los vértices que puede haber. Todo movimiento queda restringido de vértice a vértice, excepto al principio, y al final. Por lo tanto, la posición inicial, la deseada y los vértices son como los nodos de una red. La red se conoce como **grafo de visibilidad**.

Reducción de problemas.

Es posible convertir metas difíciles en una o más submetas más fáciles de lograr. Cada submeta, puede dividirse en una o más submetas de nivel inferior.

Cuando un procedimiento se utiliza a sí mismo, se dice que recurre. Los sistemas en los que los procedimientos se usan a sí mismos se conocen como *recursivos*.

La idea clave del método de Reducción de Problemas es explorar un árbol de metas.

Un árbol de metas es un árbol sistemático en el que los nodos representan metas y las ramas indican la forma en que usted puede lograr metas, mediante la solución de una o más submetas.

Las metas que se satisfacen solo cuando todas sus submetas inmediatas quedan satisfechas se conocen como **metas Y**. Se les señala colocando arcos en sus ramas.

Las **Metas O** se satisfacen cuando cualesquiera de sus submetas inmediatas quedan satisfechas. Permanecen sin señalar.

Las metas que se satisfacen directamente son **metas hojas**.

A estos árboles se los conoce como **árboles Y-O**.

El árbol de metas permite responder a preguntas ¿Cómo? y ¿Por qué?

Para tratar con preguntas de "**¿cómo?**" identifique en el árbol Y-O la meta implicada, si la meta es una Y, da a conocer todas las metas inmediatas. Si la meta es una O, menciona la submeta inmediata que se logró. Para preguntas de "**¿Por qué?**" identifique la meta y notifique la supermeta inmediata.

Verificación de restricciones

El objetivo de los problemas de verificación de restricciones es **descubrir algún estado del problema que satisfaga un conjunto dado de restricciones**.

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

El primer paso, la **propagación** se hace necesario por el hecho de que normalmente existen dependencias entre las restricciones. También surge debido a la presencia de reglas de inferencia que permiten que se infieran restricciones adicionales a partir de las que se tenían.

La propagación de restricciones termina por una razón de entre dos posibles:

- La primera, porque se detecte una contradicción. Si esto ocurre, entonces no existe una solución.
- La segunda, es que la propagación se realice de tal forma que no puedan hacerse más cambios basándose en el conocimiento actual.

Segundo paso. Para ver la forma de fortalecer las restricciones, pueden realizarse algunas **hipótesis**. Una vez que se ha hecho debe comenzar de nuevo la propagación de las restricciones a partir de este nuevo estado. Si se encuentra una solución, se muestra.

Si aún se necesitan más restricciones, se hacen. Si se detecta alguna contradicción, puede usarse una *vuelta atrás* para intentarlo con una suposición diferente y comenzar con ella.

Existen dos tipos de restricciones, las primeras son sencillas: *son una lista de posibles valores para un objeto*. Las segundas: *describen relaciones entre o en medio de objetos*.

Análisis de medios y fines

El proceso de análisis de medios y fines se centra en la *detección de diferencias entre el estado actual y el estado objetivo*.

Una vez que se ha aislado una diferencia, debe encontrarse un operador que pueda reducirla. Es posible que tal operador no pueda aplicarse en el estado actual por lo tanto, se crea el subproblema que consiste en alcanzar un estado en que pueda aplicarse dicho operador. Este es un tipo de encadenamiento hacia atrás, recibe el nombre de *realización de subobjetivos para un operador*.

Es posible que el operador no produzca el estado objetivo. En este caso, se tiene un segundo subproblema que consiste en llegar desde ese estado hasta un objetivo. Estos dos subproblemas serán más fáciles de resolver que el problema original.

El análisis de medio y fines cuenta con un conjunto de reglas que pueden transformar un estado problema en otro. Estas reglas se representan con un **lado izquierdo** que describe las condiciones que deben cumplirse para que pueda aplicarse la regla (precondiciones) y un **lado derecho** que describe aquellos aspectos del estado problema que cambiarán al aplicar la regla. Existe una estructura de datos separada denominada **tabla de diferencias** que ordena las reglas atendiendo a las diferencias que pueden reducir.

Búsqueda en problemas de juego

Juegos de dos jugadores

Una forma de contemplar todos los procedimientos de búsqueda que se han explicado es que esencialmente se trata de **procedimientos de generación y prueba**. En un extremo, el generador proporciona propuestas de soluciones complejas, que el comprobador evalúa. En el otro, el generador genera movimientos individuales en el espacio de búsqueda, cada uno de los cuales se evalúa a continuación mediante el comprobador para pasar a elegir el más prometedor de ellos. Para mejorar la efectividad de un programa resolutor de problemas es necesario hacer dos cosas:

- **Mejorar el procedimiento de generación** de forma que solo se generen movimientos buenos.
- **Mejorar el procedimiento de prueba** para que solo se reconozcan y exploren en primer lugar los mejores movimientos.

En los programas para jugar es particularmente importante que se hagan las dos cosas.

Si se usa un generador de movimientos legales, el procedimiento de prueba deberá procesar cada uno de ellos. Como el procedimiento de búsqueda debe tener en cuenta muchas posibilidades, debe ser rápido. Probablemente no pueda realizar su trabajo con precisión.

Si se usa un generador de movimientos plausibles en el que solo se genera un número pequeño de movimientos prometedores.

Con un generador de movimientos más selectivo, el procedimiento de prueba puede permitir emplear más tiempo en la evaluación de cada uno de los movimientos. Con la incorporación de conocimiento heurístico tanto en el generador como en el comprobador, se mejora el rendimiento del sistema total.

El procedimiento minimax

Es un procedimiento de búsqueda en profundidad limitada.

La idea consiste en comenzar en la posición actual y usar el generador de movimientos plausibles para generar un conjunto de posiciones sucesivas posibles. Se puede aplicar la función de evaluación estática a esas posiciones y escoger la mejor. Después puede llevarse hacia atrás ese valor hasta la posición de partida. Esta es tan buena como la posición generada por el mejor movimiento que podamos hacer a continuación.

El procedimiento alfa-beta

Una de las ventajas de los procedimientos primero en profundidad es que a menudo puede mejorarse su eficiencia usando técnicas de ramificación y acotación, en las que pueden abandonarse rápidamente aquellas soluciones parciales que son claramente peores que otras soluciones conocidas.

Esta estrategia requiere de dos valores umbrales, una **cota inferior del valor** que puede asignarse en último término a un **nodo maximizante** (alfa) y una **cota superior** que puede asignarse a un **nodo minimizante** (beta).

La búsqueda en un nivel minimizante puede terminar cuando se descubre un nivel menor que ALFA mientras que en un nivel maximizante la búsqueda puede terminar al descubrir un valor mayor que BETA.

Búsqueda con sistemas evolutivos

Algoritmos genéticos (AG)

Son métodos adaptativos que pueden ser utilizados para implementar búsquedas y problemas de optimización. Basados en los procesos genéticos de organismos biológicos, codificando una posible solución a un problema en un "cromosoma" compuestos por una cadena de bits o caracteres.

Estas cromosomas representan individuos que son llevados a lo largo de varias **generaciones**, evolucionando de acuerdo a los principios de **selección natural y supervivencia** del más apto. Emulando estos procesos, los algoritmos genéticos son capaces de "evolucionar" soluciones a problemas del mundo real.

Los AG trabajan con una **población de individuos**, cada uno representando una posible solución a un problema dado. A cada individuo se le asigna una **puntuación de adaptación**, dependiendo de que tan buena fue la respuesta al problema. A los más adaptados se les da la oportunidad de reproducirse, produciendo descendientes con características de ambos padres, los miembros menos adaptados desaparecen.

Una nueva población de posibles soluciones es generada mediante la **selección** de los mejores individuos de la generación actual. Esta nueva generación contiene una proporción más alta de las características poseídas por los mejores miembros de la generación anterior. A lo largo de varias generaciones, las características buenas son difundidas a lo largo de la población. Favoreciendo el emparejamiento de los individuos mejor adaptados.

Los dos procesos que más contribuyen a la evolución son el **crossover** y la adaptación basada en la **selección**. La **mutación** no debe ser utilizada demasiado, ya que el AG se puede convertir en una búsqueda al azar, pero su utilización asegura que ningún punto en el espacio de búsqueda tiene probabilidad 0 de ser examinado.

Los AG son implementados siguiendo el siguiente ciclo:

1. Generar aleatoriamente la población inicial
2. Evaluar la adaptación de todos los individuos en la población
3. Crear una nueva población efectuando operaciones como **crossover**, reproducción proporcional a la adaptación y mutaciones en los individuos.
4. Eliminar la antigua población
5. Iterar utilizando la nueva población, hasta que la población converja. Cada iteración es conocida como **generación**.

Los AG no garantizan que encontrarán la solución óptima, pero son buenos encontrando soluciones aceptables en corto tiempo.

Diferencias entre los AG y los métodos tradicionales

- Trabajan con una codificación del conjunto de parámetros, no con estos directamente.
- Buscan simultáneamente la solución en una población de puntos, no en uno sólo.
- Utilizan la función objetivo.
- Utilizan reglas de transición probabilísticas.

Poblaciones

Las partes que relacionan un AG con un problema dado son la codificación y la función de evaluación.

Un problema puede ser representado por un **conjunto de parámetros** (genes), estos pueden ser unidos para formar una **cadena de valores** (cromosoma), a este proceso se le llama **codificación**. En genética este conjunto representado por un cromosoma en particular es referido como **genotipo**, este contiene la información necesaria para construir un organismo, **fenotipo**. El conjunto de parámetros especificando el diseño es el genotipo y la construcción final es el fenotipo.

Aspectos relacionados con la codificación de un problema:

- Se debe utilizar el alfabeto más pequeño posible para representar los parámetros.
- Las variables que representan los parámetros del problema deben ser discretizadas para poder representarse con cadenas de bits.
- Existen relaciones ocultas entre las variables que conforman la solución, esta relación es referida como **epístasis** y es necesario tomarla en cuenta para una representación adecuada del problema.

Operadores genéticos

Son las distintas funciones que se aplican a las poblaciones, los tipos de operadores más usados son: *Selección, Crossover, Mutación, Migración y Otros operadores.*

Selección

Determina como los individuos son elegidos para el apareamiento. Los más comunes son: método de la ruleta, torneos o ranking.

Crossover (entrecruzamiento)

Consiste en el intercambio de material genético entre dos cromosomas. Es el principal operador genético.

Para aplicarlo se escogen aleatoriamente dos miembros de la población.

La forma básica de este operador toma dos individuos y corta sus cromosomas en una posición seleccionada al azar, para producir dos segmentos anteriores y dos posteriores, los posteriores se intercambian para obtener dos cromosomas nuevos.

Mutación

Es la alteración en forma aleatoria de un individuo de la población. Esto es necesario para que no se produzca una convergencia prematura y que todos los individuos de la población no tengan probabilidad cero de ser utilizados.

Se aplica para cambiar aleatoriamente a individuos parte de su material genético e introducir diversidad a la población.

Es un mecanismo generador de diversidad, pero también reduce el algoritmo a una búsqueda aleatoria.

Migración

Genera un intercambio de individuos entre subpoblaciones.

Al recibir individuos de otras subpoblaciones se introduce la diversidad y se incrementa la presión de la selección en cada subpoblación.

Evita convergencias prematuras a óptimos locales. Si la subpoblación ha llegado a un estado de equilibrio puede no ser efectiva.

Función de evaluación

Dado un cromosoma, consiste en asignarle un valor numérico de **adaptación** proporcional a la utilidad del individuo representado. El desarrollo de la función involucra hacer una simulación, en otros puede estar basada en el rendimiento y representar sólo una evaluación parcial del problema.

Debe ser rápida ya que hay que aplicarla a cada individuo, gran parte del tiempo de un AG se emplea en la función de evaluación.

Convergencia

Es la progresión hacia la uniformidad. Un gen ha convergido cuando el 95 % de la población tiene el mismo valor. La población converge cuando todos los genes de cada individuo lo hacen.

Convergencia prematura

Una problema de los AG dado por una mala formulación del modelo es aquel en el cual los *genes de unos pocos individuos* relativamente bien adaptados, pero no óptimos, pueden rápidamente *dominar la población*.

Una vez que esto ocurre, la habilidad del modelo para buscar mejores soluciones es eliminada completamente, quedando sólo la mutación como vía de buscar nuevas alternativas y el algoritmo se convierte en una búsqueda lenta al azar.

Para evitar este problema, es necesario controlar el número de oportunidades reproductivas de cada individuo, tal que, no obtenga ni muy alta ni muy baja probabilidad.

Finalización lenta

Contrario al anterior, luego de muchas generaciones, la población habrá convergido, pero no habrá localizado el máximo local. Habrá poca diferencia entre el mejor y el individuo promedio.

Algoritmo genético canónico

El primer paso de un AG, establecida la función objetivo, es definir el tipo de la tira que conformará el cromosoma.

En los AGC, las mismas son **tiras binarias** de longitud L. Cada gen es un dígito 0 o 1.

Conocido N (nro de cromosomas), se procede a crear la población inicial. Se realiza en forma random.

A continuación, cada cromosoma es evaluado, usando la **función objetivo** y la **función fitness**. La **función fitness** se define para cada cromosoma, como el cociente fi/F , donde fi es el valor de la **función objetivo**, mientras que F es el **promedio de los valores de la función objetivo** para cada tira de la población.

Planificación

Componentes de un sistema de planificación

En los sistemas de resolución de problemas, era necesario llevar a cabo las siguientes funciones:

- Elegir la mejor regla para aplicar:

Consiste en aislar el conjunto de diferencias existentes entre el objetivo deseado y el estado actual para identificar aquellas reglas que pueden reducir estas diferencias. Si se encuentran varias puede usarse otro tipo de información heurística para poder elegir entre ellas. Esta técnica se basa en el método de análisis de medios y fines.

- Aplicar la regla elegida:

Debemos ser capaces de trabajar con reglas que solo especifican una parte pequeña de un estado completo del problema. Existen muchas formas de lograrlo.

Una forma consiste en describir para cada acción los cambios que realiza en la descripción del estado. También son necesarias sentencias que hagan que lo demás permanezca inalterado.

Un estado se describe por un conjunto de predicados que representan los hechos que son ciertos en ese estado.

Necesitamos un conjunto de reglas denominadas **axiomas marco** que describen los componentes del estado que no se ven afectados por cada operador.

La ventaja de este enfoque es que se pueden llevar a cabo todas las operaciones necesarias en la descripción de los estados con un único y sencillo mecanismo como es el de resolución. Sin embargo el precio que se paga es que el número de axiomas necesarios puede hacerse muy grande si las descripciones de los estados del problema son complejas. Para poder manipular dominios de problemas complejos, es necesario disponer de un mecanismo que no involucre un conjunto de axiomas marco demasiado grande.

- Detectar una solución:

Un sistema de planificación ha tenido éxito al encontrar una solución a un problema cuando encuentra una secuencia de operadores que transforman el estado inicial del problema en un estado objetivo.

¿Cómo saber cuando ocurre el éxito? En los sistemas sencillos, comparándolo con las descripciones del estado. Pero si no están representados explícitamente los estados completos, entonces el problema se hace más complejo. El modo de resolverlo depende de cómo estén representadas las descripciones de los estados.

* Si podemos construir esta demostración, entonces el proceso de resolución del problema termina.

* Si no se puede, debe proponerse una secuencia de operadores que podría resolverlo.

- Detectar callejones sin salida:

Un sistema debe ser capaz de detectar si está explorando un camino que nunca puede conducir a una solución o no es probable que lo haga. Pueden emplearse los mismos mecanismos de razonamiento que se usaron para detectar una solución:

- * Si el proceso de búsqueda razona hacia delante a partir del estado inicial, puede podar los caminos que conduzcan a un estado a partir del cual no se alcanza un estado objetivo.
- * Si el proceso de búsqueda es hacia atrás a partir de un estado solución, se puede dar por finalizado un camino si se está seguro de que el estado inicial no puede alcanzarse o porque solo pueden hacerse pequeños progresos. En el razonamiento hacia atrás, cada objetivo se descompone en subobjetivos. Cada uno puede descomponerse en subobjetivos adicionales.

- **Identificar soluciones casi correctas:**

Para resolver problemas casi descomponibles una buena forma consiste en asumir que son completamente descomponibles, resolver los subproblemas por separado y verificar si al combinar las subsoluciones tenemos una solución. Si ocurre lo anterior no debe hacerse nada más.

Si no ocurre, se puede hacer:

- * Desechar la solución, buscar otra y esperar que resulte mejor que la anterior. Aunque es sencilla puede conducir a malgastar una gran cantidad de esfuerzo.
- * Comparar la solución que resulta al ejecutarse la secuencia de operaciones correspondientes a la solución propuesta con el objetivo deseado. En la mayoría de los casos las diferencias entre las dos serán menores que las diferencias entre el estado inicial y el objetivo. Puede llamarse de nuevo al sistema de resolución de problemas y pedirle que encuentre una forma de eliminar estas nuevas diferencias.

La primera solución se puede combinar con la segunda para formar una solución para el problema original.

* Otra solución consiste en dejarlas especificadas de forma incompleta hasta que sea posible. Entonces, cuando esté disponible tanta información como sea posible, se completa la especificación de forma que no surjan conflictos. Esto se denomina *estrategia de mínimo compromiso*.

Planificación mediante una pila de objetivos

Se usa una pila que contiene tantos objetivos como operadores que deben proponerse para satisfacer estos objetivos.

El resolutor de problemas también usa una base de datos que describe la situación actual y un conjunto de operadores descritos mediante las listas PRECONDICION, AÑADIR y BORRAR.

1. En cada paso del proceso de resolución del problema se seguirá la pista del objetivo situado en la cima de la pila. Cuando se encuentra una secuencia de operadores que lo satisfacen, esta secuencia se aplica a la descripción del estado, obteniendo una representación.
2. Se explora el objetivo situado en la cima de la pila y se intenta hacer que se satisfaga, comenzando a partir de la situación que se produjo como resultado de satisfacer el primer objetivo.
3. Este proceso continúa hasta que la pila de objetivos esté vacía.
4. Como última verificación, el objetivo original se compara con el estado final.
5. Si en este estado no se satisfacen algunas partes del objetivo, entonces las partes no resueltas del subobjetivo se reinseran en la pila y se repite el proceso.

Planificación no lineal mediante fijación de restricciones

Los problemas difíciles provocan interacciones entre los objetivos. Los operadores que se utilizan para resolver un subproblema pueden interferir en la solución de un subproblema anterior.

La mayoría de los problemas necesitan un plan entrelazado en el que se trabaje simultáneamente con múltiples subproblemas. Este tipo de plan se denomina **plan no lineal** ya que no está compuesto por una secuencia lineal de subplanes completos.

La idea de **fijación de restricciones** es construir un plan mediante operadores incrementalmente hipotéticos, ordenamientos parciales entre operadores y enlaces entre variables y operadores.

Una solución es un conjunto de operadores parcialmente ordenados e instanciados para generar un plan intermedio, se convierte el orden parcial en un número de órdenes totales.

Planificación jerárquica

Para resolver problemas complicados, es importante poder eliminar algunos de los detalles del problema hasta que se encuentre una solución.

Enfoque para resolver un problema:

- Resuelve el problema considerando solo aquellas precondiciones cuyo valor crítico sea el más alto posible. Para lograrlo, ignora las precondiciones que caen por debajo de un cierto nivel crítico.
- Utiliza el plan construido y considera las precondiciones del siguiente nivel de criticidad más bajo.
- Aumenta el plan con los operadores que satisfacen estas precondiciones. Al elegir los operadores, ignora todas aquellas precondiciones cuya criticidad sea menor que el nivel que ahora se está considerando.

Este proceso se denomina *búsqueda primero en longitud*. Explora planes completos a un nivel de detalle antes de mirar los detalles de más bajo nivel.

La *asignación de valores de criticidad apropiados* es un aspecto crucial. Aquellas precondiciones que no tengan operadores que puedan satisfacerlas son las más críticas.

Para que un sistema de planificación jerárquica funcione con reglas, debe darse junto con las propias reglas, el valor de criticidad para cada término que pueda aparecer en la precondición. No se malgastarán esfuerzos en eliminar los detalles de planes que no estén cercanos a la resolución del problema.

Sistemas reactivos

Consiste en *evitar planificar totalmente y utilizar la situación observable como pista a la que simplemente reaccionar*.

Un sistema reactivo debe tener acceso a una base de conocimiento que describa las acciones que deben realizarse bajo ciertas circunstancias. Un sistema reactivo elige solo una acción cada vez; no anticipa y selecciona una secuencia completa de acciones antes de realizar una primera acción.

- **Ventajas** frente a los planificadores tradicionales: es que funcionan de forma robusta en dominios difíciles de modelar con exactitud de forma completa.
- Evitan un modelado completo y basan sus acciones directamente en sus percepciones del mundo.
- Son extremadamente sensibles, ya que evitan la explosión combinatoria. Son muy atractivos para tratar con tareas en tiempo real.
- Como no mantienen modelo del mundo ni estructuras explícitas del objetivo, su rendimiento es limitado en este tipo de tareas.

Unidad 3: Representación del Conocimiento y Razonamiento

El problema de la Representación del Conocimiento

Correspondencia entre Conocimiento y RC

En los programas de IA una característica que está presente en todas las representaciones, el hecho que tiene dos tipos de entidades:

- **Hechos:** es aquello que queremos representar
- **Representaciones de los hechos en un determinado formalismo:** son las entidades que realmente seremos capaces de manipular.

Una posible estructuración consiste en clasificar estas entidades en dos niveles:

- **El nivel del conocimiento:** donde se describen los hechos
- **El nivel simbólico:** donde se describen los objetos del nivel del conocimiento en términos de símbolos manipulables por programas.

La relación que se establece entre los hechos reales y la representación de los mismos se denomina **correspondencia de la representación**.

- Representación hacia delante: establece una correspondencia entre los hechos y sus representaciones
- Representación hacia atrás: establece una correspondencia desde las representaciones a los hechos.

Propiedades de un buen sistema de RC:

- **Suficiencia de la representación:** Capacidad de representar todos los tipos de conocimiento necesarios en el dominio.
- **Suficiencia deductiva:** Capacidad para manipular las estructuras de la representación con el fin de obtener nuevas estructuras que se correspondan con un nuevo conocimiento deductivo a partir del antiguo.
- **Eficiencia deductiva:** Capacidad de incorporar información adicional en las estructuras de conocimiento.
- **Eficiencia en la adquisición:** Capacidad de adquirir nueva información con facilidad. Existen múltiples técnicas para la representación del conocimiento.

Modelos de Representación del Conocimiento

Espectro sintáctico-semántico de las representaciones

Sistemas puramente sintácticos: No se tiene en cuenta el significado del conocimiento que está siendo representado. Poseen reglas simples y uniformes para manipular la representación.

Sistemas puramente semánticos: Cada aspecto de la representación corresponde a una parte de información diferente y las reglas de inferencia son complicadas.

Estructuras declarativas diferentes para representar el conocimiento:

- | | | |
|--|---|------------|
| • Lógica de predicados | } | Sintáctico |
| • Reglas de producción | | |
| • Sistemas no monótonos | | |
| • Sistemas de razonamiento estadístico | | |
| • Redes semánticas | } | Semántico |
| • Marcos | | |
| • Dependencia conceptual | | |
| • Guiones CYC | | |

Conocimiento relacional simple

El modo más sencillo de representar los hechos declarativos es mediante un conjunto de relaciones del mismo tipo que las utilizadas en los sistemas de bases de datos. Esta representación es **simple** debido a la escasa capacidad deductiva que ofrece.

Los sistemas de bases de datos están diseñados para proporcionar el soporte adecuado al conocimiento relacional.

Conocimiento heredable

Es posible extender la representación básica con unos mecanismos de inferencia que operen sobre la estructura de la representación. Una de las fórmulas más útiles de inferencia es la **herencia de propiedades**, donde los elementos de una clase heredan los atributos y los valores de otras clases más generales en los que están incluidos.

Los objetos se deben organizar en **clases** y las clases se deben disponer como una **jerarquía de generalizaciones**. Las líneas representan atributos, los **nodos recuadrados** representan objetos y valores de los atributos de los objetos. Estos valores, se pueden ver como objetos con atributos y valores. Las **flechas** conectan los objetos con sus valores a través de los correspondientes atributos. La figura 3.2 se denomina **estructura de ranura y relleno** o red semántica.

Las dos excepciones que se utilizan son el atributo **es-un** que indica que una clase está contenida en otra y el atributo **instancia** que se usa para indicar pertenencia a una clase.

Estos dos atributos son la base de la herencia de propiedades como una técnica de inferencia. Mediante esta técnica se puede acceder a la base de conocimiento y recuperar los hechos que se han almacenado en ella, así como otros hechos que se pueden deducir.

Conocimiento deductivo

En algunas ocasiones es necesario echar mano de toda la potencia de la lógica tradicional y aún más para describir las inferencias apropiadas.

Necesitaremos un mecanismo de inferencia que pueda aprovechar el conocimiento.
El **procedimiento de inferencia** implementa las **reglas lógicas de la inferencia**.

Conocimiento procedimental

Especifica qué hacer cuando se da una determinada situación. Se puede representar de maneras distintas en los programas, la más habitual consiste en especificarlos como un código que hace algo. La máquina utiliza el conocimiento cuando ejecuta el código para llevar a cabo una determinada tarea.

Problemas de la Representación del Conocimiento

A. Atributos importantes

Los atributos **instancia** y **es-un** son importantes debido a que en ellos se apoya la herencia de propiedades. Dichos atributos representan la pertenencia a una clase y la inclusión de una clase en otra, y que la inclusión de clases es transitiva.

B. Relaciones entre atributos

Los atributos para describir a los objetos pueden ser a su vez entidades representables. Propiedades:

- **Inversos:** se usan atributos que fijen un determinado punto de vista, pero utilizándolos por parejas, de manera que **uno sea el inverso del otro**.
- **Existencia en una jerarquía es-un:** Se puede hablar de atributos y de especializaciones de los atributos. Este tipo de relaciones de **generalización-especialización**, sirven de soporte a la herencia.
- **Técnicas para el razonamiento acerca de los valores**
El sistema debe razonar sobre valores que no ha recibido explícitamente:
 - * Información acerca del tipo de valor.
 - * Restricciones sobre el valor.
 - * Reglas para el cómputo de un valor cuando sea necesario.
 - * Reglas que describen las acciones que se deberían llevar a cabo en el caso de que se llegase a conocer un determinado valor.
- **Atributos univaluados:** es un atributo que solo puede tomar un valor. Cuando se pretenda definir un nuevo valor para uno de estos atributos puede ser por dos razones: porque se ha producido un cambio en el mundo o existe una contradicción en la base de conocimiento que es necesario resolver.

C. Selección de la granularidad de la representación.

Mientras más bajo sea el nivel que escojamos, más sencillo será razonar para ciertos casos, a costa de un proceso de inferencia más complejo y de un mayor espacio de almacenamiento.

D. La representación de conjuntos de objetos.

Es importante por:

- Existen propiedades que se verifican en conjuntos de objetos pero no así en los elementos particulares de los conjuntos.
- Cuando existe una propiedad que verifican todos los elementos del conjunto, es más eficiente asociar esta propiedad al conjunto que a cada uno de sus componentes.
Hay dos formas de definir un conjunto y sus elementos:
 - Enumerar todos sus elementos (definición por extensión)
 - Dar una determinada regla (definición por comprensión)

Es más fácil determinar cuando son iguales dos conjuntos definidos por extensión, que cuando son definidos por comprensión.

Las representaciones por comprensión permiten definir conjuntos infinitos y conjuntos en los que no se conocen todos sus elementos.

Las representaciones por comprensión se pueden definir dependientes de parámetros modificables.

E. Búsqueda de la estructura adecuada a cada circunstancia.

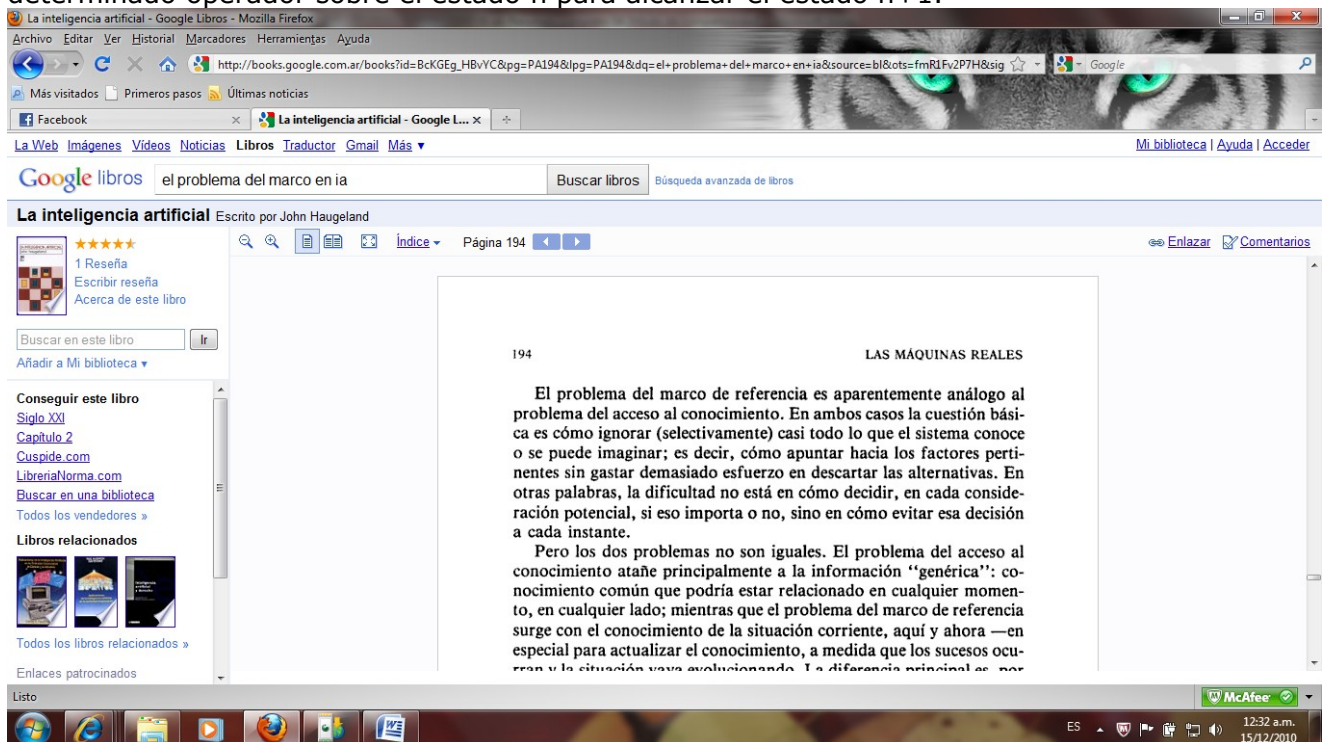
Tener en cuenta los siguientes puntos:

- Tomar aquellas partes de la estructura seleccionada que se hayan conseguido identificar en la presente situación y utilizarlas en la búsqueda de las posibles alternativas.
- Obviar el fallo de la estructura actual y seguir utilizándola.
- Utilice enlaces que conecten las estructuras y que sugieran posibles direcciones de búsqueda.
- Si las estructuras de conocimiento están almacenadas como una jerarquía es-un, ascender por la jerarquía hasta encontrar una estructura lo bastante general como para no entrar en contradicción con la situación en curso.

El problema del Marco

Es el problema de la representación de los hechos que cambian, así como de aquellos que no lo hacen.

Se utilizan **axiomas del marco**, que describen las cosas que cambian cuando se aplica un determinado operador sobre el estado n para alcanzar el estado $n+1$.



La inteligencia artificial - Google Libros - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

http://books.google.com.ar/books?id=BcKGEG_HBvYC&pg=PA194&lpg=PA194&dq=el+problema+del+marco+en+ia&source=bl&ots=fmR1Fv2P7H&sig=...

Más visitados Primeros pasos Últimas noticias

Facebook La inteligencia artificial - Google Libros

La Web Imágenes Videos Noticias Libros Traductor Gmail Más

Google libros el problema del marco en ia Buscar libros Búsqueda avanzada de libros

La inteligencia artificial Escrito por John Haugeland

★★★★★ 1 Reseña Escribir reseña Acerca de este libro

Buscar en este libro Ir

Añadir a Mi biblioteca

Conseguir este libro

Siglo XXI

Capítulo 2

Cuspid.com

Librería Norma.com

Buscar en una biblioteca

Todos los vendedores

Libros relacionados

Todos los libros relacionados

Enlaces patrocinados

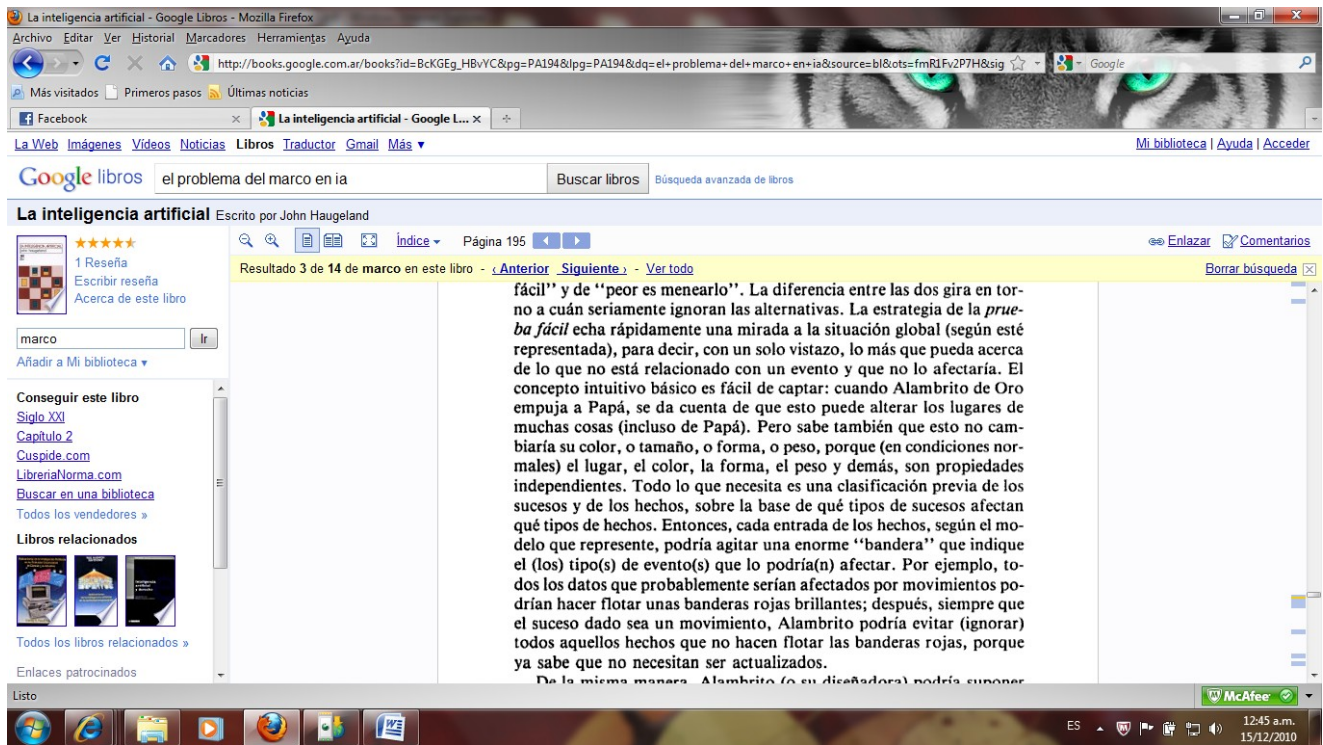
194 LAS MÁQUINAS REALES

El problema del marco de referencia es aparentemente análogo al problema del acceso al conocimiento. En ambos casos la cuestión básica es cómo ignorar (selectivamente) casi todo lo que el sistema conoce o se puede imaginar; es decir, cómo apuntar hacia los factores pertinentes sin gastar demasiado esfuerzo en descartar las alternativas. En otras palabras, la dificultad no está en cómo decidir, en cada consideración potencial, si eso importa o no, sino en cómo evitar esa decisión a cada instante.

Pero los dos problemas no son iguales. El problema del acceso al conocimiento atañe principalmente a la información "genérica": conocimiento común que podría estar relacionado en cualquier momento, en cualquier lado; mientras que el problema del marco de referencia surge con el conocimiento de la situación corriente, aquí y ahora —en especial para actualizar el conocimiento, a medida que los sucesos ocurren y la situación va evolucionando. La diferencia principal es...

McAfee

ES 12:32 a.m. 15/12/2010



Lógica simbólica

La lógica y el lenguaje

El estudio de la lógica, es el estudio de los métodos y principios usados al distinguir entre los argumentos correctos y los argumentos incorrectos.

El estudio de la lógica:

- La práctica ayudará a alcanzar la perfección.
- Incrementará la capacidad de razonamiento.
- Dará técnicas para probar la validez de todos los argumentos.

La lógica es la ciencia del razonamiento. El razonamiento es la clase especial de pensamiento llamada **inferencia**, en la que se sacan conclusiones partiendo de premisas.

El lógico no se interesa en el proceso real del razonamiento, le importa la corrección del proceso completado.

Si las premisas son un fundamento adecuado para aceptar la conclusión, si afirmar que las premisas son verdaderas garantiza el afirmar la verdad de la conclusión, entonces el razonamiento es correcto. De otra forma es incorrecto.

Naturaleza del argumento

Las proposiciones son o verdaderas o falsas y en esto difieren de las preguntas, órdenes y exclamaciones.

Una oración declarativa es siempre parte de un lenguaje, las proposiciones no son privativas de ninguna de las lenguas en que se expresa.

La misma oración articulada en diferentes contextos puede afirmar diferentes proposiciones.

A cada inferencia corresponde un argumento, y de estos argumentos trata la lógica primordialmente.

Argumento: grupo cualquiera de proposiciones o enunciados de los cuales se afirma que hay uno que se sigue de los demás, considerando éstos como fundamento de la verdad de aquel.

Todo argumento tiene una estructura, **premisa y conclusión**.

Conclusión: es la proposición afirmada basándose en las otras proposiciones del argumento y estas otras proposiciones que se afirman como fundamento o razones para la aceptación de la conclusión son las **premisas** de ese argumento.

Toda proposición puede ser premisa o conclusión dependiendo del contexto.

Es una premisa cuando se presenta en un argumento en el que se le supone para demostrar alguna otra proposición y es una conclusión cuando se presenta en un argumento que se pretende la demuestra basándose en las otras proposiciones que se suponen.

Los argumentos pueden ser **deductivos o inductivos**.

En un argumento deductivo se pretende que sus premisas proveen un fundamento absolutamente concluyente.

Un **argumento deductivo** es **válido** cuando sus premisas y conclusiones están relacionadas de modo tal que es imposible que las premisas sean verdaderas, a menos que la conclusión lo sea también.

En los inductivos, sus premisas proporcionan algún fundamento para sus conclusiones.

Verdad y validez

La **verdad y falsedad** caracterizan las proposiciones.

La **validez e invalidez** caracterizan los argumentos.

La validez de un argumento no garantiza la verdad de su conclusión.

La falsedad de su conclusión no garantiza la invalidez de un argumento. Pero la falsedad de su conclusión sí garantiza que o el argumento es inválido o por lo menos una de sus premisas es falsa.

Condiciones que debe satisfacer un argumento para establecer la verdad de su conclusión:

Debe ser válido y todas sus premisas deben ser verdaderas.

Lógica simbólica

Los símbolos especiales de la lógica moderna nos permiten exhibir con mayor claridad las estructuras lógicas de argumentos cuya formulación puede quedar oscura en el lenguaje ordinario.

Argumentos que contienen Enunciados compuestos

Enunciados simples y compuestos

Los enunciados pueden dividirse en:

- **Simple:** no contiene otro enunciado.
- **Compuesto:** contiene otro enunciado como componente. Las partes componentes de un enunciado compuesto pueden ser enunciados compuestos.

Conjunción: un enunciado compuesto que se forma insertando la palabra "y" entre los dos enunciados. Los enunciados se llaman **conyuntos**. El "." reemplaza a "y".

Una **conjunción es verdadera** si sus conyuntos son ambos verdaderos, pero es falsa en cualquier otra circunstancia.

Cualquier enunciado compuesto cuyo valor de verdad está determinado por los valores de verdad de sus enunciados componentes es un **enunciado compuesto función de verdad**.

Tabla de verdad:

p	Q	p.q
T	T	T
T	F	F
F	T	F
F	F	F

Conjunción: "y", "aunque", "sin embargo", ".", ";".

Negación: se representa con el tilde ~.

Tabla de verdad:

p	~p
T	F
F	T

Disyunción: cuando dos enunciados se combinan disyuntivamente insertando la palabra "o" entre ellos, el enunciado compuesto es una **disyunción** y los dos enunciados se llaman **disyuntos**.

La palabra "o" puede ser **débil o inclusivo**: puede ser una opción, la otra o ambas.

O puede ser **fuerte o exclusivo**: puede ser una opción o la otra pero no ambas.

Una disyunción que usa el "o" exclusivo afirma que por lo menos uno de los disyuntivos es verdadero, pero no ambos son verdaderos.

Tabla de verdad:

p	q	p v q
T	T	T
T	F	T
F	T	T
F	F	F

Expresiones: "ambas ...y... no son ...", "Ni ... ni ... es ..."

Enunciados condicionales

El enunciado componente situado entre el "si" y el "entonces" es llamado el **antecedente** y el componente que sigue al "entonces" es el **consecuente**.

Un condicional afirma que si su antecedente es verdadero, entonces su consecuente es también verdadero.

Podemos afirmar:

- El consecuente se sigue **lógicamente** del antecedente.
- El consecuente se sigue del antecedente por la **definición**.
- La conexión afirmada es causal y debe cubrirse **empíricamente**.

Cualquier condicional de antecedente verdadero y consecuente falso debe ser falso.

Para que el condicional sea verdadero la condición indicada deberá ser falsa.

Se utiliza el símbolo herradura " \rightarrow ". Maneras de leer: "si p entonces q", "p implica q", "p solo si q".

Tabla de verdad:

p	q	$\sim q$	$p \cdot \sim q$	$\sim(p \cdot \sim q)$	p v q
T	T	F	F	T	T
T	F	T	T	F	F
F	T	F	F	T	T
F	F	T	F	T	T

Formas de argumentos y Tablas de verdad

Cualquier argumento se prueba que es inválido si es posible construir otro argumento de exactamente la misma forma con premisas verdaderas y una conclusión falsa.

Variables sentenciales: se representan con letras minúsculas. Son letras por las cuales, o en lugar de las cuales, se pueden sustituir enunciados.

Forma argumental: arreglo de símbolos que contiene variables sentenciales, de modo que al sustituir enunciados por las variables sentenciales, el resultado es un argumento.

Cualquier argumento que sea resultado de la sustitución de enunciados en lugar de variables sentenciales de una forma argumental, se dice que tiene esa forma o que es **una instancia de sustitución de esa forma argumental**.

Forma específica de un argumento: aquella forma argumental de la cual resulta el argumento reemplazando cada variable sentencial por un enunciado simple diferente.

Si la forma específica de un argumento dado puede mostrarse que tiene una instancia de sustitución con premisas verdaderas y conclusión falsa, entonces el argumento dado es inválido.

Para determinar la validez o invalidez de una forma argumental debemos examinar todas las instancias de sustitución posibles de ella para ver si algunas tienen premisas verdaderas y conclusiones falsas.

Las tablas de verdad proporcionan un método mecánico o efectivo de decisión de la validez o invalidez de cualquier argumento.

Para probar una forma argumental que contiene **n** variables sentenciales distintas requiere una tabla de verdad de **2ⁿ** renglones.

Hay dos formas de argumentos inválidas que tienen un parecido superficial con las formas válidas Modus Tollens y Modus Ponens y se conocen con el nombre de **Falacias de Afirmación del Consecuente y Negación del Antecedente**.

Formas sentenciales

Forma Sentencial: cualquier sucesión de símbolos conteniendo variables sentenciales. El resultado es un enunciado.

Todo enunciado que resulta de la sustitución de las variables sentenciales por enunciados en una forma sentencial, se dirá que tiene esa forma o que es una instancia de sustitución de ella.

Una forma sentencial que solo tiene instancias de sustitución verdaderas se dice **tautológica** o que es una **tautología**.

Un enunciado se dice que contradice, o que es una contradicción de otro enunciado, cuando es lógicamente imposible que ambos sean verdaderos.

Una forma sentencial que solo tiene instancias de sustitución falsas se dice que es una **contradicción** y los mismos términos se aplican a sus instancias de sustitución.

Enunciados y formas sentenciales que no son ni tautológicas ni contradictorias se dice que son contingentes o que son **contingencias**.

Dos enunciados se dicen **materialmente equivalentes** cuando tienen el mismo valor de verdad y se simboliza " \equiv ".

Tabla de verdad:

p	q	p q
T	T	T
T	F	F
F	T	F
F	F	T

Dos formas sentenciales se dicen **lógicamente equivalentes** si, no importando que enunciados se sustituyan por sus variables sentenciales, los pares resultantes de enunciados son equivalentes.

A todo argumento corresponde un enunciado condicional cuyo antecedente es la conjunción de las premisas del argumento y cuyo consecuente es la conclusión del argumento.

Ese condicional correspondiente es una tautología si y solo si el argumento es válido.

El método de deducción

Prueba formal de validez

Cuando los argumentos contienen más de dos o tres enunciados simples diferentes se hace difícil utilizar tablas de verdad para probar su validez.

Un método es deducir las conclusiones de sus premisas por una secuencia de argumentos más cortos y más elementales que ya se conoce que son válidos.

Que la conclusión se deduce de las premisas usando argumentos válidos exclusivamente, prueba que el argumento original es válido.

Prueba formal de validez para un argumento: sucesión de enunciados, cada uno de los cuales es una premisa de ese argumento o se sigue de los precedentes por un argumento válido elemental y tal que el último enunciado de la secuencia es la conclusión del argumento cuya validez se está demostrando.

Argumento válido elemental es cualquier argumento que es una instancia de sustitución de una forma de argumento válida. Hay nueve formas de argumentos (formas de argumento válidas elementales) y aceptadas como **Reglas de inferencia**.

Reglas de Inferencia

1. Modus Ponens (MP)

6. Dilema Destructivo (DD)

2. Modus Tollens (MT)

7. Simplificación (Simp)

3. Silogismo Hipotético (HS)

8. Conjunción (Conj)

4. Silogismo Disyuntivo (DS)

9. Adición (Ad)

5. Dilema Constructivo (CD)

La regla de reemplazo

Nos permite inferir de cualquier enunciado el resultado de reemplazar todo o parte de ese enunciado por otro enunciado lógicamente equivalente a la parte reemplazada.

10. Teorema de De Morgan (De M)

11. Conmutación (Conm)

12. Asociación (Asoc)

13. Distribución (Dist)

14. Doble Negación (DN)**15. Transposición (Trans)****16. Implicación Material (Impl)****17. Equivalencia Material (Equiv)****18. Exportación (Exp)****19. Tautología (Taut)**

Hay una diferencia importante entre las primeras 9 y las últimas 10 reglas de inferencia:

- Las primeras pueden aplicarse a renglones enteros de una demostración
- Cualquiera de las últimas puede aplicarse a renglones enteros o partes de renglones.

Demostración de la invalidez**No completitud de las 19 reglas**

Hay argumentos válidos función de verdad cuya validez no es demostrable usando solo las 19 reglas.

Regla de demostración condicional

Se aplica tan solo a argumentos cuyas conclusiones son enunciados condicionales.

- A todo argumento le corresponde un enunciado condicional cuyo antecedente es la conjunción de las premisas del argumento y cuyo consecuente es la conclusión del argumento.
- Un argumento es válido si y solo si su correspondiente condicional es una tautología.

Dado cualquier argumento cuya conclusión es un enunciado condicional, una demostración de su validez usando la regla de Demostración Condicional, se construye suponiendo que el antecedente de su conclusión es una premisa adicional y luego deduciendo el consecuente de su conclusión por una sucesión de argumentos válidos elementales.

Regla de demostración indirecta

Una demostración indirecta de validez para un argumento dado se construye suponiendo como premisa adicional, la negación de su conclusión y deduciendo entonces una contradicción explícita del conjunto aumentado de las premisas.

Lógica de predicados

La lógica de predicados, a pesar de su indecidibilidad teórica, puede resultar útil como medio de representación y manipulación de ciertos tipos de conocimiento que pueden ser necesarios en un sistema de IA.

Representación de hechos simples en lógica

El atractivo de la lógica proposicional consiste en su simplicidad y en la disponibilidad de un procedimiento de decisión. Es muy sencillo representar los hechos del mundo real como proposiciones lógicas escritas en forma de fórmulas bien formadas (fbf).

La lógica de predicados de primer orden permite representar cosas que no serían representables de forma razonable utilizando la lógica proposicional. Se pueden representar hechos del mundo real como sentencias escritas en forma de fbf.

Cuestiones involucradas en el proceso de conversión de frases en lenguaje natural a sentencias lógicas:

- Muchas frases en lenguaje natural son ambiguas
- Normalmente hay más de una forma de representar el conocimiento

- Aún en los casos más sencillos, no es probable que en un conjunto de frases dado, esté contenida toda la información necesaria para razonar sobre el tema en cuestión.

La representación de las relaciones instancia y es-un

Ideas diferentes:

- No es necesario representar explícitamente las relaciones de pertenencia a una clase e inclusión de una clase en otra por medio de los predicados instancia y es-un.
- Es posible obtener varias representaciones de un determinado hecho en un sistema de representación dado. La elección dependerá, en parte del tipo de deducciones que se pretenda agilizar, y en parte del gusto de cada cual. La norma que se debe respetar es el uso consistente de una determinada representación en toda la base de conocimiento. Es necesario que todo el conocimiento sobre el que estas reglas son aplicables esté representado de la forma que las reglas necesiten.

Representación de Funciones calculables y Predicados computables

Resulta conveniente extender la representación para poder expresar **predicados computables**. Cuando se encuentre con un predicado de este tipo, en lugar de buscarlo explícitamente en la base de datos o intentar deducirlo mediante algún tipo de razonamiento, bastará con invocar un procedimiento, especificado que devolverá un valor de verdadero o falso.

Suele ser útil disponer de **funciones computables**.

Método de Resolución

El procedimiento de resolución obtiene demostraciones por refutación. Para probar una proposición se intenta demostrar que su negación lleva a una contradicción con las proposiciones conocidas.

Conversión a forma clausal

Es necesario convertir el conjunto de fórmulas bien formadas a un **conjunto de cláusulas** donde una cláusula se define como una fbf en forma normalizada conjuntiva que no contiene ninguna conectiva \wedge .

Algoritmo: conversión a forma clausal

1. *Eliminar las implicaciones.*
2. *Reducir el ámbito de las negaciones.*
3. *Normalizar las variables.*
4. *Mover todos los cuantificadores a la izquierda de la fórmula sin cambiar su orden relativo.* La fórmula es lo que se conoce como **fórmula normal prenex**. Consiste en un prefijo de cuantificadores seguido por una matriz que está libre de cuantificadores.
5. *Eliminar los cuantificadores existenciales.* Sustituyendo la variables por una referencia a una función que produzca el valor deseado. Como no se conoce la forma de producir ese valor, se debe crear un nuevo nombre de función para cada sustitución.
Estas funciones que generamos se llaman **funciones de Skolem**. Aquellas que no tienen argumentos se llaman **constantes de Skolem**.
6. *Eliminar el prefijo.*
7. *Convertir la matriz en una conjunción de disyunciones.*
8. *Crear una cláusula por cada conjunción.*
9. *Normalizar las variables.*

Después de aplicar todo este procedimiento a un conjunto de fórmulas bien formadas tendremos un conjunto de cláusulas, cada una de las cuales será una disyunción de literales. Estas cláusulas serán las que utilice el procedimiento de resolución para generar demostraciones.

Las bases de resolución

El procedimiento de resolución es un proceso iterativo simple en el cual, en cada paso, se comparan dos cláusulas llamadas cláusulas padres, produciendo una nueva cláusula que se ha inferido de ellas.

Deducción que hará el procedimiento de resolución:

- La resolución opera tomando dos cláusulas tales que cada una contenga el mismo literal. El literal debe estar en forma positiva en una cláusula y negativa en la otra.
- El resolvente se obtiene combinando todos los literales de las dos cláusulas padres excepto aquellos que se cancelan.
- Si la cláusula producida es vacía, es que se ha encontrado una contradicción.

Resolución en lógica proposicional

Una demostración por resolución de la proposición P respecto a un conjunto de axiomas F es:

Algoritmo: Resolución de proposiciones

1. Convertir todas las proposiciones de F a forma clausal.
2. Negar P y convertir el resultado a forma clausal. Añadir la cláusula resultante al conjunto de cláusulas.
3. Hasta que se encuentre una contradicción o no se pueda seguir avanzando repetir:
 - a. Seleccionar dos cláusulas. (cláusulas padres)
 - b. Resolverlas juntas. La cláusula resultante es llamada **resolvente**.
 - c. Si el resolvente es la cláusula vacía, es que se ha encontrado una contradicción. Si no lo es, añadirla al conjunto de cláusulas disponibles.

La resolución se puede ver como un proceso que toma un conjunto de cláusulas que se suponen ciertas y, basándose en la información que las otras proporcionan, genera nuevas cláusulas que representan restricciones a la satisfacibilidad de las cláusulas originales. Surge una contradicción cuando una cláusula se vuelve tan restringida que no hay forma de hacerla verdadera. Esto se indica con la generación de la cláusula vacía.

El algoritmo de unificación

Para detectar las contradicciones, se necesita un procedimiento de emparejamiento que compare dos literales y descubra si existe un conjunto de sustituciones que los haga idénticos. Existe un procedimiento recursivo directo, denominado **algoritmo de unificación** que realiza esto.

La idea de unificación es:

- En primer lugar se comprueba si los predicados coinciden. Si es así, seguimos adelante, si no, no hay forma de unificarlos.
- Si los predicados concuerdan, a continuación se van comprobando los argumentos de dos en dos. Si el primero concuerda, podemos continuar con el segundo y así sucesivamente. Una variable se puede emparejar con otra variable, con cualquier constante o con un predicado, con la restricción de que el predicado no debe contener ninguna instancia de la variable con la que se está emparejando.

La única complicación en este procedimiento es que se debe encontrar una única sustitución consistente para todo el literal.

El **objetivo del procedimiento de unificación** es descubrir al menos una sustitución que permita el emparejamiento de dos literales.

Resolución en lógica de predicados

Dos literales son **contradictorios** si uno de ellos puede unificarse con la negación del otro.

Algoritmo: Resolución

1. Convertir todas las sentencias de F a forma clausal.
2. Negar P y convertir el resultado a forma clausal. Añadirlo al conjunto de cláusulas.
3. Hasta que se encuentre una contradicción o no pueda realizarse ningún progreso o se haya realizado una cantidad de esfuerzo predeterminada, repetir:

- a. Seleccionar dos cláusulas. (Cláusulas padres)
- b. Resolverlas. El resolvente será la disyunción de todos los literales de ambas cláusulas padres, una vez realizadas las sustituciones apropiadas. Utilizar la sustitución producida por la unificación para crear el resolvente. Si existe más de una pareja de literales complementarios, en el resolvente solo se eliminará uno de ellos.
- c. Si el resolvente es la cláusula vacía, se ha encontrado una contradicción. Si no lo es, añadirla al conjunto de cláusulas.

Representación del conocimiento mediante Reglas

Representación declarativa: es aquella en la que el conocimiento está especificado pero en la que la manera en que dicho conocimiento debe ser usado no viene dado.

Para utilizar una representación declarativa se debe aumentar ésta con un programa que especifique lo que debe hacerse con el conocimiento y de qué modo debe hacerse.

Representación procedimental: es aquella en la que la información de control necesario para utilizar el conocimiento se encuentra embebida en el propio conocimiento.

Para utilizar una representación procedimental se necesita aumentarla con un intérprete que siga las instrucciones dadas por el conocimiento.

La diferencia entre el punto de vista declarativo del conocimiento y el procedimental radica en donde se encuentra la información de control.

Programación lógica

Es un paradigma de los lenguajes de programación en el cual las aserciones lógicas se consideran como programas.

Un programa PROLOG está descrito como una serie de aserciones lógicas cada una de las cuales es una **cláusula de Horn**.

Cláusula de Horn. Es una cláusula que tiene, como mucho, un literal positivo.

Los programas de PROLOG están compuestos sólo por cláusulas de Horn, lo que da lugar a dos importantes consecuencias:

1. Debido a la representación uniforme, se puede escribir un sencillo y potente intérprete que resulte eficaz.
2. La lógica de los sistemas de cláusulas de Horn es decidible.

La entrada de un programa es un objetivo que debe ser probado. Se aplica un razonamiento hacia atrás para intentar demostrar el objetivo dadas las aserciones del programa.

El programa se lee de arriba hacia abajo y de izquierda a derecha y la búsqueda será primero en profundidad con vuelta atrás.

Hechos: representan sentencias acerca de objetos específicos.

Reglas: representan sentencias acerca de las diferentes clases de objetos.

Diferencias sintácticas entre las representaciones lógicas y las de PROLOG:

1. En la lógica, las variables están específicamente cuantificadas.
En PROLOG, la cuantificación se realiza de un modo implícito por la forma en que las variables son interpretadas. Las variables comienzan con mayúsculas y las constantes con minúsculas o números.
2. En la lógica existen símbolos explícitos para "y" (\wedge) , "o" (\vee).
En PROLOG, existe un símbolo explícito para "y" (,) pero no para "o".
3. En la lógica, las implicaciones "p implica q" se escriben como $p \rightarrow q$.
En PROLOG, se escribe hacia atrás $q :- p$ ya que el intérprete trabaja hacia atrás sobre un objetivo.

La principal diferencia es que el intérprete de PROLOG fija una estrategia de control.

La estrategia de control PROLOG comienza con una sentencia problema, que en este caso es el objetivo a probar; busca las aserciones que puedan probar el objetivo; considera hechos que prueban el objetivo directamente, y además considera cualquier regla cuya cabeza se empareje

con el objetivo. Para decidir cuando puede aplicar una regla o un hecho al problema que le ocupa utiliza el **procedimiento de unificación estándar**. Razonará hacia atrás desde ese objetivo, hasta que encuentre el modo de terminar con las aserciones en el programa. Considera los caminos utilizando la **estrategia de la búsqueda primero en profundidad**, así como utilizando la **vuelta atrás**. En cada punto en que debe elegir, considera las opciones siguiendo el orden en que éstas aparecen en el programa. Si el objetivo tiene más de una parte de conjuntiva, comprueba las partes en el orden en que éstas aparecen, propagando los enlaces de las variables que determinó la unificación.

Diferencia entre Razonamientos hacia delante y hacia atrás

- Búsqueda hacia delante, a partir de los estados iniciales
- Búsqueda hacia atrás, partiendo de los estados objetivos.

Razonamiento hacia delante a partir de los estados iniciales: Se comienza por construir un árbol de secuencias de movimientos que se pueden presentar como soluciones, empezando por la configuración inicial en la raíz del árbol.

Razonamiento hacia atrás a partir de los estados objetivo: Se comienza construyendo un árbol de secuencias de movimientos que ofrezcan soluciones empezando con la configuración objetivo en la raíz del árbol.

Sistemas de reglas encadenadas hacia atrás.

PROLOG es un ejemplo. Resultan muy eficaces para la **resolución de problemas dirigidos al objetivo**.

En PROLOG, las reglas están restringidas a ser cláusulas de Horn. Esto permite crear un índice, ya que todas las reglas comparten la misma cabeza de regla. Las reglas se emparejan a través del procedimiento de unificación. La unificación intenta encontrar un conjunto de restricciones para las variables y así igualar un subobjetivo con las cabezas de algunas reglas.

Sistemas de reglas encadenadas hacia delante.

En lugar de dirigirse por objetivos, algunas veces se desea ser **dirigido por la información que se va incorporando**.

Combinación del razonamiento hacia adelante y hacia atrás.

Saber si es posible utilizar las mismas reglas tanto para el razonamiento hacia adelante como hacia atrás, depende de la propia forma de las reglas.

- Si tanto las partes derechas como las izquierdas contienen aserciones puras, entonces el encadenamiento hacia delante puede emparejar las aserciones en el lado izquierdo de una regla, y añadir a la descripción del estado las aserciones del lado derecho.
- Pero si se permiten los procedimientos arbitrarios como partes derechas de las reglas, entonces las reglas no serán reversibles.

Razonamiento simbólico bajo incertidumbre.

Razonamiento no monótono.

- **Razonamiento no monótono.** Los axiomas y/o las reglas de inferencia se extienden para que sea posible razonar con información incompleta.
- **Razonamiento monótono.** En el que se extiende la representación para permitir algún tipo de medida numérica sobre la certeza para asociar a cada sentencia.

Los sistemas convencionales de razonamiento están diseñados para trabajar con información que cumple tres propiedades:

- La información es completa con respecto al dominio de interés.
- La información es consistente.
- La única forma en que puede cambiar la información es que se añadan nuevos hechos conforme estén disponibles. Si estos nuevos hechos son consistentes con todos los demás hechos que ya se han afirmado, entonces **ninguno de los hechos pertenecientes al conjunto que eran ciertos pueden refutarse**. Esta propiedad se denomina **monotonía**.

Los sistemas de razonamiento no monótono, se diseñan para que puedan resolver problemas en los que quizás no aparezca alguna de estas propiedades.

Razonamiento por defecto

Se usa el razonamiento no monótono para llevar a cabo el razonamiento por defecto.

Se pretende llegar a unas conclusiones basadas en lo que es más probable que sea cierto. Se explican dos enfoques para lograrlo:

- Lógica no monótona
- Lógica por defecto

Clases comunes de razonamiento no monótono que pueden definirse en estas lógicas:

- Abducción
- Herencia

Lógica no monótona

Es un sistema que proporciona una base para razonar por omisión, en donde el lenguaje de la lógica de predicados de primer orden se aumenta con un operador modal M, que se lee como "es consistente".

Lógica por defecto

Es una lógica alternativa para llevar a cabo un razonamiento basado en omisiones en la que se introduce un nuevo tipo de reglas de inferencia.

Este enfoque permite reglas de inferencia de la forma:
$$\frac{A : B}{C}$$

"Si A es probable y es consistente asumir B, entonces se concluye que C"

Existen diferencias importantes entre las dos teorías:

- En la lógica por defecto las 9 reglas de inferencia se usan como base para calcular un conjunto de extensiones plausibles de la base de conocimiento.
- En la lógica por defecto, las expresiones no monótonas son reglas de inferencia en lugar de expresiones del lenguaje. Es decir, no pueden manipularse mediante otras reglas de inferencia.

Abducción

La derivación de conclusiones de esta forma es otra manera de razonamiento por defecto. Denominamos a este método **razonamiento por abducción**.

Se lee: "Dados dos fdf ($A \rightarrow B$)" y (B) para cualquier expresión A y B, si es consistente asumir A, hacerlo".

Herencia

El razonamiento no monótono se utiliza con mucha frecuencia en la herencia de los valores de los atributos desde la descripción prototipo de una clase hacia las entidades individuales que pertenecen a la clase.

Razonamiento minimalista

Estos métodos se basan en alguna variante de la idea de modelo mínimo. Un modelo es **mínimo** si no existen otros modelos en los que sean ciertas menos cosas.

La idea que hay detrás del uso de los modelos mínimos como base para el razonamiento no monótono sobre el mundo es la siguiente:

Existen muchas menos sentencias ciertas que falsas. Si algo es relevante y cierto, tiene sentido asumir que pertenece a nuestra base de conocimiento. Por lo tanto, asuma que las únicas sentencias ciertas son aquellas que necesariamente deben ser ciertas para que se mantenga la consistencia de la base de conocimiento.

La suposición de un mundo cerrado

Dice que los únicos objetos que satisfacen un predicado P son aquellos que deben hacerlo.

Circunscripción

La suposición de un mundo cerrado posee dos limitaciones esenciales:

- Opera sobre predicados individuales
- Asume que todos los predicados tienen listadas todas sus instancias.

Para manipular estos problemas, se han propuesto distintas teorías sobre la circunscripción.

En estas teorías, se añaden nuevos axiomas a la base de conocimiento existente.

El efecto de estos axiomas consiste en forzar una interpretación mínima sobre una parte seleccionada de la base de conocimiento.

Cuestiones sobre la implementación

Las técnicas de implementación de Razonamiento No Monótono se pueden dividir en dos clases dependiendo del enfoque que se da al problema del control de la búsqueda:

- **Primero en profundidad.**
- **Primero en anchura.**

La resolución de un problema puede hacerse mediante un razonamiento hacia delante o hacia atrás.

- **Razonar hacia delante a partir de lo que se conoce.**
- **Razonar hacia atrás para determinar si alguna expresión P es cierta.** Los sistemas de razonamiento no monótono que soportan este tipo de razonamiento proporcionan todas o alguna de las siguientes características:
 - Permita cláusulas por defecto.
 - Soporte algún tipo de debate, en el que se intente producir argumentos tanto a favor de P como en su contra.

Implementación búsqueda primero en profundidad

Sistemas de mantenimiento de la verdad basados en justificación (JTMS)

Un TMS permite conectar las aserciones mediante una red de dependencias del tipo hoja de cálculo. Un JTMS no conoce nada sobre la estructura de las aserciones en sí mismas. El papel del sistema es servir como libro de anotaciones para un sistema de resolución de problemas, que a su vez le proporciona tanto las aserciones como las dependencias entre las aserciones.

Sistemas de mantenimiento de la verdad basados en la lógica (LTMS)

En un JTMS, el TMS trata los nodos de la red como átomos, significa que no hay relaciones entre ellos excepto aquellas que se sitúan explícitamente en las justificaciones. En particular, un JTMS no tiene problemas en etiquetar simultáneamente a P y no P. No se detectará una contradicción en forma automática.

En un LTMS, se pueden detectar contradicciones de este tipo automáticamente.

Implementación búsqueda primero en anchura

Sistemas de mantenimiento de la verdad basados en suposiciones (ATMS)

Tanto en JTMS como en LTMS se sigue una única línea de razonamiento en cada momento, y cuando es necesario cambiar las suposiciones del sistema, surge una vuelta atrás dirigida por dependencias.

En un ATMS, se mantienen en paralelo varios caminos alternativos. La vuelta atrás se evita a expensas del mantenimiento de múltiples contextos, cada uno de los cuales se corresponde con un conjunto de suposiciones consistentes.

Razonamiento estadístico

En algunas resoluciones de problemas puede resultar adecuado describir las creencias sobre las que no se tiene certeza, pero en las que existen algunas evidencias que las apoyan. Considere este tipo de problemas divididos en dos grupos:

- Problemas en los que se da una cierta aleatoriedad
- El mundo no es aleatorio sino que se comporta normalmente hasta que surge algún tipo de excepción

La probabilidad y el teorema de Bayes

En muchos sistemas de resolución de problemas un objetivo consiste en reunir evidencias sobre la evolución del sistema y modificar su comportamiento sobre la base de las mismas. Para modelar este comportamiento se necesita una teoría estadística de la evidencia. Las estadísticas bayesianas constituyen esta teoría.

Probabilidad condicionada:

$P(H|E)$ --> probabilidad de la hipótesis H dado que se observe la evidencia E.
El teorema de Bayes se enuncia así:

El tamaño del conjunto de probabilidades combinadas que se necesitan para calcular esta función, crece como una función de la forma 2^n , donde n es el nro de proposiciones diferentes que es necesario considerar.

Esto hace que el teorema de Bayes sea inaplicable por diversos motivos:

- El problema de la adquisición de conocimiento es inabarcable. Son necesarias demasiadas probabilidades.
- El espacio que se necesitaría para almacenar las probabilidades es demasiado grande.
- El tiempo empleado en calcular las probabilidades es demasiado grande.

Las estadísticas Bayesianas proporcionan una base atractiva para los sistemas que razonan bajo incertidumbre. Se explican tres de estos mecanismos:

- Incorporación de factores de certeza a las reglas
- Redes bayesianas
- Teoría de Dempster-Shafer

Factores de certeza y sistemas basados en reglas

MYCIN, este sistema se basa en el **uso de razonamiento probabilístico**. Representa la mayor parte de su conocimiento sobre diagnósticos en forma de un **conjunto de reglas**.

A cada regla se le asocia un factor de certeza, que representa una medida sobre la evidencia que existe de que la conclusión sea el consecuente de la regla en el caso de que se describa el antecedente de la misma.

MYCIN Utiliza las reglas para hacer un **razonamiento hacia atrás** de los datos clínicos disponibles a **partir del objetivo**.

Un **factor de certeza FC[h,e]** se define en términos de dos componentes:

- **MB[h,e]**: medida entre 0 y 1 de la **creencia** de que la hipótesis h proporciona la evidencia e. Es 0 si la evidencia no soporta la hipótesis.
- **MD[h,e]**: medida entre 0 y 1 sobre la **incredulidad** de que la hipótesis h proporciona la evidencia e.

$$FC[h,e] = MB[h,e] - MD[h,e]$$

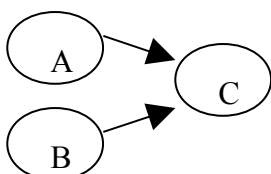
En cada regla basta un único nro para definir tanto el valor de MB como el de MD, y también el de FC, ya que cada regla de MYCIN se corresponde como una parte de la evidencia y cada parte de la evidencia o bien soporta o bien niega una hipótesis (pero nunca ambas cosas).

Los factores de certeza de las reglas de MYCIN los proporcionan los expertos que escriben las reglas.

Propiedades que sería adecuado que cumplan las funciones de combinación:

- Deberían ser conmutativas y asociativas, ya que el orden en el que reúnen las evidencias es arbitrario.
- Hasta que no alcance la certeza, las evidencias adicionales que confirman deben incrementar MB.
- Si las inferencias inciertas se encadenan juntas, el resultado debe ser de menor certeza que cada una de las inferencias por separado.

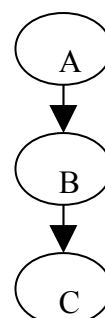
1)



2)



3)



1) Varias partes de evidencia se combinan para determinar el factor de certeza de una hipótesis. Las medidas sobre la creencia o no de una hipótesis vienen dadas dos observaciones s_1 y s_2 se calculan:

$$MB[h, s_1 \wedge s_2] = \begin{cases} 0 & \text{si } MD[h, s_1 \wedge s_2] = 1 \\ MB[h, s_1] + MB[h, s_2] * (1 - MB[h, s_1]) & \text{en caso contrario} \end{cases}$$

$$MD[h, s_1 \wedge s_2] = \begin{cases} 0 & \text{si } MB[h, s_1 \wedge s_2] = 1 \\ MD[h, s_1] + MD[h, s_2] * (1 - MD[h, s_1]) & \text{en caso contrario} \end{cases}$$

La medida sobre la creencia en h

- es 0 si no se cree en h con certeza.
- dadas dos observaciones, es la medida sobre la creencia dada solo por una observación más algún incremento debido a la segunda observación.

2) Es necesario calcular el factor de certeza de una combinación de hipótesis, es necesario cuando se necesita conocer el factor de certeza de un antecedente de una regla que contiene varias cláusulas.

Las fórmulas son:

$$MB[h_1 \wedge h_2, e] = \min (MB[h_1, e], MB[h_2, e])$$

$$MB[h_1 \vee h_2, e] = \max (MB[h_1, e], MB[h_2, e])$$

MD se calcula de forma análoga.

3) Las reglas se encadenan de forma que el resultado de la incertidumbre que sale de una regla es la entrada de la otra.

La fórmula es:

$$MB[h, s] = MB' [h, s] * \max (0, FC[s, e])$$

Redes Bayesianas

Constituyen un enfoque alternativo al de factores de certeza, en el que el formalismo de razonamiento bayesiano se preserva y se confía en la modularidad del mundo que se intenta modelar.

No es necesario utilizar una tabla de probabilidades.

Se puede usar una representación más local en donde se describen grupos de sucesos que interactúen.

Teoría de Dempster-Shafer

Considera conjuntos de proposiciones y les asigna a cada uno de ellos un intervalo:

[creencia, verosimilitud]

Con el que debe indicarse el grado de creencia. La creencia mide la fuerza de la evidencia a favor de un conjunto de proposiciones.

El rango va de 0 (que indica evidencia nula) a 1 (que denota certeza).

La verosimilitud se define:

$$Pl(s) = 1 - Bel (\neg s)$$

Su rango va de 0 a 1 y mide el alcance con que la evidencia a favor de $\neg s$ deja espacio para la creencia en s .

Si se tiene evidencia cierta a favor de $\neg s$ entonces **Bel** ($\neg s$) es 1 y **Pl**(s) es 0.

El intervalo creencia – verosimilitud mide no solo el nivel de creencia sobre algunas proposiciones, sino también la cantidad de información que se tiene.

Con los intervalos se clarifica el hecho de que no se posee información al comenzar.

Sistemas semánticos para Representación del conocimiento

Una representación es un conjunto de convenciones sobre la forma de describir un tipo de cosas.

Es evidente que la descripción nodo y enlace es una buena descripción con respecto al problema planteado, ya que resulta fácil de hacer y, una vez que se tiene, el problema resulta simple de resolver.

La idea es que una buena descripción de acuerdo con las convenciones de una buena representación, es una puerta abierta para la resolución del problema; una mala descripción, que utiliza una mala representación, es un obstáculo que impide la resolución del problema.

Estructuras de Ranura y Relleno Débiles

La **herencia monótona** se puede desarrollar más eficazmente con estas estructuras que con la lógica pura, y la **herencia no monótona** puede soportarse muy fácilmente. La razón por la que la herencia se ejecuta de un modo sencillo, es que en los sistemas de ranura y relleno el conocimiento está estructurado como un conjunto de entidades y todos sus atributos.

Enfoques de este tipo de estructuras: las redes semánticas y los marcos.

A estas estructuras de "conocimiento pobre" se les denomina "débiles". En las estructuras de ranura y relleno "fuertes" se establecen mayores compromisos en relación con el contenido de las representaciones.

Redes semánticas

La información contenida en ellas se representa como un conjunto de nodos conectados unos con otros mediante un conjunto de arcos etiquetados que representan las relaciones entre los nodos.

Búsqueda de intersección

Una de las primeras formas de usar las redes semánticas fue para encontrar relaciones entre objetos. Este proceso se llama **búsqueda de intersección**.

Utiliza una de las grandes ventajas de las estructuras de ranura y relleno sobre las representaciones puramente lógicas, ya que tienen la ventaja de la organización del conocimiento basado en entidades, que proporcionan las representaciones de ranura y relleno.

Representación de predicados no binarios

Las redes semánticas se pueden considerar como un modo natural de representar las relaciones que podrían aparecer como instancias de los *predicados binarios* en la lógica de predicados.

Los *predicados unarios* de la lógica se pueden considerar como predicados binarios.

Los predicados de *tres o más argumentos* pueden convertirse a forma binaria creando un nuevo objeto que represente todo el predicado y después introduciendo predicados binarios para describir la relación con este nuevo objeto de cada uno de los argumentos originales.

Marco

Es una colección de atributos, normalmente llamados ranuras, con valores asociados, que describe alguna entidad del mundo.

Los marcos como conjuntos e instancias

Cada marco representa, ya una clase (un conjunto), ya una instancia (un elemento de la clase).

La **relación es-un**, es la relación subconjunto. Ej.: el conjunto de los Leones es un subconjunto de los Felinos Salvajes.

La **relación instancia** corresponde con la relación **elemento-de**. Ej.: Julio es un elemento del conjunto de Leones.

Tanto la **relación es-un**, como la **relación instancia** tienen atributos inversos que se denominan **subclases y todas las instancias**.

Debido a que una clase representa un conjunto, existen dos clases de atributos que se pueden asociar con ésta. Existen atributos acerca del conjunto en sí mismo, y también atributos para ser heredados por cada elemento del conjunto.

Estructuras de Ranura y Relleno Fuertes

Las redes semánticas y los sistemas de marcos implementan estructuras muy generales para representar el conocimiento.

Dependencia conceptual (DC)

Es una teoría sobre la representación del tipo de conocimiento sobre los eventos que aparecen en las frases de lenguaje natural.

El objetivo consiste en representar el conocimiento de alguna forma que:

- facilite extraer inferencias de las frases
- sea independiente del lenguaje en el que están las frases originalmente

La DC proporciona tanto una estructura como un conjunto específico de primitivas, a un nivel concreto de granularidad.

Yo le di un libro al hombre

Los símbolos tienen los siguientes significados:

- Las flechas indican direcciones de la dependencia
- La flecha doble indica los tipos de enlaces entre el actor y la acción
- P indica tiempo pasado
- ATRANS es una de las acciones primitivas utilizadas por la teoría
- Indica transferencia de posesión
- o indica la relación OBJECT CASE
- R indica la relación RECIPIENT CASE

En DC, las representaciones de las acciones se construyen a partir de un conjunto de **acciones primitivas**. Un típico conjunto es el siguiente:

ATRANS	Transferencia de una relación abstracta (dar)
PTRANS	Transferencia de una localización física de un objeto (ir)
PROPEL	Aplicación de la fuerza física (empujar)
MOVE	Movimiento de una parte del cuerpo por su dueño (patear)
GRASP	Asimiento de un objeto por un actor (empuñar)
INGEST	Ingestión de un objeto por parte de un animal (comer)
EXPEL	Expulsión de algo del cuerpo de un animal (llorar)
MTRANS	Transferencia de información mental (decir)
MBUILD	Construcción de información nueva aparte de la vieja (decidir)
SPEAK	Producción de sonidos (hablar)
ATTEND	Concentración de un órgano sensorial hacia un estímulo (escuchar)

Un segundo conjunto de bloques contruidos de DC es el conjunto de las **dependencias permitidas entre las conceptualizaciones** descriptas en una frase.

Categorías conceptuales primitivas:

ACT Acciones

- PP** Objetos (productores de imágenes)
- AA** Modificadores de acciones (asistentes de acciones)
- PA** Modificadores de PP (asistentes de imágenes)

Las estructuras de dependencia son en sí mismas conceptualizaciones y pueden servir como componentes de estructuras de dependencias más grandes.

Guión

Es un estructura que describe una secuencia estereotipada de eventos en un contexto concreto.

Está formado por un conjunto de ranuras. Asociada a cada ranura puede estar alguna información sobre qué tipo de valores puede contener, así como un valor por defecto.

Componentes importantes de un guión:

Entry conditions	Condiciones que deben satisfacerse antes de que puedan ocurrir los eventos que se describen en el guión.
Result	Condiciones que deberán ser ciertas después de que ocurran los eventos.
Props	Ranuras que representan objetos que aparecen involucrados en los eventos.
Roles	Ranuras que representan a gente que aparece involucrada en los eventos.
Track	La variación específica sobre un patrón más general representado por este guión concreto.
Scenes	La secuencia de eventos que ocurre.

Los guiones resultan útiles porque en el mundo real aparecen patrones en la ocurrencia de los eventos. Estos patrones surgen debido a las relaciones de causalidad entre los eventos. Los agentes llevarán a cabo una acción de forma que entonces son capaces de realizar otra. Los eventos que se describen en un guión forman una gigantesca cadena causal. El comienzo de la cadena es el conjunto de condiciones de entrada, el final es el conjunto de resultados los cuales pueden capacitar posteriores eventos o secuencias de ellos.

CYC

Es un proyecto de una gran base de conocimiento cuyo propósito es el de capturar el conocimiento humano de sentido común.

El objetivo es codificar el amplio cuerpo de conocimiento. Esta base de conocimiento podría combinarse con bases de conocimiento especializadas para producir sistemas que sean menos frágiles que la mayoría.

Como DC, CYC representa una teoría concreta para describir el mundo y como DC, puede usarse para tareas de IA tales como la comprensión del lenguaje natural.

CYC es mas comprensible; mientras que DC proporciona una teoría concreta para la representación de eventos, CYC contiene representaciones de eventos, objetos, actitudes y muchas otras.

Unidad 4: Ingeniería del Conocimiento

PROLOG (Programación Lógica)

PROLOG es un lenguaje de programación que se usa para resolver problemas en los que entran en juego objetos y relaciones entre objetos. Se ha convertido en el principal entorno de programación para IA.

Una breve historia del Prolog

PROLOG significa PROgramming Logia, programación basada en la lógica. Fue inventado alrededor de 1970 en la universidad de Marsella, Francia.

¿Para qué sirve Prolog?

Está diseñado para manejar **problemas lógicos** (en los que se necesitan tomar decisiones de una forma ordenada), intenta hacer que la computadora razone la forma de encontrar una solución.

Lenguaje Procedural vs. Declarativo

Lenguaje Procedural: permiten al programador decirle a la computadora lo que tiene que hacer, paso a paso, hasta alcanzar una conclusión.

PROLOG, es **declarativo**, necesita que se declaren reglas y hechos sobre símbolos específicos y luego se le pregunte sobre si un objetivo concreto se deduce lógicamente a partir de los mismos. Al trabajar con un lenguaje declarativo se da información sobre un tema determinado, se definen las relaciones que existen entre esos datos y finalmente se construyen preguntas o cuestionamientos, quedándole al lenguaje la tarea de elaborar las conclusiones mediante un razonamiento lógico.

Inteligencia Artificial

Inteligencia: capacidad o habilidad para percibir hechos y proposiciones y sus relaciones y razonar sobre ellos.

Que un programa sea inteligente requiere que actúe inteligentemente como un ser humano.

Un programa inteligente exhibe un comportamiento similar al de un humano cuando se enfrenta a un problema similar

El programa no necesita pensar como un humano, pero debe actuar como tal.

Los dos problemas más significativos de IA son: **sistemas expertos y procesamiento del lenguaje natural.**

Sistemas Expertos

Es un programa de computadora que contiene conocimientos acerca de un determinado campo y cuando es interrogado responde como un experto humano.

Contiene información (una base de conocimiento) y una herramienta para comprender las preguntas y responder la respuesta correcta examinando la base (un motor de inferencia).

Procesamiento del lenguaje natural

Es la técnica que fuerza a las computadoras a entender el lenguaje humano.

Relación con la Lógica

Lógica: forma de representar argumentos de manera que fuera posible comprobar si estos eran válidos o no.

PROLOG trabaja con lógica proposicional o lógica de predicados o cálculo proposicional.

Tiene un motor de inferencia incorporado que busca los hechos y construye conclusiones lógicas.

La programación en PROLOG consiste en:

- declarar algunos hechos sobre los objetos y sus relaciones
- definir algunas reglas sobre los objetos y sus relaciones
- hacer preguntas sobre los objetos y sus relaciones

Hechos

La primera forma de combinar un objeto y una relación es usarlos para definir un hecho. La sintaxis es:

relación(objeto)

La **relación** se conoce como **predicado** y el **objeto** como el **argumento**.

Puntos importantes:

- Los nombres de todos los objetos y relaciones deben comenzar con minúscula.
- Primero se escribe la relación y luego los objetos separándolos mediante comas y entre paréntesis.
- Al final del hecho debe ir un ".".
- El carácter "_" en el predicado significa que es una sola palabra.
- Dos hechos coinciden si sus predicados son lo mismo y si cada uno de sus correspondientes argumentos son iguales entre sí.

Prestar atención en el orden en que se escriben los objetos dentro del paréntesis.

Variables

Se pueden utilizar nombres como X que representen objetos a los que el mismo PROLOG les dará valor.

Cuando PROLOG utiliza una determinada variable ésta puede ser instanciada o no. Una variable no está instanciada cuando, todavía no se conoce lo que representa.

Las variables deben comenzar con mayúscula.

La variable anónima es un único carácter de subrayado. Se utiliza para evitar el tener que imaginar continuamente diferentes nombres de variables cuando no se van a utilizar en ningún otro sitio de la cláusula.

Reglas

Se usa una regla cuando se quiere significar que un hecho depende de un grupo de otros hechos. Una regla consiste en una cabeza y un cuerpo. Se encuentran separadas por ":-" esto se pronuncia "si".

La cabeza describe que hecho es el que la regla intenta definir, mientras que el cuerpo describe la conjunción de objetivos que deben satisfacer para que la cabeza sea cierta.

Las reglas hacen que el PROLOG pase de ser un diccionario o una base de datos, en el que se pueda buscar, a ser una máquina lógica, pensante.

Una regla es una afirmación general sobre objetos y sus relaciones.

Cláusulas

Se usa para referirse a un hecho o a una regla.

Preguntas

Se representa igual que un hecho, salvo que delante se pone un símbolo ?-

Cuando se hace una pregunta, PROLOG efectúa una búsqueda por toda la base de datos, localizando hechos que coincidan con el hecho en cuestión.

Conjunciones y Backtracking

Un problema puede tener dos objetivos separados. El "y" representa la conjunción de los dos objetivos: lo que se quiere hacer es satisfacer ambos, primero uno y luego el otro.

PROLOG buscará el primer objetivo, si lo encuentra marcará el lugar e intentará satisfacer el segundo, si lo logra se marcará también dicha posición en la base de conocimientos, lo que determina que se encontró la solución.

Si el segundo objetivo no se logra, PROLOG intentará re-satisfacer el objetivo anterior, continuando en la marca de posición y luego satisfacer el segundo.

A este comportamiento del PROLOG de intentar repetidamente satisfacer o resatisfacer los objetivos de una conjunción, se lo llama **reevaluación (o backtracking)**.

Estructura de un Programa en PROLOG

Composición de un programa. Cláusulas. Predicados. Dominios.

Los programas están organizados en 4 etapas:

- Cláusulas
- Predicados
- Dominios
- Objetivos

No todas necesitan estar presentes.

Cláusulas

En esta sección se listan los hechos que construyó con los objetivos y las relaciones. También puede contener reglas.

Predicados

Son las relaciones. Siempre que el programa vaya a usar un predicado particular en las cláusulas, necesita declararlo.

nombrepredicado (argumento1, argumento2)

Se pueden tener múltiples declaraciones de predicados, con diferentes dominios de argumentos.

Dominios

Necesita decirle los argumentos que van a usar los predicados. Necesita conocer que "tipo" de cosas puede ser un argumento.

Tipos de dominios estándares:

- **Symbol**

Hay dos tipos de símbolos:

- Un grupo de caracteres consecutivos que comienzan con un carácter en minúscula.
- Un grupo de caracteres consecutivos que comienzan y terminan con un marca de dobles comillas ("").

Los símbolos y cadenas parecen iguales, PROLOG los interpreta diferentes. Los símbolos necesitan más memoria que las cadenas, pero se identifican más rápidamente cuando se ejecuta un programa.

- **String**

Cualquier grupo de caracteres consecutivos que comienza y termina con una marca de doble comilla ("). Es lo mismo que el segundo tipo de símbolo, sin embargo los símbolos y cadenas no son tratados de la misma forma por PROLOG.

- **Integer**

Cualquier nro comprendido entre (e incluyendo) -32768 y 32767. Los enteros se almacenan como valores de 16 bits.

- **Real**

Nro real en el rango +/- 1 E-307 a +/- E+308. El formato incluye estas opciones: signo, nro, punto decimal, fracción, E, signo para el exponente, exponente. El entero necesita menos memoria.

- **Char**

Cualquier carácter posicionado entre dos marcas de comillas sencilla (' ').

- **File**

Para declarar un dominio de archivo, se debe declarar los nombres de archivos simbólicos de los archivos que se usarán.

Los nombres de archivos simbólicos deben comenzar con letra minúscula. Hay dos tipos de archivos:

- 1- *Archivos predefinidos*: son los archivos que están siempre disponibles en PROLOG. Ej.: printer, keyboard, screen.

- 2- *Archivos definidos por el usuario*: son los archivos a los que uno da nombres. Puede usarse cualquier nombre que no esté reservado por PROLOG.

Objetivos

Le dice a PROLOG lo que ha de encontrar o lo que desea que la computadora haga con la información que se le ha suministrado en las otras tres secciones.

Un programa PROLOG puede tener dos tipos diferentes de goal: interno o externo.

- **Goal interno:** se ejecutará y automáticamente procederá a aprobar o desaprobar el objetivo y le informará de lo obtenido.
- **Goal externo:** comenzará a ejecutarse y luego esperará que se le introduzca un objetivo. Una vez que se verifica si se junta el objetivo, el programa le pedirá uno nuevo y así sucesivamente.

Ingeniería del Conocimiento

Es la disciplina que trata de la forma en que se organizan, construyen y verifican las bases de conocimiento.

El Ingeniero en Conocimiento es el encargado de entrevistar a los expertos reales, para aprender sobre lo que ellos saben, y poder así organizar la información obtenida para construir una Base de Conocimientos.

Organización de una Base de conocimiento

Una buena forma de organizar la información es situar los objetos más probables lo más cerca del comienzo. Para saber cuales son los objetos más probables, puede crear una base de conocimientos y dejar que sea usada un período corto de tiempo y va registrando el nro de veces que se selecciona cada objeto.

Otra forma es situar cerca del comienzo de la base de conocimientos aquellos atributos que causan la poda de la mayor parte del "árbol". Puede ser difícil determinar que atributos causan mayor efecto.

Encontrando el Experto

- Es difícil determinar quien es o no es un verdadero experto.
- Dos expertos diferentes en la misma materia tienen a menudo opciones contradictorias, lo que hace difícil saber cual usar.

Cuando las opiniones de los expertos difieren se puede elegir entre 3 opciones:

- Seleccionar un experto e ignorar al otro. Puede ser que la base de conocimiento contenga alguna información errónea.
- Promediar la información. Usar solo la información que es común entre los expertos.
- Incluir el conocimiento de ambos expertos en el sistema y dejar al usuario decidir.
- Otro punto problemático es que la mayoría de los expertos humanos no saben lo que saben. En consecuencia, puede ser difícil extraer toda la información necesaria.
- Algunos expertos estarán simplemente menos inclinados a perder tiempo diciéndole todo lo que sabes sobre la materia.

Verificaciones de la BC

Si se encontró un experto que es cooperativo y da una descripción completa de su materia de experto, todavía se enfrenta el problema de verificar que ha transcritto e introducido correctamente los conocimientos en la computadora. Debe testear la base de conocimientos.

Pero como? Para la mayoría de los sistemas expertos no hay forma de verificar completamente que la BC sea exacta.

Soluciones:

- Usando técnicas de muestreo estadísticas, puede idear una serie de tests que producirán cualquier nivel de confianza que se desee.
- Autocomprobación del sistema por consistencia y ver que toda la información de la BC concuerda consigo misma.

Definición de Sistema Experto

Podríamos concretar en dos las ideas claves que han conducido a la construcción de los "sistemas expertos" y a su popularización.

1. En énfasis en la representación, adquisición y uso del conocimiento especializado.
2. El reconocimiento de que los sistemas, además de resolver problemas, deben poseer otros atributos que se suponen propios de los expertos humanos:
 - Capacidad para adquirir nuevos conocimientos y perfeccionar los que ya poseen.
 - Capacidad para justificar sus conclusiones.
 - Capacidad para explicar por qué hacen sus preguntas cuando están intentando resolver un problema.
 - Capacidad conversacional para todo ello.

Definición funcional

Para que un sistema informático pueda llamarse "experto", ha de satisfacer un criterio similar al conocido "test de Turing". Visto como un caja negra, sea indistinguible por su comportamiento de un experto humano.

Relaciones de funciones que un sistema experto debe ser capaz de realizar:

- Resolver problemas muy difíciles tan bien o mejor que un experto humano.
- Razonar heurísticamente, usando reglas que los expertos humanos consideran eficaces.
- Interactuar eficazmente y en lenguaje natural con las personas.
- Manipular descripciones simbólicas y razonar sobre ellas.
- Funcionar con datos erróneos y reglas imprecisas.
- Contemplar simultáneamente múltiples hipótesis alternativas.
- Explicar por qué plantean sus preguntas.
- Justificar sus conclusiones.

Definición estructural

Cuando diseñamos un sistemas informático, mezclamos en el código, los conocimientos y los procedimientos que, actuando sobre esos conocimientos, permiten resolver cada problema específico.

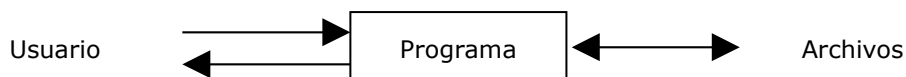
Es aconsejable separa los dos componentes: conocimientos y procedimientos. Será más fácil conseguir que el sistema posea dos características básicas:

- flexibilidad para adquirir y modificar sus conocimientos
- transparencia en la explicación de sus razonamientos y conclusiones.

Ello se traduce en la aparición de un nivel estructural nuevo en el diseño de los sistemas, podemos ilustrar con las siguientes fases evolutivas:

A. En un sistema elemental, el programa contiene no sólo el conocimiento y los procedimientos, sino también las informaciones sobre la forma de comunicación con el usuario y las formas de acceder a los datos.

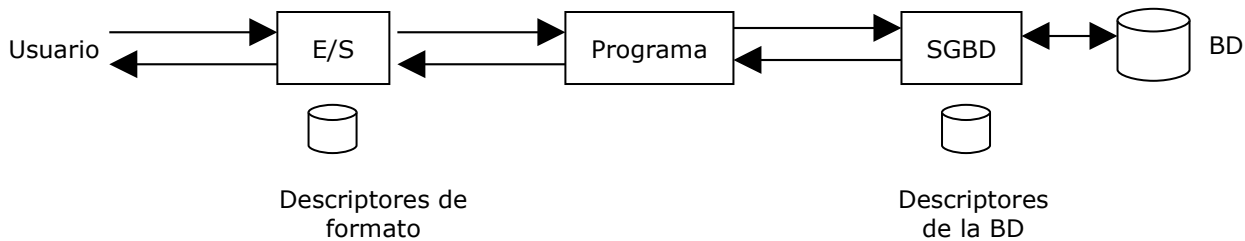
Sistema elemental:



B. El programa se independiza de los formatos de comunicación con el usuario gracias a la capa de entrada – salida del sistema operativo. Se independiza de las estructuras de datos mediante el uso de un sistema de gestión de bases de datos.

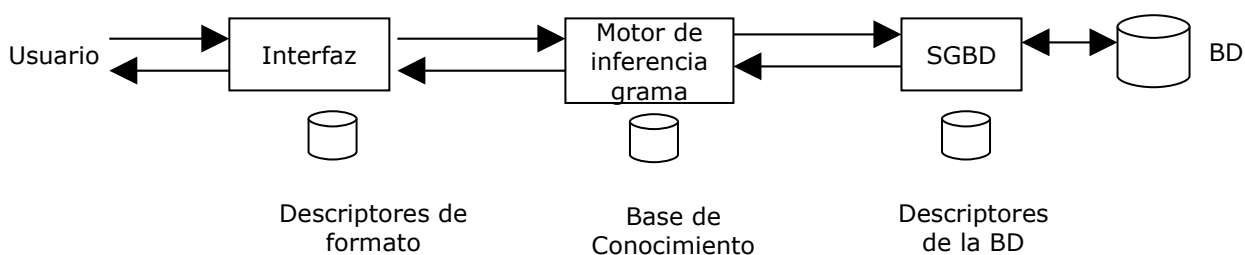
Problemas: todo intento de modificación del conocimiento exige cambios en el programa.

Sistema con subsistema de E/S



C. En un sistema experto se independiza el conocimiento de los procedimientos que hacen uso de él, por tanto, dos módulos diferenciados: la base de conocimiento y el motor de inferencia.

Sistema con Base de conocimiento



Así podemos distinguir tres componentes estructurales básicos en un sistema experto:

- **Base de hechos:** contiene el conocimiento declarativo.
- **Base de conocimientos:** formado por el conocimiento específico y procedimental acerca de la clase de problemas en los que el sistema es experto.
- **Motor de inferencia:** controla al resto del sistema en sus funciones deductivas.

Tres son los tipos de personas que se relacionan con un sistema experto:

- **Usuario final:** dialoga con el sistema para resolver problemas o para aprender.
- **Experto humano:** comunica su conocimiento para construir el sistema.
- **Ingeniero de conocimiento:** diseña las estructuras de datos mas adecuadas para la representación del conocimiento y que traduce a tales estructuras los conocimientos del experto humano.

Armazones de Sistemas Experto

Los sistemas expertos tenían bastantes puntos en común. Fue posible separar el intérprete y el conocimiento específico del dominio, y así se pudo crear un sistema capaz de construir nuevos sistemas expertos sin más que añadir nuevos conocimientos correspondientes al dominio del nuevo problema.

Los intérpretes resultantes se llaman **armazones (shells)**.

- Los primeros armazones de sistemas expertos proporcionaban mecanismos para la representación del conocimiento, el razonamiento y la explicación.
- Se le añadieron herramientas para la adquisición del conocimiento.
- Integrar sistemas expertos con otros tipos de programas.

Una de las características importantes que debe tener un almacén es un interfaz fácil de usar, entre un sistema experto y un gran entorno de programación.

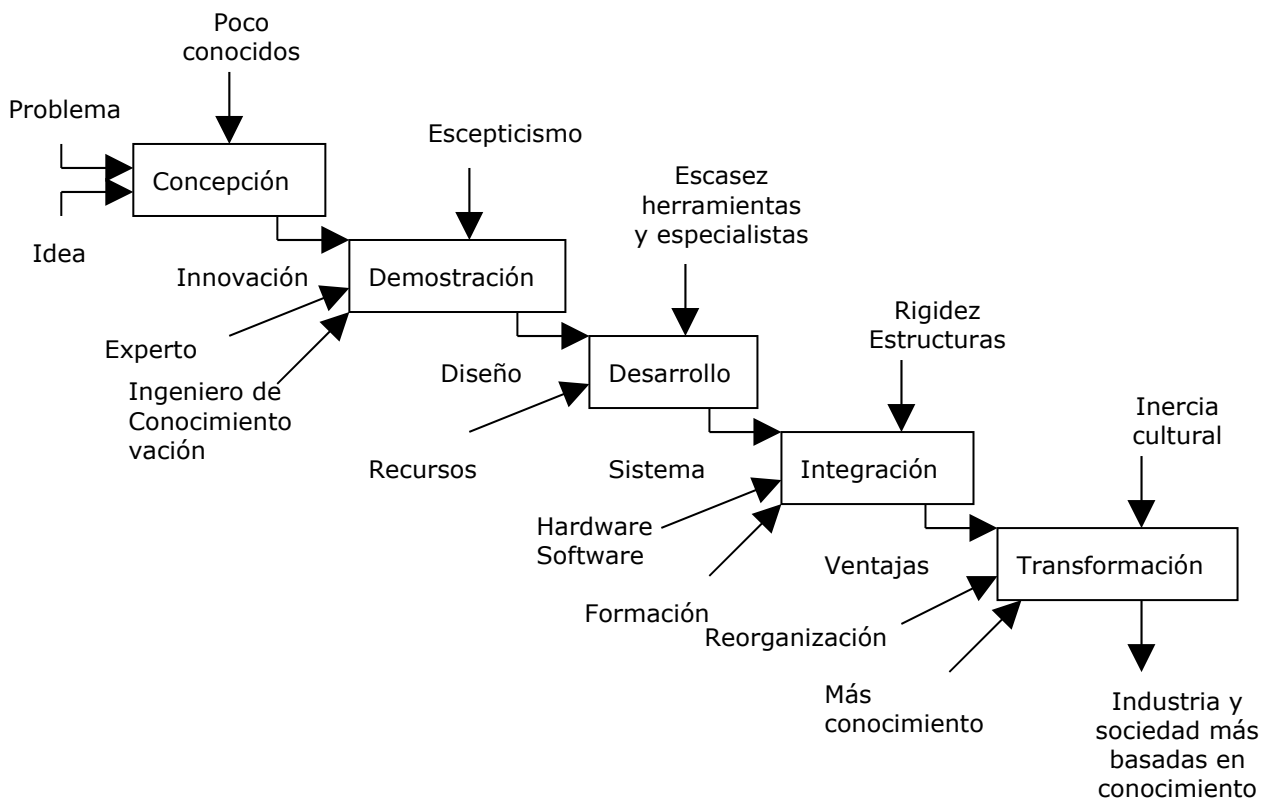
Aplicaciones de los Sistemas Expertos

Ventajas de la aplicación de sistemas expertos

La utilidad de un sistema experto se basa principalmente en la **eficiencia y conveniencia**.

1. Está accesible para usarse las 24 horas.
2. Se pueden crear muchos sistemas expertos, mientras que el nro de expertos humanos puede ser limitado.
3. Los expertos informatizados nunca mueren, llevándose el conocimiento con ellos.
4. El experto informatizado está siempre rindiendo al máximo.
5. El sistema no tiene personalidad.
6. Puede adquirir un nuevo experto simplemente copiando el programa de una máquina a otra.

Fases de la "inserción social" de los Sistemas Expertos



Sistemas Expertos más conocidos

Evolución de los sistemas expertos

- **Etapas de invención:** se construyen sistemas.
- **Etapas de prototipos:** se diseñan los sistemas expertos más conocidos actualmente.
- **Etapas de experimentación:** las ideas obtenidas sobre la representación del conocimiento en los prototipos se van asentando y aparecen los armazones con los que se desarrollan rápidamente otros muchos sistemas expertos.
- **Etapas de industrialización:** se fundó la primera empresa dedicada a sistemas expertos y herramientas para su desarrollo.

Unidad 5: Redes Neuronales

Modelos Conexionistas

Aunque todavía se ignora mucho sobre la forma en que el cerebro aprende a procesar la información, se han desarrollado modelos que tratan de mimetizar tales habilidades, denominados **Redes Neuronales o modelos de computación conexionista**.

La elaboración de estos modelos supone en primer lugar la deducción de características esenciales de las neuronas y sus conexiones y en segundo lugar la implementación del modelo en una computadora de forma que se pueda simular.

Origen del paradigma de computación conexionista

La IA se separó casi desde su inicio en dos ramas bien diferenciadas:

- Se trató de modelar la actividad racional mediante **sistemas formales de reglas y manipulación simbólica**, constituyendo la rama **simbólico-deductiva**.
- Se desarrollaron modelos computacionales inspirados en las redes neuronales biológicas, denominados **inductivos** o **subsimbólicos**.

Podríamos situar el origen de los modelos conexionistas con la definición de la neurona: dispositivo binario con varias entradas y salidas.

Ideas fundamentales que han influido en el campo de las redes neuronales:

- una percepción o concepto se representa en el cerebro por un conjunto de neuronas activas simultáneamente.
- La memoria se localiza en las conexiones entre las neuronas.

Regla de aprendizaje de Hebb: indica que las conexiones entre dos neuronas se refuerzan si ambas son activadas.

Redes Neuronales Biológicas

El cerebro humano se compone de decenas de billones de neuronas interconectadas entre sí formando circuitos o redes que desarrollan funciones específicas.

Una **neurona** recoge señales procedentes de otras neuronas a través de **dendritas**. La neurona emite impulsos de actividad eléctrica a lo largo del **axón**, que se divide en millares de ramificaciones.

Las extremidades de estas ramificaciones llegan hasta las dendritas de otras neuronas y establecen unas conexiones llamadas **sinapsis**, en las cuales se produce una transformación del impulso eléctrico en un mensaje neuroquímico, mediante la liberación de unas sustancias llamadas **neurotransmisores**.

El efecto de los neurotransmisores sobre la neurona receptora puede ser **excitatorio o inhibitorio**.

Las señales inhibitorias recibidas por una neurona se combinan y en función de la estimulación total recibida, la neurona toma un cierto nivel de activación, que se traduce en la **generación de breves impulsos nerviosos** con una frecuencia de disparo, y su propagación a lo largo del axón hacia las neuronas con las cuales sinapta.

De esta manera la información se transmite de unas neuronas a otras y va siendo procesada a través de las conexiones sinápticas y las propias neuronas.

Redes Neuronales Artificiales

Red Neuronal: un sistema compuesto de muchos elementos simples de procesamiento los cuales operan en paralelo y cuya función es determinada por:

- la estructura de la red y
- el peso de las conexiones

realizándose el procesamiento en cada uno de los nodos o elementos de cómputo.

Es un sistema compuesto por:

Unidades de procesamiento simples que trabajan en paralelo.

La UP trabajan en paralelo y el procesamiento se da en UP

La función de la red esta determinada por la **ESTRUCTURA DE LA RED Y EL PESO DE CONEXIONES.**

Otra definición: es un procesador paralelo masivamente distribuido que tiene una facilidad natural para el almacenamiento de conocimiento obtenido de la experiencia para luego hacerlo utilizable.

Estructura de las redes neuronales

Las neuronas se modelan mediante **unidades de proceso.**

Cada unidad de proceso se compone de:

- una red de conexiones de entrada
- una función de red encargada de computar la entrada total combinada de todas las conexiones
- un núcleo central de proceso, encargado de aplicar la función de activación
- la salida, por donde se transmite el valor de activación a otras unidades.

La función de red es típicamente el sumatorio ponderado, mientras que la función de activación suele ser alguna función de umbral o sigmoideal.

Conexiones ponderadas: hacen el papel de las conexiones sinápticas.

Peso sináptico (w_i): el peso de la conexión equivale a la fuerza de la sinapsis.

El valor de los pesos y el signo define el tipo (excitatorio/inhibitorio) y la intensidad de la influencia.

Función de red o de Propagación: calcula el valor de base o entrada total a la unidad, generalmente como suma ponderada de todas las entradas recibidas.

Equivale a la combinación de las señales excitatorias e inhibitoras de las neuronas biológicas.

Función de activación: característica principal de las neuronas.

Se encarga de calcular el nivel o estado de activación de la neurona en función de la entrada total.

Salida: calcula la salida de la neurona en función de la activación de la misma.

El valor de salida cumpliría la función de la tasa de disparo en las neuronas biológicas.

Función de Red o de Propagación

Podemos citar entre las más importantes:

A. Función lineal de base (FLB): es la más utilizada.

Consiste en la sumatoria ponderada de todas las entradas. Función de primer orden.

$$\text{Net}(X_i, W_{ij}) = X_i * W_{ij}$$

B. Función radial de base (FRB): función de segundo orden, no lineal. Representa la distancia a un determinado patrón de referencia.

Función de Activación

Podemos distinguir entre:

1. **Funciones lineales:** La salida es proporcional a la entrada.
2. **Funciones de umbral:** La salida es un valor discreto que depende de si la estimulación total supere o no un determinado valor de umbral.
3. **Funciones no lineales:** No proporcionales a la entrada.
 - a. **Función sigmoideal o logística:** La función de activación más empleada en la actualidad.

Función continua no lineal. Posee un rango comprendido entre 0 y 1.

El problema de trabajar con modelos no lineales radica en que son difíciles de describir en términos lógicos o matemáticos convencionales.

Comparación entre RNB y RNA

Redes Neuronales Biológicas	Redes Neuronales Artificiales
Neuronas	Unidades de Proceso
Conexiones sinápticas	Conexiones ponderadas
Efectividad de las sinapsis	Peso de las conexiones
Efecto excitatorio o inhibitorio de una conexión	Signo del peso de una conexión
Efecto combinado de las sinapsis	Función de propagación o de red
Activación → tasa de disparo	Función de activación → salida

Formas de interconexión de las RNA

Para diseñar una red debemos establecer como estarán conectadas unas unidades con otras y determinar los pesos de las conexiones.

Lo más usual es disponer las unidades en forma de capas, pudiéndose hablar de redes de una, de dos o de más de dos capas, (redes multicapa).

Lo usual es disponer de tres o más capas:

- **Capa de entrada:** es la primera capa y actúa como buffer de entrada, almacenando la información bruta o realizando un pre-proceso de la misma.
- **Capa de salida:** actúa como interfaz o buffer de salida, almacenando la respuesta para que pueda ser leída.
- **Capas ocultas:** capas intermedias, encargadas de extraer, procesar y memorizar la información.

Clasificación de las RNA en función de cómo se interconectan unas capas con otras:

- **Redes en cascada:** La información fluye unidireccionalmente de una capa a otra, no se admiten conexiones intracapa.(conexiones laterales)
- **Redes recurrentes:** La información puede volver a lugares por los que ya había pasado, formando bucles, y se admiten las conexiones intracapa, incluso consigo misma. (conexiones laterales).

Las **conexiones** entre una capa y otra pueden ser:

- **Totales:** cada unidad se conecta con todas las unidades de la capa siguiente.
- **Parciales:** una unidad se conecta con solo algunas de las capas de la unidad siguiente.

Desde una **aproximación temporal** se puede distinguir entre:

- **Conexiones sin retardo**
- **Conexiones con retardo**

Características de las RNA

1. **Aprendizaje inductivo:** no se le indican las reglas, extrae sus propias reglas a partir de los ejemplos de aprendizaje, modifican su comportamiento en función de la experiencia. Estas reglas quedan almacenadas en las conexiones y no representadas explícitamente como en los sistemas basados en conocimiento.

2. **Generalización:** una vez entrenada, se le puede presentar a la red datos distintos a los usados durante el aprendizaje.

3. **Abstracción o tolerancia al ruido:** las RNA son capaces de extraer o abstraer las características esenciales de las entradas aprendidas, de esta manera pueden procesar correctamente datos incompletos o distorsionados.
4. **Procesamiento paralelo:** las neuronas reales trabajan en paralelo mediante multiprocesadores.
5. **Memoria distribuida:** el conocimiento acumulado por la red se halla distribuido en numerosas conexiones, esto tiene como consecuencia la tolerancia a fallos.

Ventajas y Desventajas de las Redes Neuronales

Ventajas que ofrecen las RNA

1. **Aprendizaje adaptativo:** capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial.
Pueden aprender mediante ejemplos y entrenamiento, no es necesario elaborar modelos a priori, ni especificar funciones de distribución de probabilidad. Las redes neuronales son **sistemas dinámicos autoadaptativos**.
 - Son adaptables debido a la capacidad de autoajuste de los elementos procesales.
 - Son dinámicos, pues son capaces de estar constantemente cambiando para adaptarse a las nuevas condiciones.
 Una red neuronal no necesita un algoritmo para resolver un problema, ya que ella puede generar su propia distribución de pesos en los enlaces mediante el aprendizaje. La función del diseñador es únicamente la obtención de la arquitectura apropiada. Es necesario que desarrolle un buen algoritmo de aprendizaje que le proporcione a la red la capacidad de discriminar, mediante un entrenamiento con patrones.
2. **Auto-organización:** puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.
La autoorganización consiste en la modificación de la red neuronal completa para llevar a cabo un objetivo específico. Esta autoorganización provoca la generalización: facultad de las redes neuronales de responder apropiadamente cuando se les presentan datos o situaciones a las que no había sido expuesta anteriormente.
3. **Tolerancia a fallos:**
Si se produce un fallo en un número no muy grande de neuronas y aunque el comportamiento del sistema se ve influenciado, no sufre una caída repentina.
Aspectos respecto a la tolerancia a fallos:
 - las redes pueden aprender a reconocer patrones con ruido, distorsionados o incompletos. Es una tolerancia respecto a los datos.
 - Las redes pueden seguir realizando su función aunque se destruya parte de la red.
 La razón de esta tolerancia es que tienen su información distribuida en las conexiones entre neuronas, existiendo cierto grado de redundancia en este tipo de almacenamiento.
4. **Operación en tiempo real:**
Necesidad de realizar los procesos con datos de forma muy rápida.
Las redes neuronales se adaptan bien a esto debido a su implementación paralela.
5. **Fácil inserción dentro de la tecnología existente.**

Desventajas que ofrecen las RNA

1. **Definición de muchos parámetros:** Antes de aplicar la metodología. Por ejemplo hay que decidir: la arquitectura más apropiada, el nro de capas ocultas, el nro de nodos por capa, interconexiones, función de transformación, etc.
2. **Caja negra:** No ofrecen una interpretación fácil. En lugar de ser un sistema de apoyo a la decisión, la caja negra se puede convertir en el "tomador" de la decisión

Mecanismos de Aprendizaje

El aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada.

Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones, se puede afirmar que este proceso se ha terminado cuando los valores de los pesos permanecen estables.

Criterios para cambiar el valor de los pesos:

- **Aprendizaje supervisado**
- **Aprendizaje no supervisado**

Otro criterio que se usa para diferenciar las reglas de aprendizaje se basa en:

- **Aprendizaje on-line:** si la red puede aprender durante su funcionamiento habitual.
- **Aprendizaje off-line:** Si el aprendizaje supone la desconexión de la red, es decir, su inhabilitación hasta que el proceso se termine.

Aprendizaje supervisado

Se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo que determina la respuesta que debería generar la red a partir de una entrada determinada.

El supervisor controla la salida de la red y en caso que ésta no coincida con la deseada, se procederá a modificar los pesos de las conexiones.

Se consideran 3 formas de llevarlo a cabo:

A. Aprendizaje por corrección de error

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos a la salida de la red.

Error cometido = Valor deseado – Valor obtenido

B. Aprendizaje por refuerzo

La función del supervisor se reduce a indicar mediante una señal de refuerzo si la salida obtenida en la red se ajusta a la deseada y en función de ello se ajustan los pesos basándose en un mecanismo de probabilidades.

C. Aprendizaje estocástico

Consiste en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

Se realiza un cambio aleatorio de los valores de los pesos y se determina la energía de la red. Si es menor después del cambio, es decir el comportamiento de la red se acerca al deseado, se acepta el cambio. En caso contrario, se aceptaría el cambio en función de una determinada y preestablecida distribución de probabilidad.

Aprendizaje No supervisado

Las redes con aprendizaje no supervisado no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta.

Existen varias interpretaciones de la salida de estas redes, que dependen de su estructura y del algoritmo de aprendizaje empleado:

- La salida representa el grado de similitud entre la información que se le está presentando en la entrada y las informaciones que se le han mostrado hasta entonces.
- Podría realizar un establecimiento de categorías, indicando la red a la salida, a qué categoría pertenece la información presentada a la entrada, siendo la propia red quien debe encontrar las categorías apropiadas a partir de las correlaciones entre las informaciones presentadas.

Se consideran dos tipos de algoritmos de aprendizaje no supervisado:

A. Aprendizaje hebbiano

Los sistemas neuronales biológicos no nacen preprogramados con todo el conocimiento y las capacidades que llegarán a tener eventualmente.

Un proceso de aprendizaje que tiene lugar a lo largo de un período de tiempo modifica de alguna forma la red para incluir la nueva información.

La suposición de Hebb determina que se produce algún cambio entre A y B, de tal modo que la influencia de A sobre B se ve incrementada. Si el experimento se repite con frecuencia suficiente, A será capaz de lograr eventualmente, que se dispare B incluso en ausencia de la estimulación visual procedente de B.

Llevado esto al terreno de las RNA, significa que el peso de la conexión entre ambas neuronas se ve incrementado.

B. Aprendizaje competitivo y comparativo

El aprendizaje competitivo se orienta a la clasificación de los datos de entrada. En este tipo de aprendizaje, las unidades de salida luchan por el control sobre porciones del espacio de entrada.

Las unidades de entrada se conectan directamente a las unidades de salida, pero éstas también se conectan entre sí con conexiones precableadas negativas o inhibitorias.

La unidad de salida con la mayor activación en sus entradas tendrá más fuerzas que sus competidores. Como resultado, los competidores se vuelven más débiles.

El perceptrón

Fue uno de los primeros modelos de redes neuronales.

Un perceptrón imita una neurona tomando la suma ponderada de sus entradas y enviando a la salida un 1 si la suma es más grande que algún valor umbral ajustable, y enviando un 0 en caso contrario.

Un perceptrón es una representación, una red neuronal en la que:

- Sólo hay una neurona.
- Las entradas son binarias.
- Las cajas lógicas pueden interponerse entre las entradas y los pesos del perceptrón.
- La salida del perceptrón es 0 o 1.

Las entradas son 0 o 1, al igual que las salidas de las cajas lógicas. Si la suma de las salidas ponderadas de las cajas lógicas es mayor que 0, la salida del perceptrón es 1 y se dice que el perceptrón dice SI, se ha reconocido una clase. En cualquier otro caso, se dice que el perceptrón dice NO, no se ha reconocido una clase.

En un **perceptrón limitado por el orden**, de orden n , cada caja lógica atiende a n o menos entradas.

En un **perceptrón directo**, cada caja lógica tiene solo una entrada y la salida es siempre la misma que la entrada.

Aprendizaje del perceptrón

Existe un procedimiento que descubre un buen conjunto de pesos para un perceptrón, dado que tal conjunto exista.

Se empieza con todos los pesos en 0. Después se intenta el perceptrón con todas las muestras. Siempre que el perceptrón cometa un error, se cambian los pesos de modo que el error se haga menos probable, en cualquier otro caso, no se hace nada.

- Si el perceptrón dice no, al producir un 0, cuando debería decir que sí, produciendo un 1. Se incrementan los pesos asignados a aquellas cajas lógicas que producen 1 en el instante en que se produce el error.
- Si dice SI cuando debería decir NO, se disminuyen los pesos a las cajas lógicas que producen 1, con la disminución igual a 1.
- No se alteran los pesos asignados a las cajas que producen 0.

Para conseguir el efecto de un umbral entrenable, se añade una entrada virtual extra, cuyo valor siempre se supone es 1. Con esta añadidura, el perceptrón se puede ver como si estuviera un umbral de 0 y dice sí siempre que la suma ponderada de las salidas de la caja lógica sea mayor que 0.

Para entrenar a un perceptrón:

- Hasta que el perceptrón produzca el resultado correcto para cada muestra de entrenamiento, para cada muestra:

- Si el perceptrón produce una respuesta errónea:
 - Si el perceptrón dice NO cuando debería decir SI, añada el vector de salida de caja lógica al vector peso.
 - En cualquier otro caso, reste el vector de salida de caja lógica del vector peso.
- En cualquier otro caso, no haga nada.

La separación lineal y el problema del XOR

El teorema de convergencia del perceptrón garantiza que el perceptrón encontrará un estado solución, es decir, aprenderá a clasificar cualquier conjunto de entradas linealmente separables.

Mientras el teorema de convergencia garantizaba una correcta clasificación de la información linealmente independiente, la mayoría de los problemas no proporcionaban este tipo de datos tan bellos.

El perceptrón es incapaz de aprender a resolver algunos problemas sencillos. Un ejemplo es el problema del OR-Exclusiva (XOR).

No hay una única recta que pueda separar las salidas 1 de las salidas 0.

La deficiencia no está en el algoritmo de aprendizaje del perceptrón, sino en el modo en el que el perceptrón representa el conocimiento.

Redes de Hopfield

Hopfield introdujo una red neuronal que propuso como una nueva teoría de la memoria. Tiene importantes características:

1. **Representación distribuida:** una memoria se almacena como un patrón de activación a través de un conjunto de elementos de activación. Las memorias pueden estar superpuestas una sobre otra. Las diferentes memorias se representan por diferentes patrones sobre el mismo conjunto de elementos de proceso.
2. **Control asíncrono y distribuido:** cada elemento de proceso toma decisiones basadas en su propia situación local. Todas las situaciones locales se unen para alcanzar una solución global.
3. **Memoria direccionable por contenido:** se puede almacenar un determinado número de patrones en una red. Para recuperar un patrón se necesita una parte específica de él.
4. **Tolerancia a fallos:** aunque elementos procesadores de la red fallen, funcionará adecuadamente.

Los elementos de proceso se llaman **unidades**, siempre se encuentran en uno de los dos estados posibles, activos o inactivos. Las unidades están conectadas unas con otras por conexiones simétricas y con pesos.

Una conexión con **peso positivo** indica que las dos unidades tienden a activarse la una a la otra, con peso **negativo** permite que una unidad activa desactive a su unidad vecina.

La red funciona del siguiente modo:

Se elige una unidad aleatoriamente. Si alguna de sus vecinas está activada, la unidad calcula la suma de los pesos en las conexiones de esas unidades. Si la suma es positiva, la unidad se activa y si ocurre de modo contrario, se desactiva.

Se elige otra unidad aleatoriamente y se repite el proceso hasta que la red alcanza un **estado estable**. Este proceso se denomina **relajación paralela**.

La mayor contribución de las redes de Hopfield es la de mostrar que dado un conjunto de pesos y un estado inicial, su algoritmo de relajación paralela en algún momento llevará a la red hacia un estado estable. No puede no existir divergencia u oscilación.

Tenemos un comportamiento corrector de errores. Aún cuando en el estado inicial existan inconsistencias, la red de Hopfield se asentará en la solución que viole el menor número posible de restricciones que ofrecen las entradas.

Máquinas de Boltzman

Es una variación de una red de Hopfield.

El primer problema de las redes de Hopfield es que convergen hacia mínimos locales.

Tener muchos mínimos locales es adecuado para construir una memoria direccionable por contenido, pero para las tareas de verificación de restricciones es necesario encontrar el estado globalmente óptimo de la red.

Las redes de Hopfield no pueden encontrar soluciones globales porque se sitúan en los estados estables por medio de un algoritmo completamente distribuido.

El **enfriamiento simulado** es una técnica para encontrar soluciones globalmente óptimas en problemas combinatorios.

Combinaron las redes de Hopfield y el enfriamiento simulado para producir redes denominadas **máquinas de Boltzmann**.

Las unidades de la máquina de Boltzmann actualizan sus estados binarios individuales mediante una regla estocástica.

La probabilidad de que una determinada unidad se active viene dada por p :

Donde ΔE es la suma de las líneas de entrada activas de las unidades y T es la temperatura de la red.

A altas temperaturas, las unidades presentan un comportamiento aleatorio, y a bajas, las unidades se comportan como redes de Hopfield.

El enfriamiento es el proceso de ir gradualmente de altas a bajas temperaturas. La aleatoriedad que añade la temperatura ayuda a la red a escapar de los mínimos locales.

Si el enfriamiento se produce de forma adecuada, las máquinas de Boltzmann podrían evitar los mínimos locales y aprender a calcular cualquier función calculable de entradas y salidas de tamaño fijo.

Redes Recurrentes

Una deficiencia de los modelos de redes neuronales en comparación con los modelos simbólicos es la dificultad en conseguir modelos de redes neuronales que traten tareas de IA temporales como la planificación y el análisis del lenguaje natural.

Las redes recurrentes constituyen un intento de remediar esta situación.

La red nunca converge hacia un estado estable. En lugar de esto, cambia en cada paso de tiempo.

Estas redes se pueden entrenar mediante el algoritmo de propagación hacia atrás.

En cada paso se comparan las activaciones de las unidades de salida con las activaciones deseadas y se propagan los errores hacia atrás por la red.

Cuando se completa el entrenamiento, la red podrá llevar a cabo una secuencia de acciones.

Unidad 1: Introducción a la inteligencia artificial

1.1 ¿Qué es la inteligencia artificial?

- 1.1.1 Pueden
- 1.1.2 Máquinas
- 1.1.3 Pensar

1.2 Aproximaciones a la IA

- 1.2.1 Basados en procesamiento de símbolos
- 1.2.2 Aproximaciones subsimbólicas

1.3 Breve historia de la IA

1.4 Mundo especial para agentes con IA

1.5 Sistemas que resuelven problemas de la IA

- 1.5.1 Acciones que deben llevar a cabo el sistema
- 1.5.2 Definición mediante búsqueda en espacio de estados
- 1.5.3 Sistemas de producción
- 1.5.4 Análisis del problema
- 1.5.5 Características de los sistemas de producción

Unidad 2: Búsqueda y planificación

2.1 Técnicas de búsqueda a ciegas

- 2.1.1 Explosión combinatoria
- 2.1.2 Ramificación y acotación
- 2.1.3 Búsqueda primero en anchura
- 2.1.4 Búsqueda primero en profundidad

2.2 Técnicas de búsqueda heurística

- 2.2.1 Generación y prueba
- 2.2.2 Escalada o remonte de colinas
- 2.2.3 Búsqueda El primero mejor
- 2.2.4 Reducción de problemas
- 2.2.5 Verificación de restricciones
- 2.2.6 Análisis de medios y fines

2.3 Búsqueda en problemas de juego

- 2.3.1 Juegos de dos jugadores
- 2.3.2 El procedimiento minimax
- 2.3.3 El procedimiento alfa-beta

2.4 Búsqueda con sistemas evolutivos

- 2.4.1 Conceptos de genética
- 2.4.2 Algoritmos genéticos
- 2.4.3 Poblaciones
- 2.4.4 Operadores genéticos
- 2.4.5 Función de evaluación

2.5 Planificación

- 2.5.1 Introducción
- 2.5.2 Componentes de un sistema de planificación
- 2.5.3 Planificación mediante una pila de objetivos
- 2.5.4 Planificación no lineal mediante fijación de restricciones
- 2.5.5 Planificación jerárquica
- 2.5.6 Sistemas reactivos

Unidad 3: Representación del Conocimiento y Razonamiento

3.1 El problema de la Representación del Conocimiento

- 3.1.1 Correspondencia entre Conocimiento y RC
- 3.1.2 Propiedades de un buen sistema de RC
- 3.1.3 Modelos de Representación del Conocimiento
 - 3.1.3.1 Conocimiento relacional simple
 - 3.1.3.2 Conocimiento heredable
 - 3.1.3.3 Conocimiento deductivo
 - 3.1.3.4 Conocimiento procedimental
- 3.1.4 Problemas de la Representación del Conocimiento
- 3.1.5 El problema del Marco

3.2 Lógica simbólica

- 3.2.1 La lógica y el lenguaje
 - 3.2.1.1 Introducción
 - 3.2.1.2 Naturaleza del argumento
 - 3.2.1.3 Verdad y validez
 - 3.2.1.4 Lógica simbólica
- 3.2.2 Argumentos que contienen Enunciados compuestos
 - 3.2.2.1 Enunciados simples y compuestos
 - 3.2.2.2 Enunciados condicionales
 - 3.2.2.3 Formas de argumentos y Tablas de verdad
 - 3.2.2.4 Formas sentenciales
- 3.2.3 El método de deducción
 - 3.2.3.1 Prueba formal de validez
 - 3.2.3.2 La regla de reemplazo
 - 3.2.3.3 Demostración de la invalidez

3.3 Lógica de predicados

- 3.3.1 Introducción y concepto de Semidecidible
- 3.3.2 Representación de hechos simples en lógica
- 3.3.3 La representación de las relaciones instancia y es-un
- 3.3.4 Representación de Funciones calculables y Predicados computables
- 3.3.5 Método de Resolución
- 3.3.6 Conversión a forma clausal

- 3.3.7 Las bases de resolución
- 3.3.8 Resolución en lógica proposicional
- 3.3.9 El algoritmo de unificación
- 3.3.10 Resolución en lógica de predicados

3.4 Representación del conocimiento mediante Reglas

- 3.4.1 Comparación entre conocimiento procedimental y conocimiento declarativo
- 3.4.2 Programación lógica
- 3.4.3 Diferencia entre Razonamientos hacia delante y hacia atrás
 - 3.4.3.1 Sistemas de reglas encadenadas hacia atrás
 - 3.4.3.2 Sistemas de reglas encadenadas hacia atrás
 - 3.4.3.3 Combinación del razonamiento hacia delante y hacia atrás

3.5 Razonamiento simbólico bajo incertidumbre

- 3.5.1 Razonamiento No Monótono
 - 3.5.1.1 Razonamiento por defecto
 - 3.5.1.1.1 Lógica no monótona
 - 3.5.1.1.2 Lógica por defecto
 - 3.5.1.2 Razonamiento minimalista
 - 3.5.1.2.1 La suposición de un mundo cerrado
 - 3.5.1.2.2 Circunscripción
 - 3.5.1.3 Cuestiones sobre la implementación
 - 3.5.1.3.1 Implementación búsqueda primero en profundidad
 - 3.5.1.3.2 Implementación búsqueda primero en anchura
- 3.5.2 Razonamiento estadístico
 - 3.5.2.1 Factores de certeza y sistemas basados en reglas
 - 3.5.2.2 Redes Bayesianas
 - 3.5.2.3 Teoría de Dempster-Shafer

3.6 Estructuras de Ranura y Relleno Débiles

- 3.6.1 Redes semánticas
- 3.6.2 Marcos

3.7 Estructuras de Ranura y Relleno Fuertes

- 3.7.1 Dependencia conceptual
- 3.7.2 Guiones
- 3.7.3 CYC

Unidad 4: Ingeniería del Conocimiento

4.1 PROLOG (Programación Lógica)

- 4.1.1 Una breve historia del Prolog
- 4.1.2 ¿Para qué sirve Prolog?
- 4.1.3 Lenguaje Procedural vs. Declarativo

4.2 Relación con la Lógica

- 4.2.1 Hechos
- 4.2.2 Variables
- 4.2.3 Reglas
- 4.2.4 Cláusulas
- 4.2.5 Preguntas
- 4.2.6 Conjunciones y Backtracking

4.3 Estructura de un Programa en PROLOG

- 4.3.1 Composición de un programa. Cláusulas. Predicados. Dominios.
- 4.3.2 Ejemplos

4.4 Ingeniería del Conocimiento

- 4.4.1 Organización de una Base de conocimiento
- 4.4.2 Encontrando el Experto
- 4.4.3 Verificaciones de la BC

4.5 Definición de Sistema Experto

- 4.5.1 Definición funcional
- 4.5.2 Definición estructural

4.6 Armazones de Sistemas Experto

4.7 Aplicaciones de los Sistemas Expertos

- Ventajas de la aplicación de sistemas expertos
- Fases de la "inserción social" de los Sistemas Expertos

4.8 Sistemas Expertos más conocidos

Unidad 5: Ingeniería del Conocimiento

5.1 Modelos Conexionistas

- 5.1.1 Origen del paradigma de computación conexionista

5.2 Redes Neuronales Biológicas

5.3 Redes Neuronales Artificiales

- 5.3.1 Definición de Red Neuronal
- 5.3.2 Estructura de las Redes Neuronales
- 5.3.3 Comparación entre RNB y RNA
- 5.3.4 Formas de interconexión de las RNA
- 5.3.5 Características de las RNA

5.4 Ventajas y Desventajas de las Redes Neuronales

- 5.4.1 Ventajas que ofrecen las RNA
- 5.4.2 Desventajas que ofrecen las RNA

5.5 Mecanismos de Aprendizaje

- 5.5.1 Aprendizaje supervisado

5.5.2 Aprendizaje No supervisado

5.6 El perceptrón

5.6.1 Aprendizaje del perceptrón

5.6.2 La separación lineal y el problema del XOR

5.7 Redes de Hopfield

5.8 Máquinas de Boltzman

5.9 Redes Recurrentes