SISTEMAS EXPERTOS

¿Qué son los sistemas expertos?

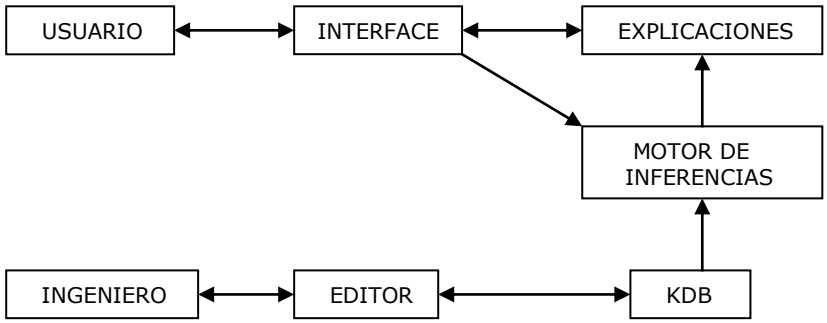
Un sistema experto es un programa destinado a generar inferencias en un área específica del conocimiento de una forma similar a la que se espera de un experto humano.

Los sistemas expertos deben resolver problemas por aplicación de conocimiento de un dominio específico. Este conocimiento es adquirido a través de la intervención de expertos humanos y almacenado en lo que se denomina *Base de datos de conocimiento*.

Se espera que un sistema experto sea capaz de explicar la solución hallada de forma comprensible para las personas. Además no pueden operar en situaciones de sentido común por ser muy extenso el dominio de conocimiento que debe tener el sistema.

Algunas características del tipo de problema resuelto por un sistema experto son:
Inexistencia de un algoritmo para hallar la solución.
El problema es resuelto satisfactoriamente por expertos humanos.
Posibilidad de contar con un experto humano para colaborar en el desarrollo.
El conocimiento experto del dominio debe ser relativamente estático.

El núcleo del SE es el Motor de Inferencias. Es modulo interactúa con los demás.



-Arquitectura básica de un SE-

Componentes básicos de un SE

Símbolos: El empleo de símbolos de significado preciso es el primer componente de un SE.

Reglas: Un segundo componente de los SE son las reglas a aplicar sobre los símbolos definidos. Estas se deben obtener del conocimiento de los expertos humanos. Las mismas se combinan luego en un sistema de inferencias. Estas reglas de los expertos son consideradas heurísticas, dado que no es posible demostrar su validez general.

Hechos: Los hechos son los predicados que se suponen verdaderos; estos constituyen la Base de Datos de Conocimiento (KDB).

Fundamentos de los sistemas expertos

Los SE clásicos se basan en la *lógica de predicados*, la cual a diferencia de la lógica proposicional puede emplear variables.

Para que la lógica de primer orden nos garantice inferencias validas, el significado de los símbolos a emplear debe ser preciso, o dicho de otro modo no deben existir ambigüedades respecto del posible significado de un símbolo.

En el caso de los SE se pretende aplicar la lógica a mundos que solo conocen en profundidad los expertos humanos.

Para que exista alguna chance de que un SE basado en el formalismo lógico produzca inferencias correctas, se requiere en primer lugar definir los símbolos (palabras y predicados) con los cuales se operará y su significado preciso. Por otra parte, quien implemente el SE, debe traducir los conocimientos expresados en lenguaje natural por el experto humano a las expresiones lógicas correspondientes, para que estas puedan ser procesadas por el SE.

Solo se intenta aplicar SE a mundos reducidos.

Resumiendo, los SE se basan en la posibilidad de aplicar las nociones de lógica formal para incrementar una base de conocimientos o demostrar hipótesis, y solo pueden aplicarse prácticamente a mundos de muy baja complejidad, en el sentido de la cantidad de símbolos y predicados que deben manejarse.

El modo más inmediato consiste en aplicar reglas del tipo si-entonces, e instanciando los antecedentes de la misma obtener una instancia válida para el consecuente.

Es importante notar que en lugar de instanciar el consecuente para ver si es soportado por sus antecedentes se podría trabajar a la inversa, esto da lugar a dos modelos fundamentales de sistemas expertos:

1. Con encadenamiento hacia atrás: Se toma la serie de afirmaciones de entrada y ensaya todas las reglas disponibles una y otra vez, incorporando nuevas afirmaciones, hasta que ninguna de las reglas pueda producir nuevas afirmaciones. Puede emplearse para verificar una hipótesis obteniendo una afirmación que concuerde con ella.

2. Con encadenamiento hacia atrás: Se comienza con una hipótesis y se trata de verificarla haciendo uso de las afirmaciones disponibles.

Existen dos lenguajes muy difundidos orientados a este tipo de procesamiento, Lisp y Prolog. Este último fue desarrollado a partir del primero y se orienta específicamente a sistemas expertos de encadenamiento hacia tras.

Lógica de predicados y sistemas expertos

Lógica

La lógica hace referencia a la posibilidad de emplear un conjunto de reglas para gestionar inferencias creíbles. Podemos tener una lógica dicotómica (verdadero o falso) o una lógica multivaluada. De ambas técnicas se derivan esquemas de procesamiento para modelar los procesos cognitivos humanos en la computadora.

El significado de la palabra *inferencia* es un proceso mental por el cual se extraen conclusiones a partir de premisas más o menos explicitas. Puede ser resultado del sentido común, o de silogismos formales e informales, o bien por aplicación de reglas muy detallistas o cálculos de la inferencia estadística.

La epistemología es el estudio del origen y de las características del conocimiento, por lo que se debe tener en cuenta a la hora de discutir métodos de inferencia. Estudia el conocimiento, en especial la forma cómo aparece y sus limitaciones.

Los formalismos a usar para representar los hechos que ocurren en el mundo deben ser lo suficientemente adecuados como para representar la información disponible.

Finalmente se entiende por silogismo a una fórmula lógica con premisas seguida de una conclusión.

La lógica de primer orden es uno de los formalismos más usado para representar conocimiento. La lógica cuenta con un lenguaje formal mediante el cual es posible representar formulas llamadas axiomas que permiten describir fragmentos del conocimiento. También cuenta con reglas de inferencia que se aplican a los axiomas para derivar nuevo conocimiento.

El alfabeto del lenguaje de la lógica de primer orden contiene dos tipos de símbolos:

1. Símbolos lógicos: Entre los que se encuentran los símbolos de constantes proposicionales true y false; los símbolos de operadores proposicionales para la negación, la conjunción, la disyunción y las implicaciones; los símbolos de operadores de cuantificación como el cuantificador universal; el cuantificador existencial; y los símbolos auxiliares de escritura como los corchetes, paréntesis y coma.

2. Símbolos no lógicos: Agrupados en el conjunto de símbolos constantes, el conjunto de símbolos de variables individuales, el conjunto de símbolos de funciones n-arias y el conjunto de símbolos de relaciones n-arias.

A partir de estos símbolos se construyen las expresiones validas en el lenguaje de primer orden: los términos y fórmulas.

1. Términos: Una constante, una variable o una expresión de la forma f(t1, …, tn) donde f es un símbolo de función n-aria y t1, …, tn son términos. Permiten nombrar los objetos del universo.

2. Fórmulas: Un ejemplo de funciones son f(100, X), Padre(X) y Sucesor(X). Las formulas atómicas o elementales son expresiones de la forma R(t1, …, tn) donde R es un símbolo de relación n-aria t t1, …, tn son términos. Permiten afirmar o negar propiedades de los objetos del universo.

La **lógica proposicional** se expresa mediante clausulas del tipo (a y b o c) donde los literales solo pueden ser ciertos o falsos y no son funciones de otros valores. La **lógica de predicados** se expresa mediante expresiones de la forma $\forall\, x\, \exists\, y\, /\, x\, < y$.

Fórmulas bien formadas

La lógica proposicional utilizando una representación primitiva del lenguaje permite representar y manipular aserciones sobre el mundo que nos rodea. Permite el razonamiento, a través de un mecanismo que primero evalúa sentencias simples y luego sentencias complejas formadas mediante el uso de conectivos proposicionales, por ejemplo, *and* y *or*. Este mecanismo determina la veracidad de una sentencia compleja analizando los valores de veracidad asignados a las sentencias simples que la conforman. Una proposición es una sentencia simple que tiene un valor asociado: verdadero o falso. Por ejemplo: Hoy es viernes.

La lógica proposicional no analiza las palabras individuales que componen la sentencia, por lo cual el ejemplo anterior sería hoy_es_viernes.

Una fórmula bien formada puede ser una proposición simple o compuesta que tiene sentido completo, y cuyo valor de veracidad puede ser determinado. La lógica proposicional proporciona un mecanismo para asignar valores de veracidad a la proposición compuesta, basado en los valores de veracidad de las proposiciones simples y en la naturaleza de los conectores lógicos involucrados.

Reglas de inferencia mas empleadas en SE

Silogismo categórico (Aristóteles): El silogismo categórico dice que si es verdadera una sentencia cualquiera de las siguientes dos premisas:

1. Todo M es A.
2. Todo B es M.

Entonces la conclusión es válida: Todo B es A.

Resolución (Aristóteles): La resolución se basa en que una afirmación no puede ser simultáneamente verdadera y falsa, lo que se emplea para inferir la validez de otras afirmaciones.

Silogismos hipotéticos: Los silogismos hipotéticos son menos empelados, y es una regla de inferencia para probar la corrección o valor de verdad de proposiciones lógicas. Sean dos afirmaciones cualesquiera, A y B, se sabe que si A es verdad B también es verdad; además se sabe que A es verdad; con lo que se infiere que B es verdad. Simbólicamente: A => B, A.

De forma más explícita, se presenta una premisa 1 que es SI A ENTONCES B acompañada de una premisa 2 que es A.

Elementos básicos del lenguaje lógico

La aplicación del paradigma lógico requiere la definición de un lenguaje acotado y preciso mediante el cual se expresen con precisión las reglas y sus resultados. Este lenguaje incluye los siguientes componentes principales:

Átomos: Son propiciaciones simples: verdadero, falso, y símbolos en general.

Conectivos lógicos: Conjunción, disyunción, implicación, doble implicación, y negación. Son expresiones que sirven para formar proposiciones compuestas a partir de proposiciones simples.

Sentencias: Átomos o clausulas formadas por la aplicación de conectivos a sentencias. Son pensamientos en los que se afirma algo y se expresan mediante enunciados u oraciones declarativas. Dichas proposiciones están unidas mediante conectivos lógicos.

Cuantificadores: Cuantificador universal (∀), que significa que todas las variables son comunes a todos los elementos de la expresión, y cuantificador existencial (∃), que se debe eliminar porque las clausulas debe cumplirse para todos los elementos y las variables deben ser comunes a toda las expresión.

Símbolos de puntuación y delimitación: Tales como: , () []

Silogismos: Los mas empleados son el hipotético, A es B y B es C entonces A es C, y disyuntivo.

Funciones de Skolem

Cuando un cuantificador existencial esta en el ámbito de un cuantificador universal la variable cuantificada debe ser reemplazada por una función de Skolem de las variables cuantificadas universalmente.

$\forall x \exists y / x < y \rightarrow \forall x / x < x + 1$

Mientras que en la fórmula $\forall y \exists x / x < y$ la elección del valor de x es dependiente del valor de y, ya que dice que para cada y hay un valor de x. En este caso resulta $\forall y / y - 1 < y$.

En cualquier caso la elección del valor de y es independiente de la elección del valor para x.

Leyes de De Morgan

–(A ^ B) es equivalente a (–A v –B)
–(A v B) es equivalente a (–A ^ –B)

Es decir, al invertir el valor de verdad de los símbolos de la expresión debe “invertirse” el conectivo.

Formas clausales

Toda *fórmula bien formada* o sentencia de lógica proposicional puede expresarse en Forma Normalizada Conjuntiva, Forma Normalizada Disyuntiva o como Cláusula de Horn. Estos modelos para expresar las *fórmulas bien formadas* se destacan de otros existentes en relación a su aplicación a los SE.

Forma Normalizada Conjuntiva: (A v –B) ^ (B v C) ^ A. Conjunción de disyunciones de literales.

Forma Normalizada Disyuntiva: (–P ^ – Q) v (–Q ^ R) v P. Disyunción de conjunciones de literales.

Cláusula de Horn: La cláusula contiene como máximo un literal positivo.

Clausulas de Horn

Una cláusula de Horn es una conjunción de disyunciones con no más de un literal positivo. Entendiendo como un literal positivo a uno que no esté negado. Por ejemplo: (–A1 v A2) ^ (–B1 v –B2 v –B3).

Estas cláusulas de Horn representan la forma implicativa escrita de otro modo, es decir, la implicación B → A equivale a –B v A y la implicación B1 ^ B2 → A equivale a –B1 v –B2 v A.

Para aplicar la lógica en programas de computadora ha sido necesario encontrar algoritmos que determinen si una expresión lógica válida se hace verdadera con alguna combinación de valores verdaderos o falsos de sus átomos. Un algoritmo conocido pero de altísimo costo es construir la tabla de verdad de la expresión y así determinar si alguna combinación hace verdadera la expresión. Un algoritmo más rápido puede aplicarse si la expresión está en la forma de cláusula de Horn, y aquí radica su importancia.

Reducción a la forma clausal

Existe un método que permite reducir reglas expresadas en lógica de predicados a un conjunto de cláusulas (afirmaciones) de modo que las mismas puedan ser procesadas por un mecanismo lógico, que luego se codificara en un programa.

Este algoritmo parte de las llamadas *fórmulas bien formadas* y reduce estas a la denominada *forma clausal* que es concretamente la expresión original transformada a un conjunto de cláusulas simples. Esto implica que una expresión en lenguaje natural solo puede ser procesada si es evaluada como fórmula bien formada o si puede expresare en esta forma.

El algoritmo de reducción a la forma clausal es el siguiente:

1. Eliminación: Eliminar las implicaciones dado que: $x \rightarrow y \equiv -x \; v \; y$

2. Reducir la aplicación de la negación haciendo uso de De Morgan y de doble negación:

$-(-x) \equiv x$
$-(x \; v \; y) \equiv -x \; ^{\wedge} \; -y$
$-(x \; ^{\wedge} \; y) \equiv -x \; v \; -y$

3. Eliminar cuantificadores

4. Aplicar la propiedad distributiva y asociativa creando una cláusula separa por cada conjunto:

$P \; v \; (Q \; v \; R) = (P \; v \; Q) \; v \; R$
$(P \; ^{\wedge} \; Q) \; v \; R \; = (P \; v \; R) \; ^{\wedge} \; (Q \; v \; R)$

5. Que cada clausula tenga nombre de variable único.

Resolución

La resolución es una regla que se aplica sobre cierto tipo de fórmulas del Cálculo de Predicados de Primer Orden, llamadas cláusulas. El principio de resolución es utilizado por el intérprete de Prolog para dar respuesta a los objetivos planteados.

Las deducciones con cláusulas generales tienen un problema que es el gran número de combinaciones posibles para llevar a cabo las resoluciones. Por ello Prolog restringe el número de cláusulas, lo que le permite llevar a cabo una prueba dirigida y en la mayoría de los casos con un universo de posibilidades explorable en tiempo de ejecución.

El procedimiento de resolución es un proceso iterativo cuya finalidad es evaluar la verdad o falsedad de una premisa. En cada paso dos clausulas padres son resueltas para obtener una tercera denominada resolvente; el resolvente es

inferido a partir de las cláusulas padres. Tomando como ejemplo las cláusulas padres P v Q y –P v R el resolvente es Q v R.

Para este método se definen las propiedades siguientes:

Para cláusulas P y –P el resolvente es la cláusula vacía.
Si un conjunto de sentencias contiene ambos P y –P el conjunto es insatisfactible.
Si ambos antecedentes son verdaderos luego el resolvente es verdadero.
La resolución verifica la refutación, es decir "si el conjunto de cláusulas implica a P luego es contradictorio con el mismo conjunto que además contenga a -P".

El algoritmo de resolución por refutación se puede sintetizar como una prueba de una proposición P con respecto a un conjunto de axiomas C. A partir de esta definición se establece el algoritmo que comprende los siguientes pasos:

1. Convertir todas las afirmaciones en C a la forma clausal.
2. Negar P y convertir a la forma clausal. Añadir al conjunto C.
3. Repetir lo siguiente hasta que no haya progreso p se detecte una contradicción:
   3.1. Seleccionar dos clausulas (padres).
   3.2. Resolver obteniendo el resolvente.
   3.3. Si el resolvente es la clausula vacía, se detecto una contradicción sino añadirlo al conjunto de cláusulas.

Unificación y ligaduras

Este concepto tiene su origen en la lógica de predicados para permitir razonar dentro de un esquema formal sin relación con la codificación de los programas.

Dado S = {L1, …, Ln} un unificador de F para S es una sustitución tal que L1F = L2F = … = LnF

Por otra parte, estas asociaciones resultantes de la unificación pueden ser sistemáticamente codificadas y la lección que se hace de adoptar una de las representaciones (CHF, DNF O Horn) podrá tener influencia en la estrategia que fundamenta al sistema experto y viceversa.

En el caso del código de un programa este se encarga de reducir las formas clausales a listas asociadas más simples de manipular por el programa. Estas listas son denominadas *ligaduras*. Una lista de ligaduras fluye a través de los filtros del programa, al principio está vacía y al final de este proceso puede ser analizada para extraer de ellas nuevas afirmaciones o demostrar hipótesis. La lista de ligaduras puede crecer o reducirse al pasar por los filtros del programa.

Encadenamiento

<u>Encadenamiento progresivo</u>

Correspondencia. Buscar afirmaciones en la base de datos que correspondan con los antecedentes de una regla.

Resolución. En caso de que se cumplan los antecedentes instanciar el consecuente produciendo una nueva afirmación.

La afirmación es una expresión común sin variables de patrón, mientras que los antecedentes hipótesis pueden tener variables de patrón, por lo que podemos hablar de correspondencia.

<u>Encadenamiento progresivo</u>

Correspondencia. Buscar una afirmación en la base de datos que corresponda a una hipótesis. La afirmación es una expresión común sin variable de patrón mientras que la hipótesis puede tener variables de patrón por lo que podemos hablar de correspondencia.

Unificación. Buscar una regla cuyo consecuente concuerde con la hipótesis. La hipótesis y el consecuente pueden tener variables de patrón, por lo que se habla de unificación.

Resolución. Para cada regla cuyo consecuente concuerde con la hipótesis se intenta verificar de manera recursiva cada antecedente de la regla considerando cada una como una subhipotesis. Si se verifica cada antecedente entonces se ha verificado la hipótesis.

LÓGICA DIFUSA

Los razonamientos concernientes a problemas reales muestran que es posible considerar cierto grado de certeza o falsedad para una afirmación y no valores absolutos de verdad o falsedad.

Para incorporar este concepto a los sistemas de inferencia y sistemas expertos se recurre a modelos probabilísticos, es decir, se asignan valores de probabilidad a las sentencias y se propagan a través de reglas (calcular qué probabilidad se le atribuye al consecuente en función de las probabilidades de los antecedentes).

Existen distintas formas de calcular como influyen las probabilidades de los antecedentes sobre los consecuentes; ello da lugar a distintos esquemas de inferencia en IA: Inferencia Difusa, Inferencia Bayesiana, Propagación de restricciones, etcétera.

Los sistemas basados en lógica difusa se han empleado con éxito en campos tales como control automático, clasificación, SE, soporte de decisión y visión computarizada; los sistemas de inferencia difusa con conocidos bajo muchas denominaciones: SE difuso, modelo difuso, sistema difuso, entre otros.

La solución fuzzy

Se plantea la solución del problema en términos que permitan capturar la esencia del mismo, a través reglas si-entonces y usando oraciones con variables lingüísticas.
Ej.: “Si el servicio es pobre luego la propina es escasa.”
“Si el servicio es excelente o la comida es deliciosa luego la propina es generosa.”
Si damos ahora significado matemático a las variables lingüísticas (que significa servicio “pobre”, comida “deliciosa”, etc.) obtendremos un sistemas de inferencias basado en lógica difusa; sumándole además que hay que determinar el modo en que se combinan las reglas o como se obtiene el significado matemático de la variable lingüística.

Algunas ventajas

La lógica difusa nos permite escribir un código para la solución que resulte más fácil de interpretar para las personas.
Como el sistema difuso está basado en sentencia de “sentido común”, podremos también agregar más sentencias al final de la lista que moldearan más apropiadamente la forma de la función → código de modificación más sencilla.
El mantenimiento del código será más simple debido a que claramente se puede desacoplar en diferentes reglas sencillas.

La esencia de un sistema fuzzy es mapear un espacio de entrada en uno de salida. El mecanismo primario para lograrlo es la inferencia basada en reglas.

Todas las reglas se evalúan en paralelo, su orden no tiene importancia. Estas reglas se refieren a variables y adjetivos que califican a esas variables; por ende, lo primero que se hace al construir un sistema difuso es definir los términos que se planean usar y los adjetivos que los describen.

Entonces la idea tras un sistema de inferencia difusa es interpretar los valores de un vector de entrada y, basado en un conjunto de reglas, asignar valores al vector de salida.

Reglas si-entonces

Los conjuntos difusos y los operadores difusos son los verbos y sujetos de la lógica fuzzy. Las sentencias condicionales, reglas si-luego, son los primeros elementos que permiten sacar provecho de la lógica difusa. Una regla

difusa toma la forma: Si ‘x’ es ‘A’ luego ‘y’ es ‘B’. Donde A y B son valores lingüísticos definidos por conjuntos difusos sobre los rangos X e Y.

La interpretación de estas reglas comprende distintas partes:
1. Evaluación del antecedente –difusión de la entrada y aplicación de operadores difusos-
2. Aplicación del resultado al consecuente –implicación-.
En general se puede expresar que si el antecedente es verdadero en alguna medida de pertenencia, luego el consecuente es también verdadero en el mismo grado.

El antecedente de una regla puede estar compuesto de varias partes; en cuyo caso las partes del antecedente son calculadas simultáneamente y resueltas como un número simple empleando operadores lógicos. El consecuente también puede tener múltiples partes. En cuyo caso todos los consecuentes son afectados igualmente por el resultado del antecedente.

Hay que considerar como afecta el antecedente al consecuente. En lógica difusa el consecuente especifica un conjunto difuso para la salida. La función de implicación modifica luego al conjunto difuso en el grado especificado por el antecedente. Los medios más usuales para modificar el conjunto de salida difuso son el truncamiento (función min) y la escalación (función producto).

Proceso fuzzy

1. Difusión de las entradas - Fuzzificacion

Se deben resolver todas las sentencias en el antecedente en función de su grado de membresía entre 0 y 1. Si el antecedente tiene solo un componente este es el grado de soporte del consecuente.

Tomar las entradas y determinar el grado en el cual ellos pertenecen a cada uno de los conjuntos difusos (adjetivos) a través de funciones de membresía o pertenencia –evaluación de una función-.

La entrada es siempre un valor numérico limitado al universo de discurso de la variable de entrada y la salida es un grado difuso de pertenencia.

2. Aplicación de operadores difusos

Si existen múltiples partes en el antecedente se aplican los operadores fuzzy y se resuelve el antecedente como un número entre 0 y 1. Esto es el grado de soporte de la regla.

En este paso ya se conoce el grado en el cual cada parte del antecedente ha sido satisfecho para cada regla. Pero, como se menciono anteriormente, si el antecedente de una regla ha sido dividido en más de una parte, se debe aplicar un operador difuso para obtener un número que representa el resultado del antecedente para esa regla.
Hay distintos métodos u operadores fuzzy: mínimo, producto, truncamiento; máximo, amplificación y adición.

3. Aplicación de la implicación

Se emplea el grado de soporte de la regla para conformar el conjunto fuzzy de salida. El consecuente de una regla fuzzy asigna un conjunto fuzzy completo a la salida. Si el antecedente es solo parcialmente cierto, luego el conjunto fuzzy de salida es truncado según el método de implicación. Es decir, es la conformación del consecuente (conjunto fuzzy) basado en el antecedente (un número). Se obtiene una salida para cada regla por separado.

4. Agregación

En general se requieren dos o más reglas que interactúen entre ellas. La salida de cada regla es un conjunto fuzzy, pero en general, se necesita que la salida sea en definitiva un número. Para obtener este número, primero los conjuntos fuzzy se salida de cada regla se amalgaman en un único conjunto. Luego este conjunto es defuzzificado, o resuelto de modo de obtener un número como valor de salida.

Este paso entonces, es el proceso de unificar las salidas para cada regla uniendo los procesos paralelos. Se crean un único conjunto difuso a partir de las salidas truncadas o modificadas retornadas por el proceso de implicación para cada regla.

5. Defuzzificacion

La entrada para el esta etapa es el conjunto difuso agregado (que engloba el rango de valores de salida) y la salida es un número. Un popular método de defuzzificacion es el cálculo del centroide, el cual retorna el centro de un área bajo una curva.

La defuzzificacion tiene lugar en dos pasos:
1. Las funciones de pertenencia son escaladas de acuerdo a sus posibles valores.
2. Estas son usadas para calcular el centroide de los conjuntos difusos asociados.

PLANIFICACIÓN

El significado que se le da a planificación en IA se refiere al problema de tener que planificar y ejecutar una secuencia de pasos (operaciones, cambios de estado, etcétera) en un mundo no completamente previsible. Se propusieron un conjunto de "mundos" en los cuales se podían probar mecanismos de planificación. El más famoso es el "Mundo de los Bloques".

Clasificación

El sentido preciso de la palabra planificación varía según el área de trabajo.

Planificación en IA
Numérica.
Lógica.
- Por pila de objetivos.
- No lineal por fijación de restricciones.
- Jerárquica. Se resuelven primero las tareas de mayor complejidad sin considerar detalles.
- Reactiva.

Planificación en Robótica
Reactiva.
Planificación de Trayectorias en manipuladores.
- Método de articulaciones interpoladas.
- Método del espacio cartesiano.

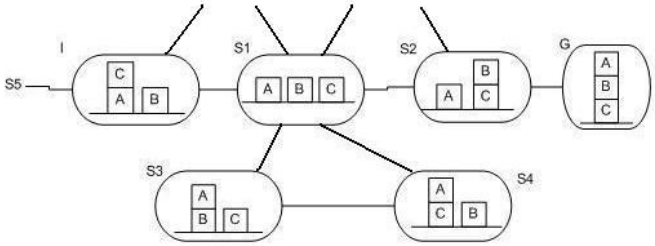
Planificación en Logística Industrial
Métodos basados en Simulación Discreta

El problema de la planificación

Corresponde a la resolución de problemas. Se ocupa de problemas "muy complejos" en el sentido conocido de que no pueden explorarse todas las opciones.

Entradas: Descripción del estado del mundo, Descripción del objetivo y Conjunto de acciones
Salida: Secuencia de acciones que pueden ser aplicadas al estado, hasta alcanzar la descripción del estado final.

Se plantea el problema de la planificación como una búsqueda en un espacio de estados:



La lógica nos ofrece una manera de reducir nuestros pasos en planificación. Se deben representar los estados mediante predicados lo que permite pasar de uno a otro (o saber que es posible pasar de uno a otro) aplicando reglas lógicas, la aplicación de la regla debería modificar el estado anterior, definiendo uno nuevo. Sin embargo la regla en sí, sólo nos dice que está permitido pasar de un estado a otro, pero no efectúa el cambio de predicados para obtener un nuevo estado. Al cambio de predicados le corresponde una acción en el "mundo" en que se planifica.

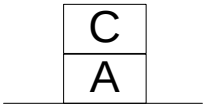
Para que en un programa el estado actual se modifique de acuerdo a la relación establecida en la regla, es necesario aplicar procedimientos (operadores) que efectúen esa modificación. Por lo tanto, además de predicados y reglas se necesitan operadores asociados a las reglas.

Por ejemplo esta regla tiene dos antecedentes que se encuentran entre los predicados que definen el estado actual:

Regla
(Despejado x) ^ (Sobre x, y) →Agarrado (x) ^ Despejado (y)

Predicados (definen el estado actual)
(Sobre C A S)
(Sobrelamesa A S)
(Despejado C S) S= estado

Reglas, operadores y estados
(Despejado x S) ^ (Sobre x, y S) → Agarrado (x, S (Desapilar (x, y))) ^ Despejado (y, S (Desapilar (x, y)))



El mundo de los bloques

Aquí existe una superficie plana sobre la que se sitúan los bloques, los cuales se pueden colocar unos sobre otros (para simplificar, sólo uno sobre otro y no varios sobre otros). Existe un manipulador que puede tomar y reubicar los bloques.

Las acciones que puede llevar a cabo el manipulador son los operadores asociados a las reglas:
(Desapilar A B)
(Apilar A B)
(Levantar A)
(Bajar A)

Los predicados que se usan para definir los estados posibles son:
(Apilado A B)
(Desapilado A B)
(Sobrelamesa A)
(Despejado A)
(Agarrado A)

Y las definiciones de los estados final e inicial:
Estado Inicial: (Sobrelamesa A) (Apilado C A) (Sobrelamesa B) (Despejado B) (Despejado C)
Estado Final: (Apilado A B) (Apilado B C)

-gráficos de los estados inicial y final están en la página 165-

Codificación de los operadores

Para pasar de un estado a otro, modificando predicados, usamos operadores. Cada operador se describe mediante una lista de nuevos predicados que el operador provoca que sean ciertos y una lista de los viejos predicados que el operador provoca que sean falsos. Estas listas se denominan AÑADIR y BORRAR.
(Desapilar A B)
P: (Apilado A B) y (Despejado A)
B: (Apilado A B) y (Despejado A)
A: (Agarrado A) y (Despejado B)

(Apilar A B)
P: (Despejado B) y (Agarrado A)
B: (Despejado B) y (Despejado A)
A: (Apilado A B)

(Levantar A)
P: (Despejado A) y (Sobrelamesa A)
B: (Despejado A) y (Sobrelamesa A)
A: (Agarrado A)

(Bajar A)
P: (Agarrado A)
B: (Agarrado A)
A: (Sobrelamesa A) y (Despejado A)

Planificación mediante aplicación de Reglas

Evitan explorar completamente el árbol o grafo. Los pasos básicos son:
Elegir la mejor regla (“Mejor según función heurística”)
Aplicar regla para obtener nuevo estado
Detectar si se ha llegado a una solución

Adicionalmente:
Detectar callejones sin salida
Refinar soluciones “casi correctas”

(Sobre C A) (Sobrelamesa A) (Sobrelamesa B) (Despejado C) (Despejado B)
(Sobre C A) (Despejado C)
A3: Mover C desde A a la mesa
-(sobre C A) (Sobrelamesa A) (Despejado A)

Planificación mediante una Pila de Objetivos

Se plantea una pila de objetivos como: (Apilado C A) y (Apilado B D) y (Sobrelamesa A) y (Sobrelamesa D)

Se tratan los objetivos como sub problemas separados a resolver. Acá hay cuatro sub problemas: (Sobrelamesa A) y (Sobrelamesa B) ya se cumplen por lo que quedan dos.

(Apilado C A)
(Apilado B D)
(Apilado C A) y (Apilado B D) y (Sobrelamesa A) y (Sobrelamesa B)

Como el primer objetivo no se cumple se debería aplicar el operador que lo transforme, lo que se hace es reemplazar al predicado objetivo por el operador y luego agregar a la pila las precondiciones que deben cumplirse para ese operador.

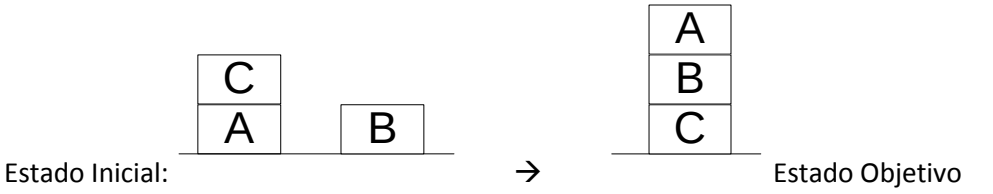
Al ir extrayendo los operadores de esa pila, se incorporan a la lista PLAN y se actualiza la BD.

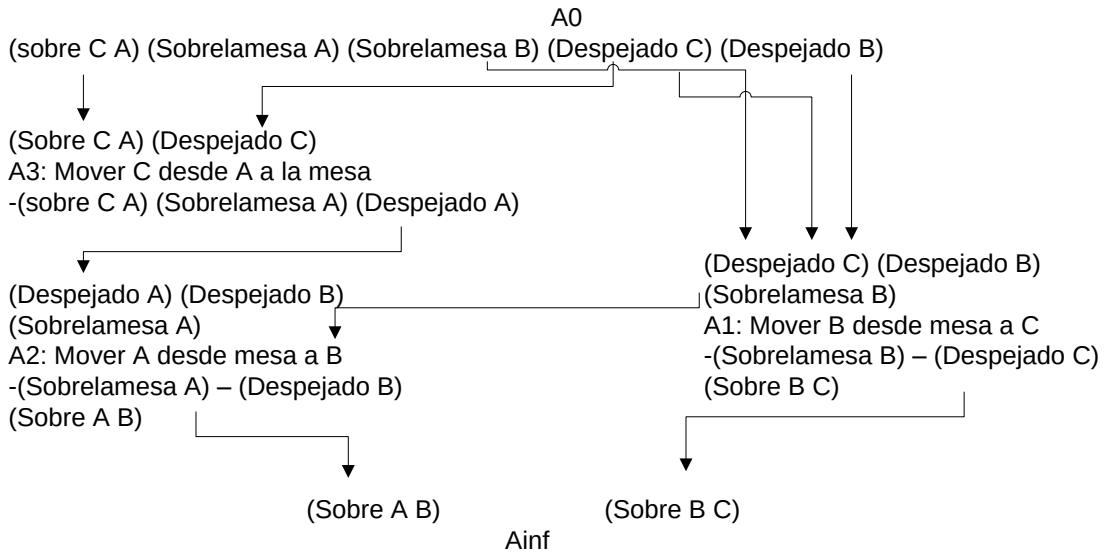
Ejemplo de Planificación mediante una Pila de Objetivos

-Ver ejemplo para entender bien en página 170-

Detección de bloqueo o amenaza de operadores a predicados

Supongamos que queremos obtener la secuencia de operadores para la siguiente transformación de estados:





Se observa en el primer nodo de la derecha se ve que (Despejado B) es “amenazado” por el segundo nodo de la rama izquierda. Manteniendo estas posibles alternativas en memoria es posible evitar potenciales bloques o repetición de pasos en el plan a generar.

Anomalía de Sussman

El método puede dar lugar a un comportamiento anómalo consistente en la inclusión en el resultado (plan) de pasos que no llevan a la solución, y que por ese motivo deben anularse posteriormente a haberlos incluido. Esto puede dar lugar a ciclos.

-Ver ejemplo en página 174-

COMPLEJIDAD

En informática, se asocia la complejidad con los recursos de cómputo requeridos para resolver un problema. Estos recursos son normalmente tiempo y espacio, y la *Teoría de la complejidad computacional* (una rama de la ciencia de la computación) se encarga del estudio de estos.

La teoría de la complejidad computacional aparece asociada a la *Teoría de la computabilidad*, aunque sus objetivos son distintos. El foco de esta última está en la existencia o no de procedimientos que permitan resolver ciertos problemas.

Si bien los avances de la tecnología vienen incrementando de forma sostenida la capacidad y calidad de los recursos de computación, los objetivos de los sistemas informáticos son cada vez más ambiciosos. Y la Inteligencia Artificial no es ajena a esta realidad, donde el permanente descubrimiento, desarrollo y aplicación de nuevas técnicas y herramientas conducen a un incremento en la complejidad. En este contexto la complejidad computacional es uno de los principales factores s ser considerados.

Procedimientos y algoritmos

Se denomina procedimiento a todo conjunto finito y ordenado de instrucciones, de tal forma que puedan ser ejecutadas por una persona o una máquina sin necesidad de conocimiento adicional a lo expresado en las propias sentencias. Lo anterior implica que un problema será considerado resuelto cuando se encuentre una forma mecánica de operar a partir de sus datos hasta alcanzar un resultado.

Así, la búsqueda de un procedimiento es el eje en la solución de todo problema computacional, la cual está asociada a dos interrogantes: ¿Existirá tal procedimiento? En caso de existir ¿Cuál será el costo computacional de su aplicación? El primero de estos interrogantes está asociado al concepto de computabilidad y el segundo al de complejidad.

Un caso particular está representado por aquellos procedimientos que permiten alcanzar siempre una solución en un número finito de pasos, los cuales son definidos algoritmos o procedimientos efectivos.

Habrá problemas en los que no es posible hallar procedimientos y otros problemas en los que sí es posible pero no existen algoritmos. Esto puede no ser intrínseco del problema y depender solo de los datos.

En resumen, un procedimiento es una secuencia finita de de instrucciones que pueden ser cumplidas en forma mecánica, como el caso de un programa de computadora. Un procedimiento que siempre termina lleva el nombre de algoritmo.

Complejidad

Intuitivamente, se considera a un problema complejo si su resolución requiere el empleo de un algoritmo complejo, es decir, que involucre cálculos complicados o su lógica sea compleja. En consecuencia, el concepto de complejidad no parece ser cuantificable o medible, sino más bien subjetivo.

Se pretende entonces establecer una escala que clasifique los problemas permitiendo establecer medidas o clases de complejidad partiendo de que un problema es más complejo que otro.
Se establecen tres aspectos para la definición de métricas:
Identificar los parámetros primitivos que serán medidos para tener datos característicos del objeto estudiado.
Definir las propias métricas, representadas por expresiones, destinadas a determinar indicadores que representen niveles de complejidad.
Establecer las condiciones en las que se harán las mediciones con el objetivo de que diferentes mediciones sean comparables.

Medidas y métricas de complejidad

Se quiere determinar indirectamente la complejidad de un problema a partir de los recursos necesarios para resolverlos. Se dice que es indirecto porque no se evalúa la complejidad del problema sino de los recursos demandados por el procesamiento empleado. Esto surge a partir de la convicción de que la solución de un problema de elevada complejidad demandara más recursos que la de un problema sencillo.

Se seleccionan el tiempo y el espacio como los indicadores más representativos.

Complejidad temporal

Se refiere a la cantidad de intervalos o unidades elementales que demanda completar la ejecución de un proceso.

Se expresa en función del tamaño del problema o de sus datos, y en este contexto se busca encontrar la razón de crecimiento entre el tiempo y la dimensión de los datos. Se hablara de complejidad lineal, logarítmica, polinómica, exponencial, entre otros, según la naturaleza de la expresión que las vincula.

El límite en el crecimiento de esta medida se denomina complejidad temporal asintótica y es lo que determina el tamaño del problema.

Se dice que una función g(n) tiene un orden de complejidad temporal O(f(n)) si existe una constante C tal que g(n) <= Cf(n) para todos los posibles valores positivos de n.

Definimos la complejidad temporal, entonces, como el número de unidades de tiempo requeridas para procesar una entrada de dimensión n.

Complejidad espacial

Se refiere a la cantidad de espacio de almacenamiento requerido para resolver un problema a través de un algoritmo. Esto se apoya en la idea de que al aumentar la complejidad de un problema también aumenta el espacio que demanda.

Mide la razón de crecimiento entre la complejidad del problema y el espacio requerido, y se representa por E(n). El límite de crecimiento de esta medida se denomina complejidad espacial asintótica y determina el tamaño del problema que puede resolver cierto algoritmo.

Clase de problemas

Todos los problemas matemáticos imaginables pueden dividirse en dos grupos, los que admiten un algoritmo para su resolución y los demostrablemente irresolubles.

Al grupo de problemas de solución algorítmica se los divide en problemas de *complejidad temporal polinómica* y los de *complejidad temporal exponencial*. Los primeros tienen su tiempo de ejecución acotado por una función de este tipo y son considerados eficientes. Los segundos carecen de interés práctico, y solo puede alcanzarse una solución cuando la dimensión de los datos “n” es pequeña (problemas denominados intratables).

En la segunda clase se encuentran los problemas que pueden ser resueltos pero que no admiten un método general, es decir, no disponen de un procedimiento. En esta segunda clase se encuentran los problemas de solución iterativa y los de solución probabilística.

GRAFICO

No siempre es factible establecer a cuál de las dos subclases pertenece un cierto problema. Esto es porque hay problemas cuyas soluciones requieren tiempos exponenciales pero no se ha demostrado que no tienen solución de tiempo polinomio. Estos son denominados *no determinísticamente polinomios* o problemas tipo NP. Un subconjunto de estos últimos se denominan NP-Completos.

Complejidad e Inteligencia artificial

Definidas las complejidad espacial y temporal, se aplica lo visto hasta aquí a algunos problemas clásicos de IA como el entrenamiento de redes neuronales artificiales y algoritmos de búsqueda.

Redes neuronales: La complejidad temporal del proceso de entrenamiento de redes multicapa de perceptrones es de tipo polinómica, entre grado 3 y 5. Se trata de un problema de solución algorítmica considerado eficiente.

Algoritmos de búsqueda: La complejidad temporal de todos los métodos tiene un límite exponencial. La solución de este tipo de problemas puede ser caracterizada como NP-Completo, pero como mínimo es un problema NP.