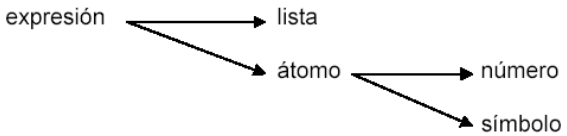
1.1. LISP

Programa interprete basado en el lenguaje funcional, no genera daños colaterales, es un hibrido ya que me permite relizar bucles y asignar variables. El nombre de LISP surge de abreviar "List Procesing" o Procesamiento de Listas. Fue propuesto por John Mc Carthy en 1959 en el MIT. En LISP toda la información (inclusive las sentencias) se expresan como listas o como los componentes de las mismas.

Un código de LISP está compuesto de formas o expresiones; el intérprete LISP lee una expresión, la evalúa e imprime el resultado. Este procedimiento se denomina ciclo de lectura, evaluación, impresión (read – eval – print loop).



Una **expresión o forma** es una lista o un átomo (símbolo, entero o cadena).

Un átomo es un símbolo o un número. Si la forma es una lista LISP trata el primer elemento como el nombre de una función, y evalúa los restantes elementos recursivamente, y luego llama a la función con los valores de los elementos restantes.

Un **número** se puede escribir en cualquiera de las notaciones habituales.
Un **símbolo** se representa por un nombre que está formado por caracteres alfanuméricos que no puedan interpretarse como números.
Una **lista** es una sucesión de cero o más expresiones entre paréntesis. El blanco (no la coma) es el separador de expresiones dentro de una lista. Cada una de estas expresiones se denomina elemento de la lista.

Cuando LISP evalúa una expresión, devuelve un valor (que será otra expresión) o bien señala un error.
- Un número se evalúa a sí mismo.
- Un símbolo se evalúa al valor que tiene asignado.
- Una lista se evalúa independientemente una por una del siguiente modo, siempre en función del primer elemento y en función de este considera como emplear los elementos siguientes:

1) **El primer elemento** de la lista debe ser un símbolo "s" cuyo significado funcional "F" esté definido.
2) Se evalúan los restantes elementos de la lista, obteniendo los valores v1,…,vn. Si el número o tipo de los valores obtenidos no es coherente con los argumentos requeridos por F, se produce un error.
3) La lista se evalúa a F(v1,…,vn).
La forma en la que se evalúan las listas corresponde a una notación pre fija, donde primero se escribe el operador y luego los parámetros.

1.2. FUNCIONES

Existen dos símbolos (T y NIL) que se evalúan a si mismos. Estos se utilizan como TRUE y FALSE o como TRUE y NULL en otros lenguajes. Un símbolo "s" puede estar ligado, es decir, contener una referencia o indicación a otra expresión "V". Llamamos ligadura al par (s,V). Un símbolo "s" (salvo T o NIL) se evalúa al valor "V" al que está ligado. Recordar que forma es una expresión que se puede evaluar.

QUOTE
Es una forma que tiene un solo argumento. (QUOTE expresión)
El valor de (QUOTE expresión) es precisamente "expresión", sin evaluar. La abreviatura de QUOTE es " ' ".
(QUOTE expresión) = ' expresión
Ejemplo: (+ 3 4) 7, '(+ 3 4) (+ 3 4), (1 2 3) error, '(1 2 3) (1 2 3)

SETQ
Asigna el segundo argumento (que es una expresión) al primer elemento (que no es evaluado). (SETQ símbolo expresión)

CAR: Un único elemento, que debe ser una lista. Si el argumento no es NIL, el valor devuelto es el primer elemento de la lista argumento. Si el argumento es NIL, el valor devuelto es NIL.

CDR
Un único argumento, que debe ser una lista. Si el argumento no es NIL, el valor devuelto es la lista obtenida quitando el primer elemento de la lista argumento. Si el argumento es NIL, el valor devuelto es NIL. Se pueden hacer combinaciones de CAR y CDR

CONS
Exactamente dos argumentos. El segundo debe ser una lista. El valor devuelto es una lista cuyo primer elemento es el primer argumento, y cuyos restantes elementos son los elementos del segundo argumento.

LIST
Un número indeterminado de argumentos. Si no hay ningún argumento, el valor devuelto es NIL. Si hay algún argumento, el valor devuelto es una lista cuyos elementos son los argumentos.

APPEND
Un número indeterminado de argumentos, que deben ser listas. Si no hay ningún argumento, el valor devuelto es NIL. Si hay algún argumento, el valor devuelto es una lista cuyos elementos son los elementos de los argumentos.

LENGTH
Un único argumento, que debe ser una lista. El valor devuelto es la longitud del argumento, es decir, el número de elementos de la lista.

EVAL: Un único argumento. El valor devuelto es el valor resultante de evaluar el argumento. De esta forma, (EVAL expresión) realiza dos evaluaciones: la primera debido al ciclo normal de evaluación y la segunda debido al uso explícito de EVAL.

DEFUN
Es una macro. Ello quiere decir que no evalúa sus argumentos en la forma explicada. DEFUN toma sus argumentos literalmente. El primer argumento es in símbolo. El segundo argumento es una lista de cero o más símbolos que se denominan parámetros (lista-lambda). El cuerpo está formado por un número cualquiera de expresiones.
(DEFUN símbolo lista-lambda cuerpo)
(DEFUN cuadrado (N) (* N N) CUADRADO
(cuadrado 7) 49

PREDICADOS
Son las funciones que devuelven T o NIL.
(EQ e1 e2) es T cuando e1 y e2 son el mismo símbolo
(EQL e1 e2) es T cuando e1 y e2 son EQ o números iguales de igual tipo
(EQUAL e1 e2) es T cuando e1 y e2 son EQL o listas iguales
(= e1 e2 … en) es T cuando e1, e2, …, en son números todos iguales de cualquier tipo
(NULL e) es T cuando e es NIL
(ATOM e) es T cuando e es un átomo
(SYMBOLP e) es T cuando e es un símbolo
(NUMBERP e) es T cuando e es un número
(LISTP e) es T cuando e es una lista
(ENDP e) es T cuando e es NIL (lista vacía). Si e no es una lista da error
(ZEROP e) es T cuando e es cero
(PLUSP e) es T cuando e > 0
(MINUSP e) es T cuando e < 0

IF
Tiene como argumentos: Una condición. Una o dos expresiones. Si la condición es T, se evalúa la primera condición, en caso contrario se evalúa la segunda.

COND
Tiene un número indeterminado de argumentos. Cada argumento es una cláusula. Una cláusula es una lista formada por una o más expresiones. La primera expresión de cada cláusula es la condición y a las restantes se las puede denominar acciones.

Recursividad
LISP también soporta el concepto de recursividad.

DO / DO*
Son las formas iterativas más generales de Lisp.
(DO (ligadura *)
(test resultado *)
expresión-cuerpo *)
donde "ligadura" puede ser:
símbolo | (símbolo) | (símbolo expr-inicial) | (símbolo expr-inicial expr-paso)

El significado es el siguiente:
- Se crea un entorno con ligaduras para los símbolos indicados.
- Se evalúa cada exp.-inicial (valor por defecto: NIL) y sus valores son asignados a los correspondientes símbolos. La evaluación es en paralelo para DO y secuencial para DO*
- Se evalúa test.
- Si test es verdadero, se evalúa cada expresión de la sucesión "resultado*", la evaluación de DO acaba y el valor que se devuelve es el de la última (por defecto NIL).
- Si test es falso, la evolución de DO continúa: primero se evalúa secuencialmente (en caso de existir) las expresiones del cuerpo. A continuación se evalúan las exp.-paso (en paralelo para DO y secuencialmente para DO*) y los valores son asignados a los correspondientes símbolos. A continuación se evalúa de nuevo el "test" y se repite todo el proceso.

DOLIST
Evalúa "expr-cuerpo" sucesivamente con "variable" ligado a cada uno de los valores "lista" y finalmente devuelve el resultado de evaluar "expr-final" con "variable" ligada a NIL.

LAMBDA
(LAMBDA lista-lambda [expresión] *) Las expresiones lambda sirven para referirse a funciones sin necesidad de darles nombre. Una expresión lambda da directamente la definición de la función. La "lista-lambda" es la lista de parámetros que serán utilizados en "expresión". Una expresión lambda puede aparecer en los mismos lugares que un símbolo con significado funcional. Por ejemplo, puede aparecer como primer elemento de una lista que se ha de evaluar; se dice entonces que tenemos una forma lambda. Los restantes elementos de la lista son los argumentos que se pasan a la definición lambda. Estos argumentos sustituyen a los parámetros en "expresión" y se devuelve el valor de la última de ellas.

FUNCTION - #'
Hasta ahora hemos considerado en Lisp dos tipos de datos: átomos y listas. Lisp considera también como datos a los objetos-procedimientos. Un objeto procedimiento no debe confundirse con el símbolo que lo tiene ligado.
Existe un objeto-procedimiento (llamémosle Función1) que es el significado funcional del símbolo F1. El objeto-procedimiento Función1 se recuperaría implícitamente al evaluar una lista de la forma: (F1 argumento)
Hasta ahora esta manera de tratar los objetos-procedimiento, propia de los lenguajes imperativos, es la única que hemos considerado. Sin embargo, en Lisp es posible manipular explícitamente un objeto-procedimiento mediante la forma especial FUNCTION

(FUNCTION argumento)
- Un solo argumento, que debe ser un símbolo o una expresión lambda.
- FUNCTION toma literalmente su argumento (no lo evalúa)
- El valor devuelto es:
o Si el argumento es un símbolo, el objeto-procedimiento que es el significado funcional del símbolo.
o Si el argumento es una expresión lambda, el objeto-procedimiento definido por la expresión.
o Otro tipo de argumento es un error.

OPTIONAL
Los parámetros que siguen a &OPTIONAL en una lista lambda son opcionales y de la forma:
(símbolo | símbolo expresión)
Las expresiones para los valores por defecto se evaluarán secuencialmente, por lo tanto es posible emplear en ellas los argumentos anteriores de la lista lambda.

KEY
Los parámetros de una lista lambda que siguen a &KEY son parámetros-clave. Los parámetros-clave son siempre opcionales, pero además de esto, cuando en una lista lambda aparece un parámetro-clave las reglas para asignar los argumentos a los parámetros no son las vistas anteriormente, es decir, los argumentos no se asignan en el mismo orden en que aparecen en la llamada.
Las reglas son las siguientes:

a) Los parámetros requeridos se emparejan con los primeros argumentos de la llamada. Si hay mas o menos, se produce un error.
b) A cada parámetro clave se le hace corresponder una palabra-clave, añadiéndole al principio el signo ":". Las palabras-clave son símbolos que se evalúan a sí mismos.
c) Si en la llamada figuran argumentos para los parámetros-clave, cada uno debe ir precedido por la correspondiente palabra-clave. El emparejamiento entre los parámetros y argumentos se produce independientemente del orden de aparición de los argumentos.

2. Capitulo 6: Sistemas Expertos

2.1. ¿QUE SON LOS SISTEMAS EXPERTOS?

Es un programa destinado a generar inferencias en aun área específica del conocimiento en forma similar a la que se espera de un experto humano. Deben resolver problemas por aplicación de conocimiento en 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. Son aplicaciones a mundos reducidos, ya que no pueden operar en situaciones llamadas de sentido común por ser muy extenso el dominio de conocimiento que debe tener el sistema.

Se utiliza SE cuando:
- No existe algún algoritmo para resolver un problema
- El problema es resuelto satisfactoriamente por expertos humanos
- Existe algún experto humano que pueda colaborar en el desarrollo de un SE
- El conocimiento del dominio debe ser relativamente estático

(dibujo)

Símbolos: Las diferentes definiciones que se van a utilizar en el SE, afirmaciones, elementos, etc.
Reglas: Se aplican sobre los símbolos, se deben obtener del conocimiento de los expertos. Son heurísticas dado que no es posible demostrar su validez general.
Hechos: son todos los predicados que se suponen verdaderos.

2.1.1. FUNDAMENTOS

Los sistemas expertos básicos se basan en la lógica de predicados. La diferencia entre esta y la proposicional es que la lógica de predicados emplea variables y permite expresar una premisa en una sola proposición mientras que en la otra deberíamos usar una proposición por cada una de las variables que estemos analizando. La representación mediante predicados es una forma de representar la estructura del conocimiento.

Para que la lógica de primer orden nos ayude a resolver problemas y garantice inferencias validas, el significado de los símbolos a emplear debe ser preciso no deben existir ambigüedades. En el caso de los SE se pretende aplicar la lógica a mundos que solo conocen en profundidad los expertos humanos, en que ellos expresan su conocimiento en lenguaje natural. Por lo que para que un SE basado en el formalismo lógico requiere definir los símbolos con los cuales se operará y su significado preciso, además de traducir el conocimiento del experto del lenguaje a natural a la expresiones lógicas correspondientes (reglas).

Los SE se basan en la posibilidad de aplicar las nociones de la lógica formal para incrementar una base de conocimientos y solo pueden aplicarse a mundos de muy baja complejidad. El modo mas inmediato consiste en aplicar reglas del tipo si-entonces e instanciando los antecedentes de la misma obtener una instancia valida para el consecuente.

En lugar de instanciar el consecuente para ver si es soportado por sus antecedentes se podría trabajar a la inversa:

Encadenamiento hacia adelante el SE toma una 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.

Encadenamiento hacia atrás se comienza con una hipótesis y se trata de verificarla haciendo uso de las afirmaciones disponibles.

2.2. LÓGICA DE PREDICADOS

2.2.1. LÓGICA

La **lógica** permite la posibilidad de utilizar un conjunto de reglas para gestionar inferencias creíbles. Una **inferencia** es la extracción de conclusiones a partir de premisas mas o menos explicitas, puede ser resultante del sentido común o de la aplicación de reglas muy detallistas o cálculos estadísticos. La **epistemología** estudia el origen y características del conocimiento, en especial la forma en como 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.

El **silogismo** es una formula lógica con premisas seguidas de una conclusión.

La **lógica de primer orden** es uno de los formalismos más utilizados para representar el conocimiento. Cuenta con un lenguaje formal mediante el cual es posible representar formulas llamadas **axiomas** que permiten describir fragmento del conocimiento y consta de un conjunto de **reglas de inferencia** que aplicadas a los axiomas permiten derivar nuevo conocimiento.

El alfabeto

Posee dos símbolos:

- Lógicos: constantes, operadores proposicionales y cuantificación, lógicos y auxiliares.
- No lógicos: variables individuales, funciones n-arias y relaciones n-arias.

A partir de los símbolos se construyen las expresiones:

- Términos: una constante, palabra, letra, variable y una expresión de la forma.
- Formulas: son expresiones de la forma R(t1….tn) donde R es un símbolo de relación y t1…tn son los términos.

Los términos permiten nombrar objetos del universo, mientras que las formulas permiten afirmar o negar propiedades de estos o bien establecen relaciones entre los objetos del universo.

2.2.2. FORMULAS BIEN FORMADAS

La lógica proposicional permite el razonamiento mediante un mecanismo que primero evalúa sentencias simples y luego sentencias complejas mediante el uso de conectivos proposicionales. Una preposición es una sentencia simple con un valor asociado. La lógica proposicional permite la asignación de un valor v o f para la sentencia compleja, no tiene la facilidad para analizar las palabras individuales que componen la sentencia. Una FBF es una proposicion simple o compuesta de significado completo cuyo valor de veracidad puede ser determinado, basado en los valores de veracidad de las proposiciones simples y en la naturaleza de los conectores lógicos involucrados.

El **Silogismo categórico** dice que si es verdadera una instancia cualquiera de las siguientes premisas: Si M es A y B es M entonces B es A. la **resolución** se basa en que una afirmación no puede ser simultáneamente verdadera y falsa.

2.2.3. ELEMENTOS BASICOS

- Átomos: proposiciones simples.
- Conectivos lógicos: expresiones que sirven para formar preposiciones compuestas a partir de preposiciones simples.
- Sentencias: átomos o clausulas formadas por la aplicación de conectivos a sentencias.
- Cuantificadores: universal $\forall$ y existencial $\exists$
- Símbolos de puntuación y delimitación.

Silogismos

Hipotético: A es B y B es C → A es C

Disyuntivo: –A $\vee$ B es A → B

Función de Skolem

Cuando un cuantificador existencial esta en ámbito de uno universal, la variable cuantificada debe ser reemplazada con una función de skolem: $\forall$ x $\exists$ y (x<y) entonces $\forall$ y (y-1<y).

Leyes de Morgan

$-(A \wedge B) = -A \vee -B$ y $-(A \vee B) = -A \wedge -B$

FBF se pueden expresar:

- Forma normal conjuntiva $(A \vee B) \wedge (A \vee B)$
- Forma normal disyuntiva $(A \wedge B) \vee (A \wedge B)$
- Clausula de Horn: Es una conjunción de disyunciones con no más de un literal positivo (no negado). Una expresión es valida satisfacible si alguna combinación de valores v o f de sus átomos hacen verdadera la expresión.

2.2.4. REDUCCIONA LA FORMA CLAUSAL

Reducir los predicados a un conjunto de clausulas (afirmaciones) para que puedan ser procesadas por un mecanismo logico

1. Eliminar implicaciones
2. Reducir la aplicación de negaciones
3. Eliminar cuantificadores
4. Aplicar propiedad disyuntiva y asociativa
5. Que cada clausula tenga nombre de variable único.

2.3. RESOLUCIÓN

La demostración se lleva a cambo mediante la reducción al absurdo. El problema puede surgir con un gran número de combinaciones posibles. Es en esencia un procedimiento 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.

2.4. UNIFICACIÓN Y LIGADURA

Permite razonar dentro de un esquema formal sin relación con la codificación de los programas. En el caso de un código, este se encarga de reducir las formas clausales a listas asociadas mas simples de manipular denominadas ligaduras, esta fluye a través de los filtros del programa y al final del proceso pueden ser analizadas para extraer nuevas afirmaciones o demostrar hipótesis.

2.5. CORRESPONDENCIA DE PATRONES

Relacionado con la codificación de SE es la correspondencia de patrones simbólicos. La correspondencia con patrones se basa en identificar si dos vectores son similares es decir se corresponden. La forma simple de entenderlo es calcular la distancia entre los mismos y si esta es lo suficientemente pequeña de acuerdo a valores preestablecidos para un problema dado, entonces puede decirse que ambos vectores se corresponden.

2.5.1. CORRESPONDENCIA SIMBÓLICA

Busca similitudes entre un conjunto de nombres de variables y no entre los valores numéricos de estas. Se consideran dos procedimientos:

Correspondencia: encadenamiento hacia adelante, comparando una expresión común (dato) con un patrón. A diferencia de los datos los patrones pueden contener variables patrón, para facilitar el reconocimiento de estas variables se asocian a un símbolo.

Unificación: encadenamiento hacia atrás. Hace corresponder dos patrones en lugar de un patrón y un dato.

2.5.2. CORRESPONDENCIA SUB SIMBOLICA

Correspondencia numérica.

2.6. ENCADENAMIENTO.

2.6.1. PROGRESIVO

- **Correspondencia**: Busca 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.

2.6.2. REGRESIVO

- **Correspondencia:** Busca afirmaciones en la base de datos que correspondan a una hipótesis.
- **Unificación:** buscar una regla cuyo consecuente concuerde con la hipótesis. La hipótesis y el consecuente pueden tener variables de patrón.
- **Resolución:** para cada regla cuyo consecuente concuerde con la hipótesis, se intenta verificar de manera recursiva cada antecedente de la regla, considerando a cada una como una hipótesis.

3. Capitulo 7: Lógica difusa (fuzzy)

En los sistemas binarios solo pueden ser evaluados como verdaderos o falsos, pero los razonamientos concernientes a problemas reales muestran que es posible considerar cierto grado de certeza o falsedad para una afirmación y no valores absolutos.

Para incorporar este concepto a los sistemas de inferencia y a los SE se recurre a modelos probabilísticos, es decir se asignan valores de probabilidad a las sentencias y se propagan a través de las reglas. Existen diferentes formas de calcular como influye la probabilidad de los antecedentes sobre los consecuentes.

Los campos de aplicación comunes son: control automático, clasificación, soporte de decisión, SE y visión computarizada.

3.1. LA SOLUCION FUZZY

Ante un problema que presenta diferentes grados de veracidad, seria deseable plantear la solución del problema en términos que permitan capturar la esencia del mismo y eliminar factores arbitrarios. Si damos un significado matemático a las variables lingüísticas obtendremos un sistema de inferencia basado en lógica difusa. Ej. El servicio es pobre la propina es poca, que significa una propina “poca”.

Si la lógica difusa nos permite escribir un código para la solución que resulte mas fácil de interpretar para las personas, entonces estamos diciendo que su contribución radica en el hecho de escribir un código ventajoso.

El sistema difuso esta de algún modo basado en el sentido común, por lo se pueden agregar sentencias al final de la lista que modelaran mas apropiadamente la forma de la función.

El mantenimiento del código será más simple.

3.2. VISIÓN GENERAL

La esencia del sistema radica en mapear un espacio de entrada en uno de salida mediante la inferencia de reglas. Todas las reglas se evalúan en paralelo. Las reglas de la lógica difusa se refieren a variables y adjetivos que califican a esas variables. Antes de discutir la construcción del sistema se definen los términos que se van a emplear y los adjetivos que los describen.

3.3. REGLAS SI-ENTONCES

Para lograr algo útil debemos construir sentencias del tipo si entonces, sentencias completas condicionales. La primera parte es un antecedente y la otra el consecuente. El antecedente es una interpretación que retorna un número entre 0 y 1, mientras que el consecuente es una sentencia que asigna el conjunto fuzzy completo a la variable de salida. Podríamos decir que si el antecedente es verdadero en alguna medida de pertenencia, luego el consecuente también es verdadero en el mismo grado. El antecedente puede estar compuesto por múltiples partes que son calculadas simultáneamente y resueltas como un número simple. El consecuente también puede tener varias partes.

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.

3.4. REGLAS

A- Difusión de entradas: se deben resolver todas las sentencias en el antecedente en función de su grado de membrecía entre 0 y 1

B- Aplicación de los operadores difusos: si existen varias partes en el antecedente se aplican los operadores y se resuelve el antecedente como un número entre 0 y 1.

C- Aplicación de la implicación: se emplea el grado de soporte de la regla para conformar el conjunto fuzzy de salida. Si el antecedente es parcialmente cierto el conjunto fuzzy de salida es truncado según el método de implicación.

D- Los conjuntos fuzzy de salida se amalgaman en un único conjunto. Luego este conjunto resultante es defuzificado, o resuelto de modo de obtener un número como valor de salida.

3.5. PASOS

3.5.1. FUZIFICACION

Tomar las entradas y determinar el grado en el cual ellos pertenecen a cada un de los conjuntos difusos a través de funciones de pertenecía. La entrada es siempre un valor numérico limitado y la salida es un grado difuso de pertenecía. Estas funciones de pertenencia pueden ser vistas como la curva de distribución de probabilidad del adjetivo para el cual han sido definidas.

3.5.2. OPERADOR DIFUSO

Conocemos el grado en el cual cada parte del antecedente ha sido satisfecho para cada regla. Si el antecedente de una regla ha sido dividió en mas de una parte, se debe aplicar un operador difuso para obtener un número que representa el resultado del antecedente para esa regla.

La entrada del operador difuso son dos mas valores de pertenecía desde las variables de entrada fuzzificadas. Existen distintos métodos conjunciones, disyunciones y negación.

3.5.3. MÉTODO DE IMPLICACION

La conformación del consecuente basado en el antecedente. La entrdad para la implicación es un numero dado por el antecedente, y la salida es un conjunto fuzzy que se obtiene para cada regla por separado.

3.5.4. AGREGACIÓN

Unificar las salidas para cada regla uniendo los procesos paralelos. La entrada es la lista de salidas truncadas. La salida de este proceso era un conjunto difuso para cada variable de salida. La agregación es conmutativa, el orden en que se ejecutan las reglas no es relevante.

3.5.5. DEFUZZIFICACION

Ingresa un conjunto difuso agregado y sale un número. Dado un conjunto difuso que engloba un rango de valores de salida se requiere obtener un único número. Un método popular es el cálculo del centroide.

3.6. HEURISTICA

El tipo de funcion de membresia de los adjetivos empleados en las reglas debe ser propuesta por un diseñador del sistema, los valores particulares también, el método de defuzzificacion ha de ser elegido por el diseñador, etc. Todo esto no solo muestra una aplicación particular de la heurística, sino que evidencia su importancia lo que hace que muchas veces se cuestionen los fundamentos para el diseño y contstruccion.

4. Capitulo 8: Búsqueda

4.1. CONCEPTO

Procedimiento cuya finalidad es encontrar datos dentro de un conjunto de estos, pero donde el objetivo del proceso es mas bien encontrar cierto camino recorrido hasta encontrar en dato antes de corroborar la existencia del dato. Esto se debe a que estos procesos se usan en planificación, optimización, resolución de problemas, juegos, etc. Para realizar las búsquedas se utilizan arboles y grafos para conocer la secuencia de estados que deben alcanzarse sucesivamente para lograr un estado objetivo.

4.2. METODOS NO INFORMADOS

4.2.1. PRIMERO EN ANCHURA BPA

Se generan para cada nodo todas las posibles situaciones resultantes de la aplicación de todas las reglas adecuadas. Se continúa:

1. Se crea una pila (abierta) y se le asigna el estado inicial.
2. Mientras la pila no este vacía.
    a. Extraer el primer nodo de la pila llamarlo m.
    b. Expandir m. Se generan las entradas sucesoras del estado actual, para cada operador aplicable hacer:
        i. Aplicar el operador al nodo obteniendo un nuevo estado.
        ii. Si el nuevo estado es el objetivo termina el proceso.
        iii. Incluir el nuevo estado el final de la pila volver a 2ª

Ventajas: No entra en callejones sin salida, si existe una solución garantiza alcanzarla, si existen varias soluciones garantiza alcanzar la optima.

Desventajas: utiliza mucha memoria, es lento.

4.2.2. PRIMERO EN PROFUNDIDAD BPP

Continúa por una sola rama del arbol hasta encontrar la solucion o hasta que se tome la decision de terminar la busqueda por esa dirección, se puede tomar por que: se llego a un callejón sin salida, se produce un estado ya alcanzado o la ruta se alarga mas de lo especificado.

1. Se crea una pila (abierta) y se le asigna el estado inicial.
2. Mientras la pila no este vacía.
    a. Extraer el primer nodo de la pila llamarlo m.
    b. Si la profundidad de m es igual al límite de profundidad regresar a A, en caso contrario continuar.

c. Expandir m. Se generan las entradas sucesoras del estado actual, para cada operador aplicable hacer:
    i. Aplicar el operador al nodo obteniendo un nuevo estado.
    ii. Si el nuevo estado es el objetivo termina el proceso.
    iii. Incluir el nuevo estado principio de la pila volver a 2ª

Ventajas: Si existe una solución garantiza alcanzarla, no recorre todo el árbol para alcanzar la solución, pero es por azar.

4.2.3. RAMIFICACIÓN Y ACOTACIÓN

Comienza generando rutas completas, manteniéndose la ruta mas corta encontrada hasta ese momento. Deja de explorar tan pronto la distancia total sea mayor a la marcada como la ruta mas corta.

4.3. HEURÍSTICA

Se refiere a un conocimiento que intuitivamente poseemos, que podría parecer experimental debido a la inspiración de quien lo propone. La diferencia entre lo heurístico y lo científico es que este último es un conocimiento debido a la observación, una regla empírica se cumple aunque no sepamos sus causas. En la regla heurística se espera que se cumpla, en casos particulares, y no en todos los casos en que podría aplicarse; no puede ser demostrada teóricamente, ni comprobada experimentalmente para todos los casos no se generaliza; puede resolver perfectamente un problema en particular, pero puede arrojar resultados erróneos con un problema similar.

4.4. METODOS INFORMADOS O DE BUSQUEDA HEURISTICA

La heurística es una técnica que aumenta la eficiencia de un proceso proporcionando una guía para este y posiblemente sacrificando demandas de complejidad. La búsqueda heurística resuelve problemas complicados con eficacia, garantizando una estructura de control que encuentra una buena solución.

Porque usar heurísticas:

- No enredarnos en una explosión combinatoria.
- Buscar una solución adecuada.
- Los peores casos, raramente ocurren.
- Profundizar nuestra comprensión del dominio del problema.

4.4.1. PRIMERO EL MEJOR

Combina las ventajas de BPA y BPP sigue solo un camino a la vez y cambia cuando alguna ruta parezca mas prometedora. Estrategia:

1. Se selecciona el nodo mas prometedor según la función heurística apropiada
2. Se expande el nodo elegido de acuerdo a las reglas de expansión adecuadas
3. Si alguno de los nodos es el optimo se termina, sino:
    a. Se añaden nuevos nodos a la lista
    b. Se vuelve al paso 1

*4.4.2. A**

Se modifica el anterior empleando una función heurística para estimar el costo desde un nodo actual al nodo objetivo. Realiza una estimación de cada uno de los nodos que se van agregando, esto permite examinar los caminos más prometedores.

Utiliza la siguiente función: f´=g+h´. Donde f´=estimación del costo desde el nodo inicial al nodo objetivo, g =nivel del árbol en que se encuentra el nodo, h´=estimación desde el nodo actual al nodo objetivo.

Se emplean dos listas. Abierta: los nodos que se les ha aplicado la función heurística. Cerrada: nodos que ya han sido expandidos.

Se toma el nodo más prometedor y que no haya expandido. Se generan sus sucesores y se le aplica la función heurística y se los añaden a la lista de nodos abiertos

4.4.3. GENERACIÓN Y PRUEBA

La generación de la posible solucion puede realizarse de formas diferentes. Consiste en generar la posible solucion, verficar sis es una solución, si no se hallo la solucion volver a empezar de lo contrario devolverla y terminar.

4.4.4. ESCALADA

Simple: se usa para ayudar al generador a decidirse por cual dirección moverse. La función de prueba se amplia con una función heurística que ayuda a evaluarlos estados y proporciona una estimación de lo cerca que nos encontramos del objetivo. Necesita más tiempo para alcanzar la solución.

Máxima Pendiente: es una variación del anterior, que considera todos los movimientos posibles a partir del estado actual y elige el mejor de ellos como nuevo estado. Tarda más en seleccionar un movimiento. Puede caer en un máximo local (mejor q todos pero no el mejor), meseta (estados vecinos produce el mismo valor, no se sabe a donde moverse) o cresta (no tengo operadores para ubicar la cima)

4.4.5. ENFRIAMIENTO SIMULADO

El objetivo es disminuir la probabilidad de caer en una meseta, máximo local o cresta. Por lo que se realiza una exploración con saltos amplios al principio que se van reduciendo paulatinamente. Se plantea minimizar la función objetivo, para alcanzar un estado final de mínima energía.

4.4.6. MEDIANTE AGENDA

Consideramos a cada nodo como una tarea que un sistema debe realizar. Al expandir una tarea van a existir diversos caminos. Cada camino que recomienda una tarea proporciona una razón por la cual debe ser realizada. Cuantas mas razones haya, aumenta la probabilidad de que la tarea lleve a algo adecuado. Para almacenar las tareas se propone el concepto de agenda, que es en definitiva el conjunto de tareas que el sistema debe realizar.

4.4.7. REDUCCIÓN DE PROBLEMAS

Consiste en dividir el problema en sub problemas hasta llegar al nivel de módulos. En este caso, un nodo no es mejor por si mismo, sino por pertenecer a una ruta mas prometedora.

4.4.8. VERIFICACIÓN DE RESTRICCIONES

Define los estados como un conjunto de restricciones que deben satisfacerse para resolver completamente el problema, de este modo lo que se busca es un estado que satisfaga un conjunto de restricciones impuestas por el problema.

4.4.9. ANÁLISIS DE MEDIOS Y FINES

Permite razonar tanto hacia adelante como para atrás, permite resolver las partes más importantes del problema y después volver atrás y resolver los pequeños problemas. Se centra en la detección de diferencias entre el estado del nodo actual y el nodo objetivo.

4.4.1. ALGORITMO DE RUTEO

No solo busca que existan las rutas a conectar, ni que sean las rutas mas cortas, sino que la ruta encontrada para cada par de puntos sea la ruta que menos interfiera con el menor número de rutas.

5. Capitulo 9: Planificación

5.1. INTRODUCCION

Se refiere al proceso de computar varios pasos de un procedimiento de resolución de un problema antes de ejecutar alguno de ellos.

Cuando nos enfrentamos con el problema de planificar y ejecutar una secuencia de pasos en un mundo que no es completamente predecible nos enfrentamos al problema de planificación en IA.

5.2. CLASIFICACION

Planificación en IA:

- Numérica
- Lógica: Por pila de objetivos, No lineal, Jerárquica
- Reactiva

Planificación en robótica

- Reactiva
- De trayectoria de manipuladores

Planificación industrial: métodos basados en simulación discreta

5.3. EL PROBLEMA

El problema de clasificación se caracteriza porque se puede aplicar a diversos dominios además de que no se pueden explotar todas las opciones.

Entradas: Descripción dele estado del mundo, descripción del objetivo, conjunto de acciones.

Salida: una secuencia de acciones que pueden ser aplicadas al estado, hasta alcanzar la descripción del estado final.

La lógica nos ofrece una manera de reducir los pasos de búsqueda, representando los estados por predicados aplicando reglas lógicas para pasar de uno a otro, dicha aplicación debería modificar el estado anterior definiendo nuevos estados. La regla en si nos dice que esta permitido pasar de un estado a otro pero no efectúa el cambio necesario para generar este paso. Para que se modifique el estado actual es necesario aplicar procedimientos (operadores).

5.4. MUNDO DE LOS BLOQUES

Para poder hacer un estudio sistemático de los métodos y poder compararlos usamos el mundo de los bloques. Se basa en una superficie palana en la que se colocan bloques (de forma cubica únicamente). Los bloques se pueden ubicar unos sobre otros y se pueden llevar a cabo acciones que pueden manipular los bloques (operadores asociados a las reglas).

5.5. CODIFICACION DE LOS OPERADORES

Para poder pasar de un estado a otro modificando predicados usamos operadores, cada operador puede describirse 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.

5.6. PLANIFICACION MEDIANTE UNA PILA DE OBJETIVOS

Elementos: Bloques, superficie y brazo operador.

Estados de los elementos: estado del bloque y del brazo.

Operadores: acciones a realizar. Se componen de tres listas: precondición (lo que se debe cumplir para aplicar el operador), Borrado (lo que se debe eliminar de la pila de objetivos), Adición (lo que se agrega al plan de tareas)

Pilas: en todo momento se cuanta con una pila de objetivos y operadores, una de estado actual y la del plan a seguir.

Procedimiento: se tratan los objetivos como sub problemas a resolver. Se pregunta si el primer objetivo se cumple. Si no se cumple se debe aplicar el operador que lo transforme, por lo que se debe reemplazar al predicado objetivo por el operador y luego agregar a la pila las precondiciones que deben cumplirse para ese operador. Para resolver el objetivo que ha quedado arriba de la pila empleamos el operador reemplazando y agregando las precondiciones. Al ir extrayendo los operadores de esa pila, se incorporan a la lista plan y se actualiza el estado actual.