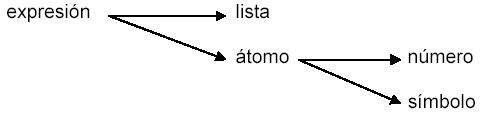
# **LIPS**

Programa interprete basado en el lenguaje funcional, no genera daños colaterales, es un hibrido ya que me permite realizar 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.

# **SISTEMAS EXPERTOS**

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

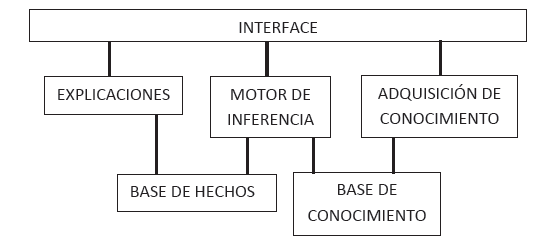
Deben resolver problemas por aplicación de conocimiento en un dominio específico. Este conocimiento es adquirido a través de la intervención de expertos humanos y almacenado en lo que se denomina Base de Datos de Conocimiento. Son aplicables a mundos reducidos, ya que no pueden operar en situaciones 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

El núcleo del sistema experto es el motor de inferencia.

Arquitectura



Los componentes básicos de un sistema experto son:

* Símbolos: Las diferentes definiciones que se van a utilizar en el SE, afirmaciones, elementos, etc.
* Reglas: son reglas a aplicar sobre los símbolos. Estas se deben obtener del conocimiento de los expertos humanos. Las reglas son consideradas heurísticas, dado que no se demuestra su validez general.
* Hechos: son predicados que se suponen verdaderos. Juntos constituyen la base de conocimiento.

1- El núcleo de un SE es el motor de inferencia  
2- Reglas a aplicar sobre los símbolos definidos.   
 - Se obtienen del conocimiento del experto humano.  
 - Se combinan con un sistema de inferencias.  
 - Se consideran heurísticas, por no poder demostrar su validez general.  
3- Hechos. Predicados que se suponen verdaderos. Constituyen el KDB.  
La función del ingeniero del conocimiento proporcionar el conocimiento para poder construir la base de conocimiento y el motor de inferencias

EXPLICACIONES

MOTOR DE INFERENCIAS

KDB

EDITOR

INGENIERO

USUARIO

INTERFACE

## Lógica de Predicados y Lógica Proposicional

El instrumento fundamental de la comunicación humana es el lenguaje, formado por frases del tipo:

* Interrogativo
* Imperativo
* Declarativo

Las frases declarativas constituyen el elemento básico de descripción del conocimiento. La lógica es la disciplina que estudia los métodos de formalización del conocimiento humano. Por lo tanto la lógica estudia métodos de formalización de frases declarativas.

La Lógica de Predicados estudia las frases declarativas con mayor grado de detalle, considerando la estructura interna de las proposiciones. Toma como elementos básicos los objetos y sus relaciones. Es decir:

* **Que se afirma (predicado o relación)**
* **De quien se afirma (objeto)**

Sócrates es hombre y todos los hombres son mortales, entonces Sócrates es mortal. La Lógica Proposicional se caracteriza en relación a su composición por emplear muy pocos símbolos que pueden ser evaluados solamente como falsos o verdaderos (mediante clausulas y/o elementos atómicos), en contraposición al alfabeto lógico usado por la lógica de predicados para expresar sus formulaciones.

* La Lógica Proposicional está basada en constantes: A B C
* La lógica de predicados está basada en variables:

## Lógica de Predicados y Sistemas Expertos

Los sistemas expertos se basan en la lógica de predicados. La palabra lógica hace referencia a la posibilidad de emplear un conjunto de reglas para gestionar inferencias creíbles.

En el caso de los sistemas expertos se pretende aplicar a mundos que solo conocen en profundidad los expertos humanos. Estos expresan su conocimiento del mundo que manejan, con sus propias palabras. Estas palabras no son otra cosa que los símbolos del lenguaje natural, cuyo significado dista mucho de estar bien establecidos. Por lo tanto para que exista una chance de que un sistema experto basado en el formalismo lógico produzca inferencias correctas, se requiere primero definir los símbolos con cuales se operará, y su significado preciso.

Los sistemas expertos se basan esencialmente en la lógica a la que incorporan algunas sencillas estrategias de programación.

La lógica de 1er orden es uno de los formalismos más utilizados para representar conocimiento en IA. La lógica cuenta con un lenguaje formal mediante el cual es posible representar formulas llamadas axiomas, que permiten describir fragmentos del conocimiento, y además consta de un conjunto de reglas de inferencia que aplicadas a los axiomas, permiten derivar nuevos conocimientos.

# **LOGICA DIFUSA, pasos**

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.

1. 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.

1. 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.

1. 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.

1. Defuzzificacion o desdifusión

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.

# **BUSQUEDA**

En IA se entiende como búsqueda a un algoritmo que busca un dato en un conjunto, pero donde el objetivo es “encontrar un cierto camino” hasta encontrar el dato antes que corroborar la existencia del mismo.

**Características de los Procesos de Búsqueda**

Los procesos de búsqueda tienen sentido en problemas que reúnen una serie de características:

* 􀁸 Cabe la posibilidad de asociar un *conjunto de estados* a las diferentes situaciones en que se puede
* encontrar el objetivo del *dominio* sobre el que se define el problema.
* 􀁸 Hay una serie de *estados iniciales* desde los que empezaría el proceso de búsqueda.
* 􀁸 Existen ciertos *operadores,* tal que un operador aplicado sobre un estado producirá otro estado.
* 􀁸 Existe al menos un *estado meta* o *estado solución.*

Cualquier proceso de búsqueda persigue, asociando nodos con estados y arcos con operadores, encontrar un camino que conduzca de un nodo inicial al otro final.

Se define *Espacio de Estados* como el conjunto de estados que podrían obtenerse si se aplicaran todos los operadores posibles a todos los estados que se fueran generando.

**Métodos de Búsqueda**



**No informados (Búsqueda Exhaustiva)**

La búsqueda sin información del dominio pretende realizar una *exploración exhaustiva* del espacio de estados, dado que no hay conocimiento que pueda guiar la misma y que en principio, no deje ningún nodo sin ser examinado.

***Búsqueda Primero en Anchura***

La búsqueda Primero en Anchura recorre el árbol o grafo por niveles utilizando una estructura tipo cola (FIFO) en la que se van introduciendo los nodos generados.

**Ventajas**

* 􀁸 No queda atrapado explorando callejones sin salida
* 􀁸 Si existe una solución, garantiza que se logre encontrarla
* 􀁸 Si existen múltiples soluciones, se encuentra la solución mínima.

***Búsqueda Primero en Profundidad***

La búsqueda en profundidad se centra en expandir un único camino desde la raíz. En el caso de llegar a un “callejón sin salida” se retrocede hasta el nodo más cercano desde donde se pueda tomar una rama alternativa para poder seguir avanzando.

Utiliza una estructura pila (LIFO) que va almacenando los nodos generados.

Suele establecerse el *límite de exploración*, que marca la máxima longitud que puede alcanzar cualquier camino desde la raíz durante los procesos de búsqueda.

**Ventajas**

* 􀁸 Necesita menos menoría ya que sólo se almacenan los nodos del camino que se sigue en ese instante.
* 􀁸 Si se tienen suerte este método, puede encontrar una solución sin tener que examinar gran parte del espacio de estados.

***Búsqueda por Ramificación y Acotación (Retroceso)***

En lugar de generar todos los sucesores del estado actual, se genera un único sucesor en cada caso. La selección de un nodo a ser visitado no implica que sea sacado de la lista.

La característica de esta técnica es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no están siendo óptimas, para «podar» (acotar) esa rama del árbol y no continuar malgastando recursos y procesos en casos que se alejan de la solución óptima.

**Informados (Búsqueda Heurística)**

Una búsqueda heurística es una técnica que aumenta la eficiencia de un proceso de búsqueda agregando información que permite reducir la cantidad de combinaciones y que permite seleccionar una ruta con menores costos.

El conocimiento dependiente del dominio puede ayudar a dirigir el proceso de búsqueda de manera que sean exploradas en primer lugar aquellas trayectorias que parecen más prometedoras a la hora de conducir a un estado solución.

El conocimiento Heurístico se incorpora en las reglas de producción o con una función heurística, determinando estados deseables y asignando pesos a ciertos aspectos del problema, permitiendo estimar que tan deseable es un nodo o un camino dado.

**Las heurísticas son criterios, métodos o principios para decidir cuál de entre varias acciones promete ser la mejor para alcanzar una determinada meta.**

El uso de heurísticas nos permite guiar nuestra búsqueda de una solución, lo que nos permitirá obtener una solución más rápidamente que si utilizásemos estrategias de búsqueda a ciegas.

En problemas de búsqueda:

* 􀁸 Una heurística será una función que utilizaremos para estimar cómo de cerca estamos de la solución.
* 􀁸 Cada heurística estará diseñada para un problema de búsqueda particular.

La familia de los algoritmos informados, frente a los desinformados o por fuerza bruta, son aquellos que poseen una información extra sobre la estructura a objeto de estudio, la cual explotan para alcanzar más rápidamente su objetivo final, con un camino de costo mínimo desde el punto inicial al final.

La búsqueda informada es aquella que utiliza el conocimiento específico del problema más allá de la definición del problema en sí mismo, la cual puede encontrar soluciones de una manera más eficiente que una estrategia no informada, increíblemente ineficiente en la mayoría de los casos.

***Búsqueda Primero Mejor***

Este algoritmo, combina las ventajas de los algoritmos primero en profundidad y primero en amplitud. Sigue un camino a la vez, pero puede cambiarse a otro camino que parece más prometedor que el que está siguiendo.

En este sentido, puede considerarse que es un algoritmo que realiza su proceso de búsqueda en un grafo de tipo O, ya que todos sus ramales representan una alternativa de solución.

Para su operación, el algoritmo necesita dos listas de nodos y una función heurística que estime los méritos de cada nodo que se genere:

1. **ABIERTOS**: Es una variable que contiene los nodos que han sido generados. La función heurística ha sido aplicada a ellos, pero todavía no han sido examinados, es decir no se han generado sus sucesores. **ABIERTOS** puede considerarse como una COLA DE PRIORIDADES en la que los elementos con mayor prioridad son los que tienen los valores más prometedores, dados por la función heurística.
2. **CERRADOS**: Es una variable que contiene los nodos que han sido examinados.
3. **FUNCIÓN HEURÍSTICA**: Permite que el algoritmo busque primero por senderos que son o parecen más prometedores.

**Estrategia**

1. Se selecciona el nodo más prometedor, utilizando alguna *función heurística.*

2. Se expande el nodo elegido aplicando las reglas para generar a sus sucesores.

3. Si algún sucesor es solución el procedimiento termina.

Si no:

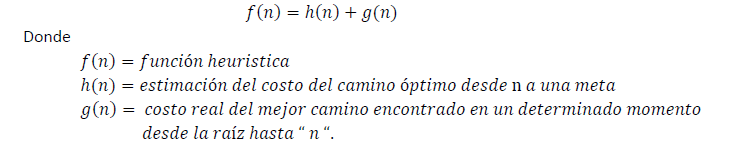
a. Se añaden los nodos a la lista.

b. Se vuelven al punto inicial.

***Búsqueda A\****

El problema de algunos algoritmos de búsqueda informados, es que se guían en exclusiva por la función heurística, la cual puede no indicar el camino de coste más bajo, o por el coste real de desplazarse de un nodo a otro, pudiéndose dar el caso de que sea necesario realizar un movimiento de coste mayor para alcanzar la solución. Es por ello que un buen algoritmo de búsqueda informada debería tener en cuenta ambos factores, el valor heurístico de los nodos y el coste real del recorrido.

Por este motivo el algoritmo de Búsqueda A\* utiliza la siguiente Función de Heurística



Además, para su operación, el algoritmo necesita dos listas de nodos y una función heurística que estime los méritos de cada nodo que se genere:

* **ABIERTOS**: Es una variable que contiene los nodos que han sido generados. La función heurística ha sido aplicada a ellos, pero todavía no han sido examinados, es decir no se han generado sus sucesores. **ABIERTOS** puede considerarse como una COLA DE PRIORIDADES en la que los elementos con mayor prioridad son los que tienen los valores más prometedores, dados por la función heurística.
* **CERRADOS**: Es una variable que contiene los nodos que han sido examinados.

**Estrategia**

1. Comenzar con ABIERTOS conteniendo solo el estado inicial

2. Hasta que se llegue a un objetivo o no queden 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, añadirlo a ABIERTOS y almacenar a su padre

ii. Si ya se ha generado antes:

1. Si el nuevo camino es mejor que el anterior, cambiar padre.

2. Actualizar el coste empleado para alcanzar el nodo y sus sucesores

# **PLANIFICACION**

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

**Planificación mediante una pila de objetivos**

Los pasos son:

1. Se plantea una pila de objetivos.

2. Se trata a los objetivos como subproblemas separados a resolver.

3. Se pregunta si el 1er objetivo se cumple, si no se cumple se debería aplicar el operador que los transforme, lo que hace es reemplazar al predicado objetivo por el operador y luego agregar a la pila las precondiciones que deben cumplirse para ese operador.

4. Para resolver el objetivo que ha quedado arriba de la pila debemos emplear un operador y, como antes, reemplazar y agregar las precondiciones.

5. Posteriormente al ir extrayendo los operadores de esa pila, se incorpora a la lista PLAN.

# **COMPLEJIDAD**

Desde el punto de vista informático, se asocia la complejidad con los recursos de cómputos requeridos para resolver un problema, es decir la complejidad operativa. La teoría de la Complejidad Computacional se ocupa de estudiar estos recursos, que son normalmente tiempo y espacio.

El objetivo de la complejidad es establecer una escala que clasifique los problemas computacionales, permitiendo establecer medidas o “clases” de complejidad.

Se pretenderá buscar algún indicador de la complejidad intrínseca de los problemas, procurando prescindir de la habilidad de la persona que deba resolverlos, de los algoritmos que puedan ser utilizados y de la característica de los medios de cálculo disponibles.

**Complejidad Temporal**

La complejidad temporal se refiere a la cantidad de intervalos o unidades elementales que demanda completar la ejecución de un proceso. La Complejidad Temporal se expresa en función del tamaño del problema o de sus datos.

Se procurará conocer la razón de crecimiento entre el indicador tiempo y la dimensión de los datos, que representa el parámetro medible. Como resultado se hablará de complejidad temporal lineal, polinómica, logarítmica, exponencial, etc. Según la naturaleza de la expresión que las vincula.

El límite de crecimiento de esta medida se denomina Complejidad Temporal Asintótica y es finalmente lo que determina el tamaño del problema que puede ser resuelto con cierto algoritmo. Tómese como ejemplo un algoritmo que procesa un conjunto de datos de entrada de tamaño “n” en un tiempo , donde C representa una constante que es propia del problema: su complejidad temporal es de orden y esto se representa como . En la forma general, se dice que una función g(n) tiene un orden de complejidad temporal si existe una constante C tal que:

para todos los posibles valores positivos de “n”.

**Se define la Complejidad Temporal como el número de unidades de tiempo requeridas para procesar una entrada de dimensión “n”.**

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 más 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.

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.

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.

Correspondencia Sub simbolica

Correspondencia numérica.

3)Una clausula de Horn es una conjunción de disyunciones como 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.

Tipos de formas clausales

- 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.

**Encadenamiento**

**Encadenamiento progresivo**

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.

**Encadenamiento progresivo**

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.