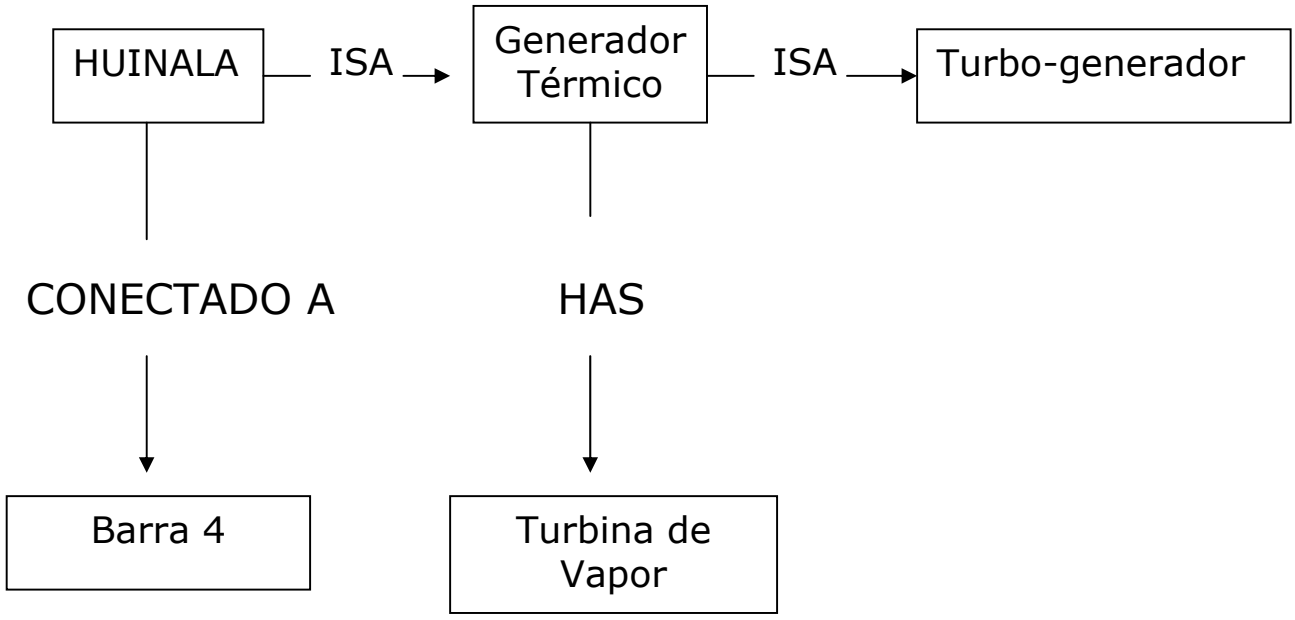
ALTERNATIVAS DE REPRESENTACIÓN DE CONOCIMIENTOS

- Redes semánticas.
- Marcos (Frames).
- Cálculo de Predicados.
- Redes Neuronales.
- Reglas de Producción.
  (if - then)

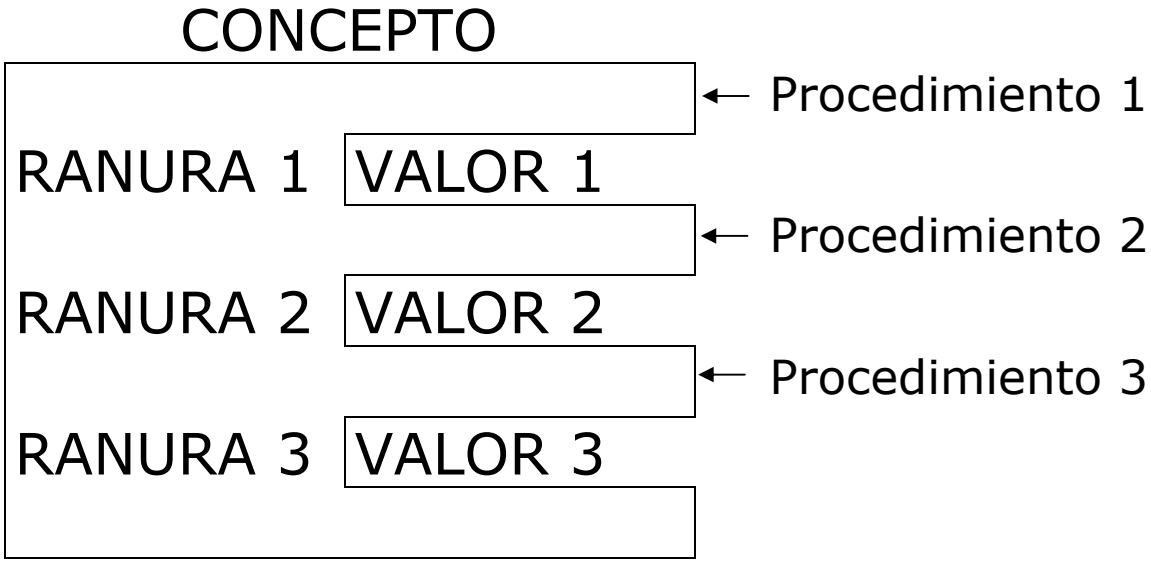
RED SEMÁNTICA

Método de representación de conocimientos que consiste en una red que representan conceptos u objetos conectados por arcos que describen las relaciones existentes entre ellos.



MARCOS (FRAMES)

Método de representación de conocimiento que asocia características con nodos que representan conceptos u objetos. Las características son descritas en términos de atributos (slots) y sus valores correspondientes.



TÉCNICAS DE EXTRACCIÓN DE CONOCIMIENTO

MÉTODO	DESCRIPCIÓN
Observación	Se observa al experto resolver el problema
Discusión del problema	Examinar datos y conocimiento para la solución del problema
Descripción del problema	El experto describe un conjunto de problemas prototipo
Análisis del problema	Analizar los pasos del razonamiento al resolver problemas reales
Refinamiento del sistema	Resolver el problema utilizando las reglas generadas por el experto
Examinación del sistema	El experto examina y critica las reglas generadas por el experto
Validación del sistema	Solución de casos en forma simultánea por el experto y el sistema

ADQUISICIÓN DE CONOCIMIENTO

Tabla de inducción

Es una forma tabular de representar un conjunto de condiciones y acciones integradas en reglas de decisión.

Condiciones

Factores que afectan la decisión

Acciones

Descripción declarativa de las acciones a realizar en dependencia del cumplimiento de las condiciones

Reglas de deducción

Descripción de las acciones a tomar en dependencia de un conjunto específico de condiciones

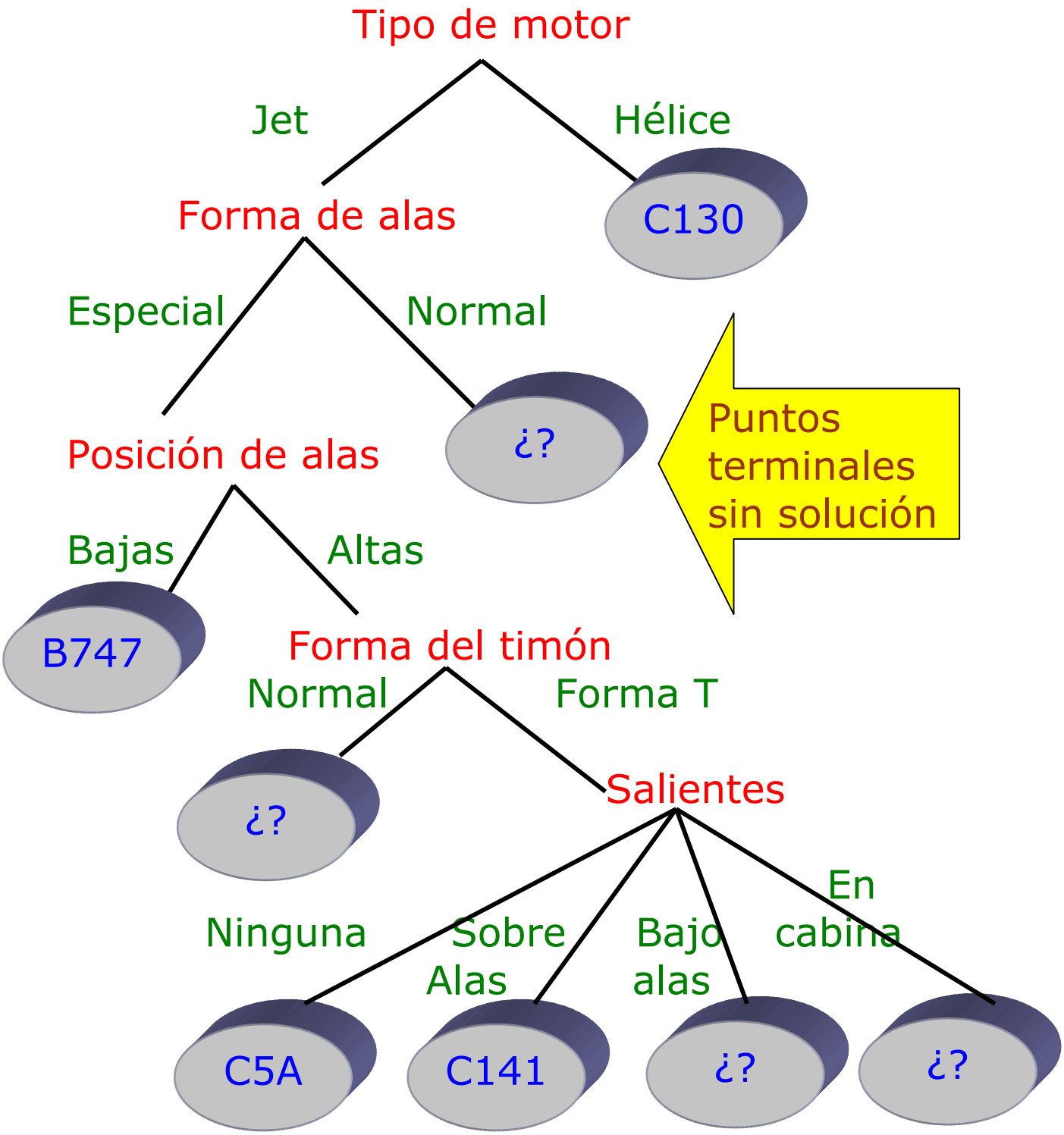
TABLA DE INDUCCIÓN

Una alternativa para la adquisición de conocimiento a través de la interfase con una persona experta es convertir una base de datos existente en un conjunto de reglas.

Ejemplo

TIPO DE AVIÓN				
Atributo	C130	C141	C5A	B747
Motor	Hélice	Jet	Jet	Jet
Alas	Altas	Altas	Altas	Bajas
Forma de alas	Normal	Especial	Especial	Especial
Forma del timón	Normal	Forma T	Forma T	Normal
Salientes	Bajo las alas	Sobre alas	Ninguna	En cabina

ÁRBOL DE DECISIÓN



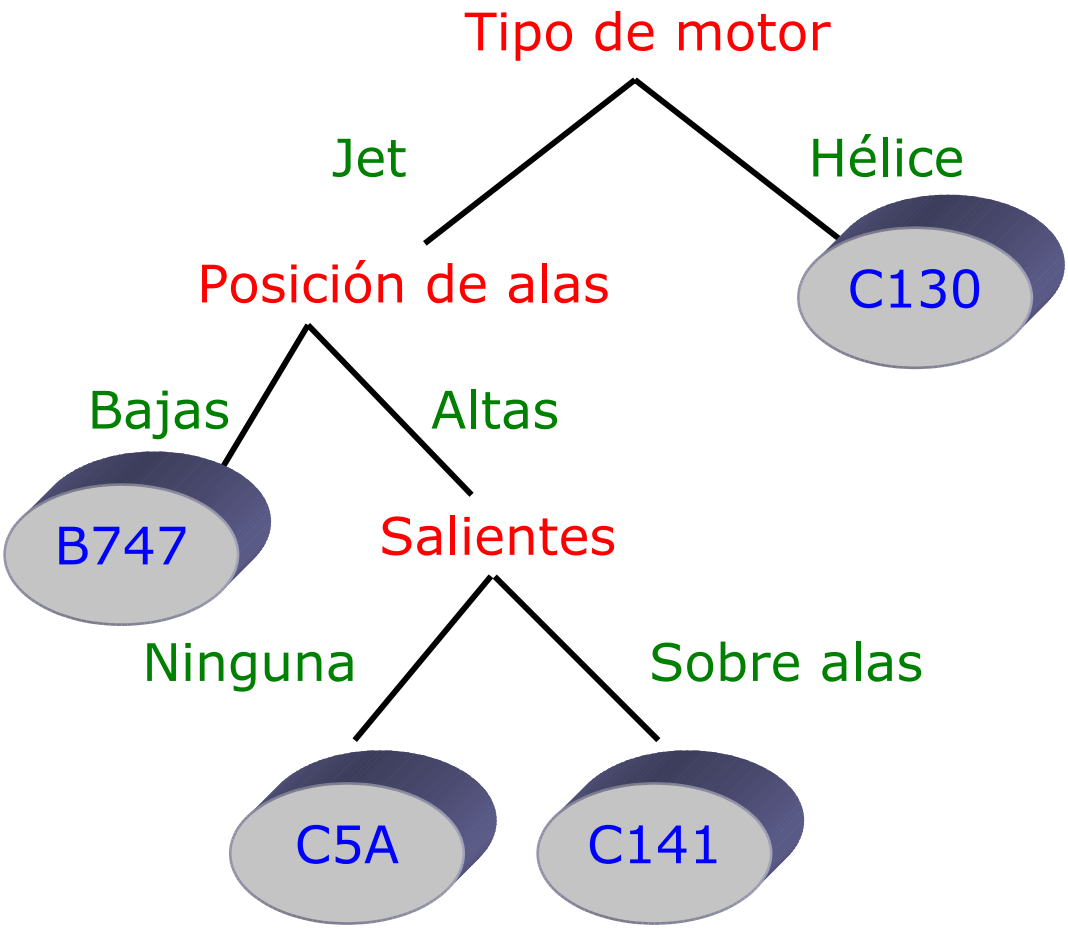
RAZONAMIENTO BASADO EN REGLAS

Una alternativa para la adquisición de conocimiento a través de la interfase con

Ejemplo de Regla ineficiente

IF **motor**=**hélice**
AND **posición_alas**=**altas**
AND **forma_alas**=**normal**
AND **forma_timón**=**normal**
AND **salientes**=**bajo alas**
THEN ***tipo_avión***=**C130**

REORDEN DE ÁRBOL DE DECISIÓN



- **Requiere menor cantidad de atributos**
- **No tiene puntos terminales sin solución**

REGLAS DE PRODUCCIÓN

Regla 1: IF **motor**=**hélice**
THEN ***tipo_avión***=**C130**

Regla 2: IF **motor**=**jet**
AND **posición_alas**=**bajas**
THEN ***tipo_avión***=**B747**

Regla 3: IF **motor**=**jet**
AND **posición_alas**=**altas**
AND **salientes**=**ninguna**
THEN ***tipo_avión***=**C5A**

Regla 4: IF **motor**=**jet**
AND **posición_alas**=**altas**
AND **salientes**=**sobre alas**
THEN ***tipo_avión***=**C141**

```
! AVION.KBS
! Sistema Experto para la identificación de aviones
! Autor: Ing. Bruno Lopez Takeyas
! Desarrollado en VPX
!runtime;  !Clausula para desaparecer las ventanas de
depuracion

ACTIONS

 DISPLAY "Sistema Experto para la identificacion de aviones"
 find tipo_avion
 color=8
 display "El tipo de avion es {tipo_avion}";

RULE 1
  IF tipo_motor=jet AND posicion_alas=bajas
  THEN tipo_avion=B747
  color = 4
  DISPLAY "El avion es un Boeing 747 debido a que es un JET
con alas BAJAS";

rule 2
 if tipo_motor=helice
 then tipo_avion=C130
 color = 4
 display "El avion es un C130 porque tiene HELICE";

rule 3
 if tipo_motor=jet and posicion_alas=altas and
salientes=ninguna
 then tipo_avion=C5A
 color=4
 display "El avion es un C5A porque es un JET con alas ALTAS
y NO tiene salientes";

rule 4
 if tipo_motor=jet and posicion_alas=altas and
salientes=sobre_alas
 then tipo_avion=C141
 color=4
 display "El avion es un C141 porque es un JET con alas
ALTAS y salientes SOBRE LAS ALAS";

ask tipo_motor:"Seleccione el tipo de motor:";
choices tipo_motor: Jet, Helice;

ask posicion_alas:"Seleccione la posicion de las alas:";
choices posicion_alas: Bajas, Altas;

ask salientes: "Seleccione el tipo de salientes: ";
choices salientes: Ninguna, Sobre_alas;
```

```
% AVION.PRO
%
% Sistema Experto para la identificacion de aviones
% Autor: Ing. Bruno Lopez Takeyas

% Desarrollado en Amzi Prolog
% ============================================================

main :- identify.

identify:-
  retractall(known(_,_,_)),   % Limpia la informacion
previamente almacenada
  write('SISTEMA EXPERTO PARA LA IDENTIFICACION DE
AVIONES'),nl,
  write('Por: Ing. Bruno Lopez Takeyas'),nl,nl,nl,
  write('INSTRUCCIONES:'),nl,
  write('Seleccione una opcion de cada menu y coloque un
punto al final'),nl,

  tipo_avion(X),
  write('El tipo de avion es '),write(X),nl.
identify:-
  write('No se puede identificar el tipo de avion'),nl.

% ============================================================
% Reglas :
% ============================================================

tipo_avion(b747):-
   tipo_motor(jet),
   posicion_alas(bajas).

tipo_avion(c130):-
  tipo_motor(helice).

tipo_avion(c5a):-
  tipo_motor(jet),
  posicion_alas(altas),
  salientes(ninguna).

tipo_avion(c141):-
  tipo_motor(jet),
  posicion_alas(altas),
  salientes(sobre_alas).

% ============================================================
% Seleccion de opciones de los atributos
% ============================================================

tipo_motor(X):-menu_opciones(motor,X,[jet,helice]).

posicion_alas(X):-menu_opciones(alas,X,[bajas,altas]).
```

```
salientes(X):-
menu_opciones(salientes,X,[ninguna,sobre_alas]).

% ============================================================
% Menu de Seleccion
% ============================================================

menu_opciones(Atributo,Value,_):-
  known(yes,Atributo,Value),     % seleccion correcta !
  !.
menu_opciones(Atributo,_,_):-
  known(yes,Atributo,_),         % falla con cualquier otra
opcion
  !, fail.

menu_opciones(Atributo,ValorSeleccionado,Menu):-
  nl,write('Seleccione '),write(Atributo),write(' ...'),nl,
  display_menu(Menu),
  write('Opcion ? '),
  read(Num),nl,
  pick_menu(Num,Respuesta,Menu),
  asserta(known(yes,Atributo,Respuesta)),
  ValorSeleccionado = Respuesta.

display_menu(Menu):-
  disp_menu(1,Menu), !.

disp_menu(_,[]).
disp_menu(N,[Item | Rest]):-            % Despliega
recursivamente el primer elemento de la lista de opciones
  write(N),write('.- '),write(Item),nl,    % y disp_menu
muestra el ultimo
  NN is N + 1,
  disp_menu(NN,Rest).

pick_menu(N,Val,Menu):-
  integer(N),                      % Valida la captura de un
numero
  pic_menu(1,N,Val,Menu), !.       % Inicia con 1
pick_menu(Val,Val,_).              % Si no se captura un
numero, usa el valor capturado

pic_menu(_,_,ninguna_de_anteriores,[]).  % Si se agota la
lista
pic_menu(N,N, Item, [Item|_]).       % Contador de las
opciones
pic_menu(Ctr,N, Val, [_|Rest]):-
  NextCtr is Ctr + 1,                % Sig. opcion
  pic_menu(NextCtr, N, Val, Rest).
```

BÚSQUEDA DE SOLUCIONES

Lleva a cabo una selección de entre distintas alternativas de solución en un Espacio de Estados.

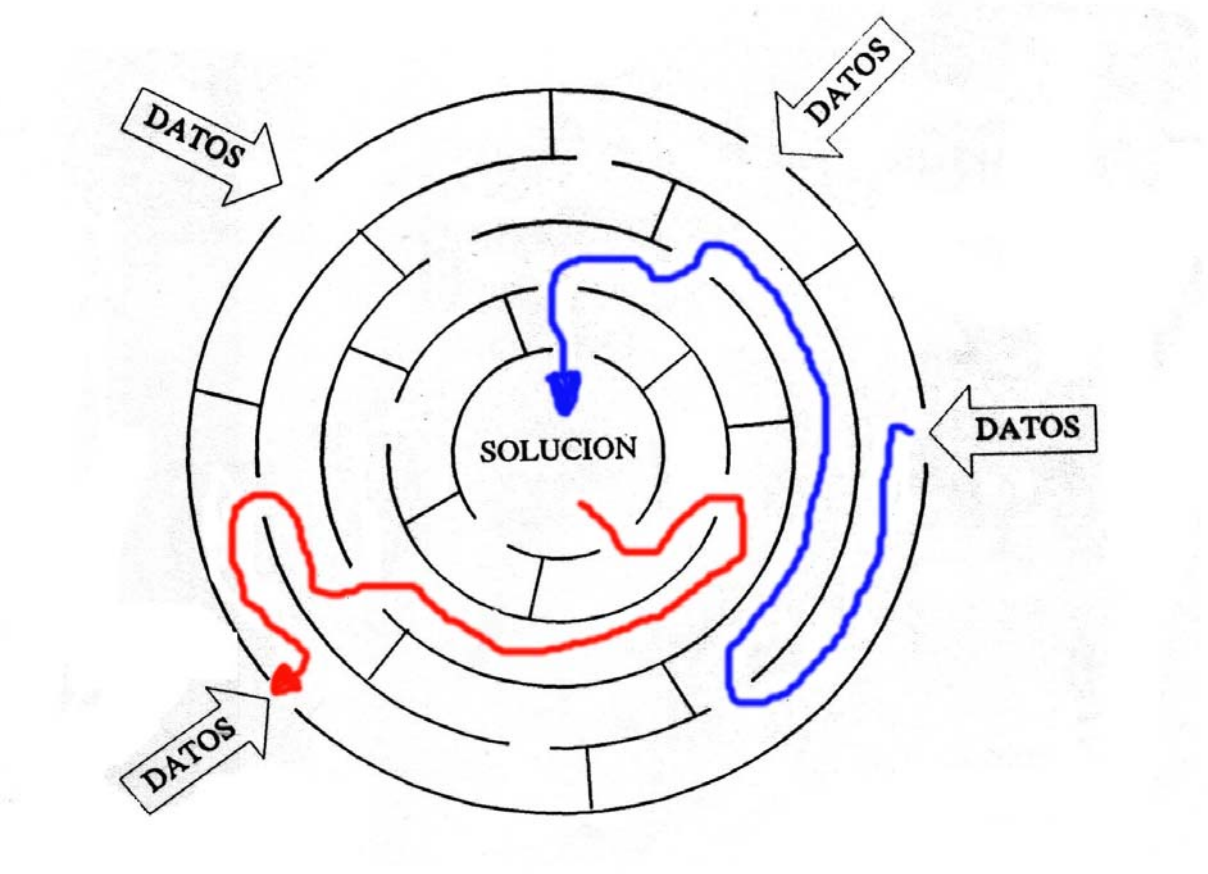
Encadenamiento de reglas hacia adelante

Se parte de la información inicial del problema para establecer una posible solución.

Encadenamiento de reglas hacia atrás

Se parte de una solución establecida para el problema y se trata de verificar o deshechar.

ENCADENAMIENTO HACIA DELANTE Y HACIA ATRÁS



Hacia adelante

Hacia atrás

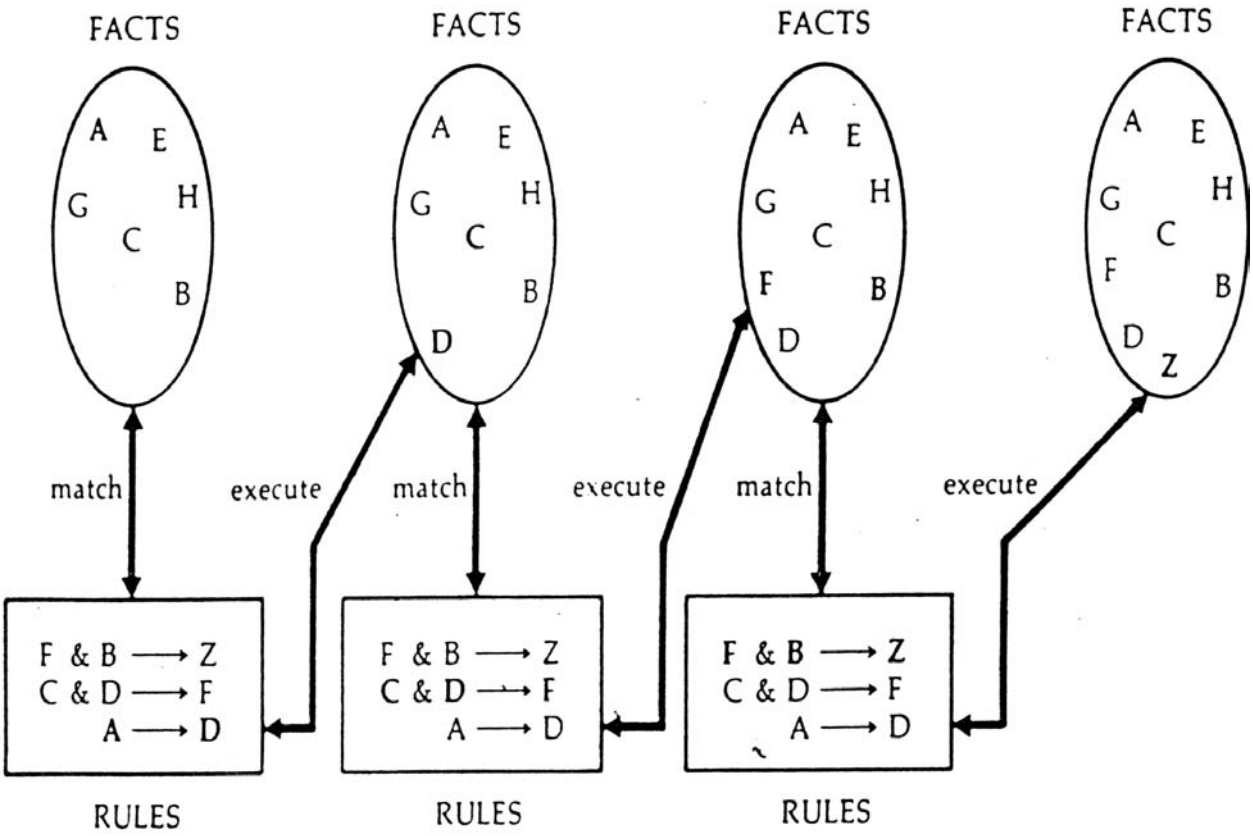
ENCADENAMIENTO HACIA ADELANTE (FORWARD CHAINING)

Método de inferencia que realiza comparaciones entre las reglas y los hechos disponibles de manera que se establezcan nuevos hechos hasta llegar al objetivo deseado.

BASE DE CONOCIMIENTOS	DATOS DEL PROBLEMA
F & B → Z	**A, G, C, E, H, B**
C & D → F	
D & W → S2	Posibles soluciones:
A → D	**Z, S2**
1) A → D	A, G, C, E, H, B, **D**
2) C & D → F	A, G, C, E, H, B, **D, F**
3) F & B → Z	A, G, C, E, H, B, **D, F, Z**

Este mecanismo de búsqueda se ejecuta en lenguajes declarativos, NO en lenguajes procedimentales !!!
Esto significa que cuando se cumple una búsqueda, se reinicia el proceso

ENCADENAMIENTO HACIA ADELANTE (FORWARD CHAINING)



ENCADENAMIENTO HACIA ADELANTE (FORWARD CHAINING)

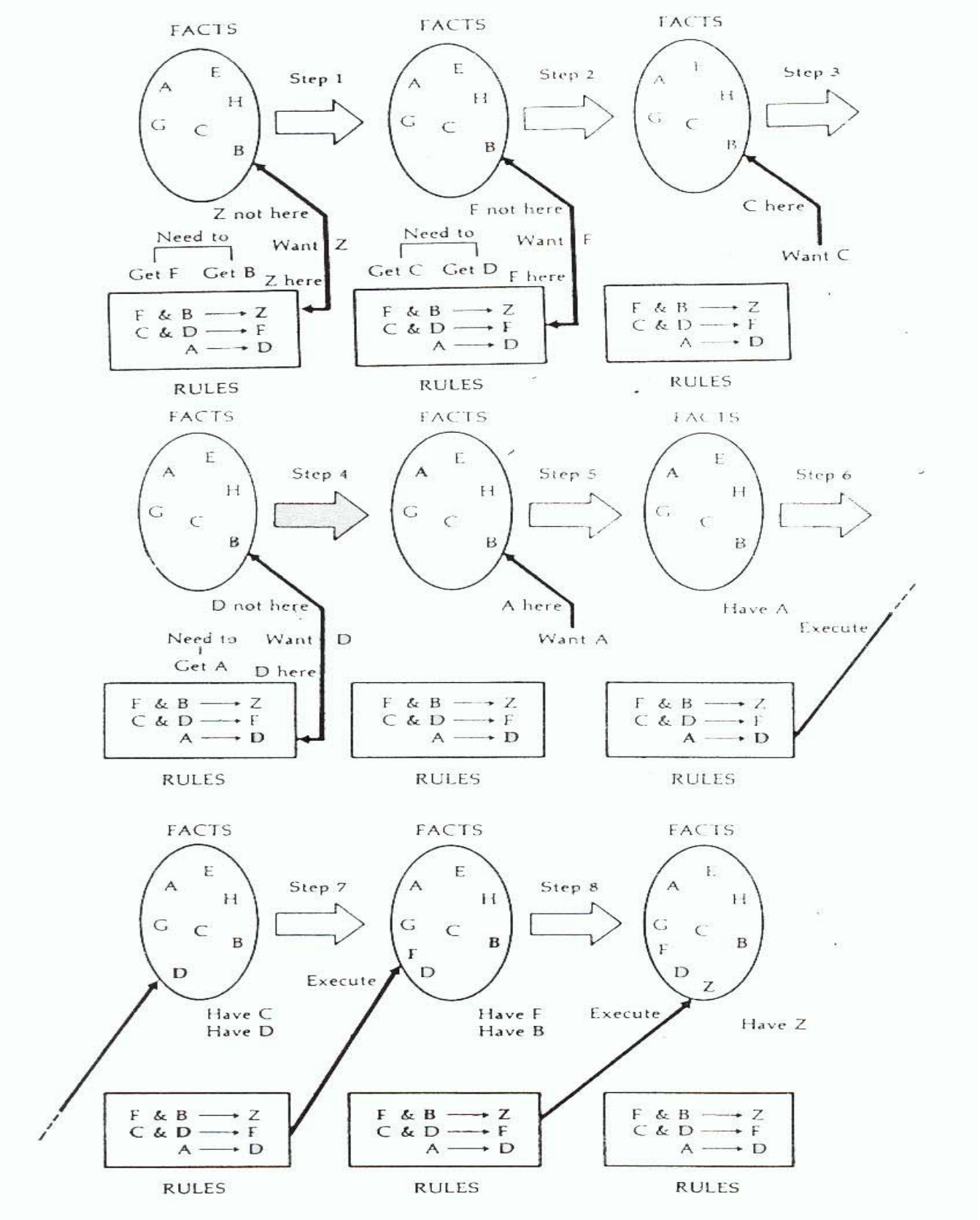
Otro ejemplo:

BASE DE CONOCIMIENTOS	DATOS DEL PROBLEMA
IF A AND E THEN W IF D AND W THEN A IF F AND B THEN Z IF A AND G THEN D IF W AND B THEN K IF H AND W THEN Z IF C AND D THEN F IF K AND G THEN M	**A, E, H, G, C, B.** Posibles soluciones **F, Z**
1) IF A AND E THEN W 2) IF A AND G THEN D 3) IF W AND B THEN K 4) IF H AND W THEN Z	**A, E, H, G, C, B, W** **A, E, H, G, C, B, W, D** **A, E, H, G, C, B, W, K** **A, E, H, G, C, B, W, K, Z**

ENCADENAMIENTO HACIA ATRÁS (BACKWARD CHAINING)

Método de inferencia que inicia con la conclusión que se desea demostrar y procura establecer la certeza de los hechos que conducen a ella.

BASE DE CONOCIMIENTOS	DATOS DEL PROBLEMA
F & B → Z	**A, G, C, E, H, B**
C & D → F	
D & W → S2	Posibles soluciones:
A → D	**Z, S2**
1) F & B → Z	No se conoce **F**
2) C & D → F	No se conoce **D**
3) A → D	A, G, C, E, H, B, **D**
4) C & D → F	A, G, C, E, H, B, **D**, **F**
5) F & B → Z	A, G, C, E, H, B, **D**, **F**, **Z**


FACTS
Step 1
Z not here
Need to
Get F Get B
Want Z
Z here
F & B → Z
C & D → F
A → D
RULES
FACTS
Step 2
F not here
Need to
Get C Get D
Want F
F here
F & B → Z
C & D → F
A → D
RULES
FACTS
Step 3
C here
Want C
F & B → Z
C & D → F
A → D
RULES
FACTS
Step 4
D not here
Need to
Get A
Want D
D here
F & B → Z
C & D → F
A → D
RULES
FACTS
Step 5
A here
Want A
F & B → Z
C & D → F
A → D
RULES
FACTS
Step 6
Have A
Execute
F & B → Z
C & D → F
A → D
RULES
FACTS
Step 7
Have C
Have D
Execute
F & B → Z
C & D → F
A → D
RULES
FACTS
Step 8
Have F
Have B
Execute
F & B → Z
C & D → F
A → D
RULES
FACTS
Have Z
F & B → Z
C & D → F
A → D
RULES

ENCADENAMIENTO HACIA ATRÁS (BACKWARD CHAINING)

Otro ejemplo:

BASE DE CONOCIMIENTOS	DATOS DEL PROBLEMA
D & W → S2	**A, G, C, E, H, B**
F & B → Z	
C & D → F	Posibles soluciones:
A → D	**Z, S2**
1) D & W → S2	No se conoce **D**
2) A → D	A, G, C, E, H, B, **D**
3) D & W → S2	No se conoce **W (pero no es solución)**
4) F & B → Z	No se conoce **F**
5) C & D → F	A, G, C, E, H, B, D, **F**
6) F & B → Z	A, G, C, E, H, B, D, F, **Z**

Nótese que se realiza una búsqueda recursiva de soluciones de acuerdo a la colocación de las reglas de producción

PAQUETES DE SOFTWARE

- 1st-CLASS
- ACQUIRE

(http://vvv.com/ai/acquire/acquire.html)

- Rule Master
- G2
- Nexpert Object
- Smart Elements
- Level 5
- VP-Expert
- K-Vision

ENLACES DE INTERNET

- Instituto de Ciencias del Conocimiento de la Universidad de Calgary (http://ksi.cpsc.ucalgary.ca/KSI/KSI.html)
- Razonamiento basado en casos de la Universidad de Kaiserslauten (http://wwwagr.informatik.uni-kl.de/~Isa/CBR/CBR-Homepage.html)
- Universidad de Maryland Baltimore (http://www.cs.umbc.edu/agents/)
- Laboratorios Daxtron (http://www.polaris.net/~daxtron/mailbot.htm)
- Razonamiento cualitativo de MIT (http://www.context.mit.edu)
- Desarrollos de la Universidad de Texas en Austin (http://www.cs.utexas.edu/users/qr/)
- Universidad de NorthWestern (http://multivac.ils.nwu.edu)
- Demos del Autor Efraim Turban (http://www.prenhall.com/turban)