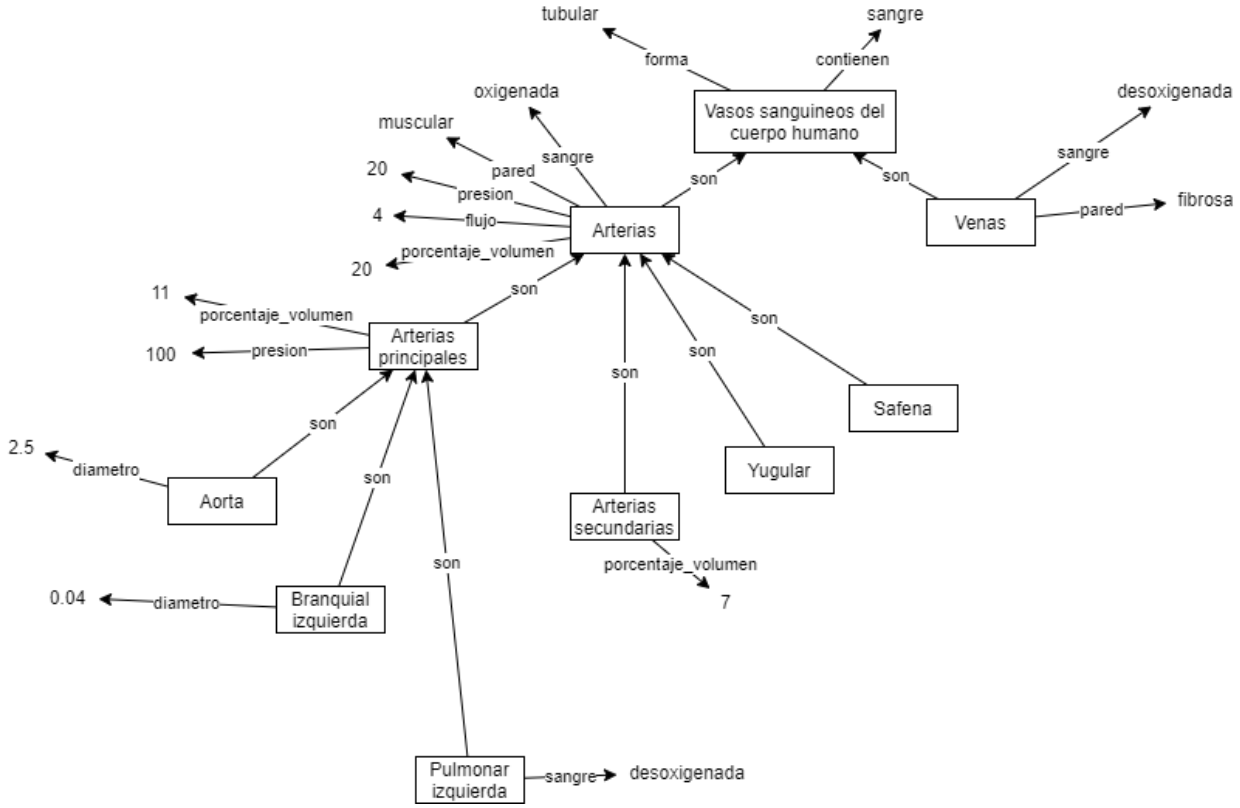
2do Parcial, Inteligencia Artificial 18/11/2021
Tobías Demoor

1.

a.



2do Parcial, Inteligencia Artificial 18/11/2021
Tobías Demoor

FRAME: Vasos sanguineos del cuerpo humano
forma: tubular
contienen: sangre

FRAME: Arterias
clase_de: Vasos sanguineos del cuerpo humano
sangre: oxigenada
pared: muscular
presion: 20
flujo: 4
procentaje_volumen: 20

FRAME: Venas
clase_de: Vasos sanguineos del cuerpo humano
sangre: desoxigenada
pared: fibrosa

FRAME: Arterias principales
clase_de: Arterias
presion: 100
procentaje_volumen: 11

FRAME: Arterias secundarias
clase_de: Arterias
procentaje_volumen: 7

FRAME: Aorta
clase_de: Arterias principales
diametro: 2.5

FRAME: Branquial izquierda
clase_de: Arterias principales
diametro: 0.04

FRAME: Pulmonar izquierda
clase_de: Arterias principales
sangre: desoxigenada

FRAME: Yugular
clase_de: Arterias

FRAME: Safena
clase_de: Arterias

b.

```
valor(Frame, Slot, Val):- valorSuper(Frame, Frame, Slot, Val).


valorSuper(Frame, SuperFrame, Slot, Val):-
    Consulta=..[ SuperFrame, Slot, Informacion ],
    call(Consulta), proceso(Informacion, Frame, Val),!.

valorSuper(Frame, SuperFrame, Slot,Val):-
    padre(SuperFrame, PadreSuperFrame),
    valorSuper(Frame, PadreSuperFrame, Slot, Val).

padre(Frame,PadreFrame):-
    (Consulta=..[Frame, clase_de, PadreFrame] ; Consulta=..[Frame, instancia_de,
PadreFrame]),
    call(Consulta).

proceso( ejecutar(Meta, Frame, Val), Frame, Val):- !, call(Meta).
proceso(Val, _, Val).

vasos_sanguineos_humanos(forma, tubular).
vasos_sanguineos_humanos(contienen, sangre).

arterias(clase_de, vasos_sanguineos_humanos).
arterias(sangre, oxigenada).
arterias(pared, muscular).
arterias(presion, 20).
arterias(flujo, 4).
arterias(porcentaje_volumen, 20).
arterias(resistencia, ejecutar(resistencia(Obj, Val),Obj, Val)).

resistencia(Objeto, Val):-
    valor(Objeto, presion, Presion),
    valor(Objeto, flujo, Flujo),
    Val is Presion/Flujo.

venas(clase_de, vasos_sanguineos_humanos).
venas(sangre, desoxigenada).
venas(pared, fibrosa).

arterias_principales(clase_de, arterias).
arterias_principales(presion, 100).
arterias_principales(porcentaje_volumen, 11).

arterias_secundarias(clase_de, arterias).
```

```
arterias_secundarias(porcentaje_volumen, 7).

aorta(clase_de, arterias_principales).
aorta(diametro, 2.5).

branquial_izquierda(clase_de, arterias_principales).
branquial_izquierda(diametro, 0.04).

pulmonar_izquierda(clase_de, arterias_principales).
pulmonar_izquierda(sangre, desoxigenada).

yugular(clase_de, arterias).

safena(clase_de, arterias).
```

c. (se agrega lo siguiente al final del código anterior)

```
aorta_juan(instancia_de, aorta).

aorta_juan(presion, 120).
aorta_juan(flujo, 5).

% valor(aorta_juan, resistencia, R).
% R = 24
```

2.

a. (para la consulta se ejecuta lo siguiente y el resultado es retornado en Plan

plan([posR1(h1), posR2(h3), libre(h2), libre(p), libre(h4)], [posR1(h3), posR2(h4)], Plan, E).)

```
plan(Estado, Metas, [], Estado):- satisfecho(Estado, Metas).

plan(Estado, Metas, Plan, EstadoFinal):-
    append(PrePlan, [Accion|PostPlan], Plan),
    seleccionar(Estado, Metas, Meta),
    logra(Accion, Meta),
    puede(Accion, Condicion),
    plan(Estado, Condicion, PrePlan, EstadoMedio1),
    aplica(EstadoMedio1, Accion, EstadoMedio2),
    plan(EstadoMedio2, Metas, PostPlan, EstadoFinal).

% verifica si una estado satisface todas las metas
satisfecho(_, []).
satisfecho(Estado, [Meta|Metas]):-
    member(Meta, Estado),
    satisfecho(Estado, Metas).

% selecciono una meta de metas si no está ya en estado
seleccionar(Estado, Metas, Meta):-
    member(Meta, Metas),
    not(member(Meta, Estado)).

% verifica si una acción logra una meta
logra(Accion, Meta):-
    agregar(Accion, Metas),
    member(Meta, Metas).

% aplica una acción a un estado y retorna el nuevo estado
aplica(Estado, Accion, NuevoEstado):-
    eliminar(Accion, ListaDel),
    borrar(Estado, ListaDel, Estado1),!,
    agregar(Accion, ListaAgr),
    append(ListaAgr, Estado1, NuevoEstado).

% elimina una lista de elementos de un estado
borrar([],_,[]).
borrar([X|L1],L2,Dif):-
    member(X,L2),!,
    borrar(L1,L2,Dif).
borrar([X|L1],L2,[X|Dif]):-
    borrar(L1,L2,Dif).
```

```
% ------------------------------------------------------------
% Definidos para el problema particular
% ------------------------------------------------------------

% indica que se puede realizar una condición dadas ciertas condiciones
puede(moverR1(Desde,Hasta), [posR1(Desde), libre(Hasta)]):- habitacion(Desde),
habitacion(Hasta), (conectadas(Desde, Hasta); conectadas(Hasta, Desde)).
puede(moverR2(Desde,Hasta), [posR2(Desde), libre(Hasta)]):- habitacion(Desde),
habitacion(Hasta), (conectadas(Desde, Hasta); conectadas(Hasta, Desde)).

% modifica el estado al hacer una acción
agregar(moverR1(Desde, Hasta), [libre(Desde), posR1(Hasta), t(2)]).
agregar(moverR2(Desde, Hasta), [libre(Desde), posR2(Hasta), t(5)]).

% modifica el estado al deshacer una acción
eliminar(moverR1(Desde, Hasta), [libre(Hasta), posR1(Desde)]).
eliminar(moverR2(Desde, Hasta), [libre(Hasta), posR2(Desde)]).

% hechos
habitacion(h1).
habitacion(h2).
habitacion(h3).
habitacion(h4).
habitacion(p).

conectadas(h1, h2):-!.
conectadas(h1, p):-!.
conectadas(h2, p):-!.
conectadas(h3, p):-!.
conectadas(h4, p):-!.
conectadas(h3, h4):-!.
```

b. (se agrega al final del código anterior, la consulta es la siguiente y el valor se retorna en Tiempo

```
incisob([posR1(h1), posR2(h3), libre(h2), libre(p), libre(h4)],  [posR1(h3),
posR2(h4)], Plan, Tiempo).)
```

```
sumaT([], 0).

sumaT([H|R], S):- sumaT(R, S1), S is S1 + H.

incisob(Estado, Metas, Plan, Tiempo):- plan(Estado, Metas, Plan, EstadoFinal),
findall(X, member(t(X), EstadoFinal), L), sumaT(L, Tiempo).
```

2do Parcial, Inteligencia Artificial 18/11/2021
Tobías Demoor

3. a.
i.

¿Declara habilidades no comprobables?	No, entonces parece ser honesto
¿Está dispuesto a conversar con un ejecutivo?	Si, entonces es capaz de mantener una conversación
¿Parece ser honesto?	Si, entonces contesta firmemente
¿Contesta firmemente?	Si, entonces tiene facilidad de palabra
¿Es capaz de mantener una conversación y tiene facilidad de palabra?	Si, entonces es cordial
¿Es cordial?	Si, entonces tiene habilidades interpersonales
¿Tiene habilidades interpersonales?	Si, entonces **le corresponde el empleo**

ii.

Le corresponde el empleo si	¿Tiene habilidades interpersonales?
Tiene habilidades interpersonales si	¿Es cordial?
Es cordial si	¿Es capaz de mantener una conversación y tiene facilidad de palabra?
Es capaz de mantener una conversación si	¿Está dispuesto a conversar con un ejecutivo? NO, el aspirante no lo está por lo tanto **no le correponde el empleo**

iii. (supongo “capaz de establecer una conversación” equivalente a “capaz de mantener una conv”)

¿Parece ser honesto?	Si, entonces contesta firmemente
¿Contesta firmemente?	Si, entonces tiene facilidad de palabra
¿Es capaz de mantener una conversación y tiene facilidad de palabra?	Si, entonces es cordial
¿Es cordial?	Si, entonces **tiene habilidades interpersonales**

b.

```
; Si el aspirante contesta firmemente entonces el aspirante tiene facilidad de
palabra

(defrule regla1
    (constesta_firmemente si)
    => (assert (facilidad_de_palabra si))
)

; Si el aspirante parece ser honesto contesta firmemente
(defrule regla2
    (parece_honesto si)
    => (assert (constesta_firmemente si))
)

; Si el aspirante declara habilidades no comprobables entonces el aspirante
noparece ser honesto sino el aspirante parece ser honesto
(defrule regla3_1
```

```
    (declara_habilidades_no_comprobables si)
    => (assert (parece_honesto no))
)

(defrule regla3_2
    (declara_habilidades_no_comprobables no)
    => (assert (parece_honesto si))
)

; Si el aspirante está dispuesto a conversar con un ejecutivo entonces aspirante
es capaz de mantener una conversación
(defrule regla4
    (dispuesto_conversar_con_ejecutivo si)
    => (assert (capaz_de_mantener_conver si))
)

; Si el aspirante es capaz de mantener una conversación y tiene facilidad palabra
entonces el aspirante es cordial
(defrule regla5
    (capaz_de_mantener_conver si) (facilidad_de_palabra si)
    => (assert (cordial si))
)

; Si aspirante es cordial entonces el aspirante tiene habilidades interpersonales
(defrule regla6
    (cordial si)
    => (assert (habilidades_interpersonales si))
)

; Si aspirante tiene habilidades interpersonales entonces le corresponde el
empleo
(defrule regla7
    (habilidades_interpersonales si)
    => (assert (corresponde_el_empleo si))
)

(defrule exito
    (corresponde_el_empleo si)
    => (printout t "Al aplicante le corresponde el empleo" crlf)
)

(deffacts inciso_i
    (declara_habilidades_no_comprobables no)
    (dispuesto_conversar_con_ejecutivo si)
)
```