Paradigma Lógico – functores, negación y forall.

Ejercicio 1

En una empresa multinacional se da servicio de outsorcing a los servidores de varias empresas. De cada servidor se conoce su ubicación en una fila de servidores del Centro de Cómputos, y qué criticidad tiene asociada en el contrato de outsorcing. La criticidad (del 1 al 4) define qué tan rápido se tiene que dar solución a los problemas que se presentan en el server.
En el Centro de Monitoreo de la empresa se reciben las alertas de los problemas que acontence en el Centro de Cómputos. Los eventos que se informa son:

- cortes de luz que afectan toda una fila de servidores
- rebooteos de servers
- "cuelgues" de aplicaciones en un server.

Escribir las cláusulas que hagan falta para definir los predicados requiereAtencionNormal/2 y requiereAtencionInmediata/2, sabiendo que

- ante un corte de luz, todos los servidores de la fila requieren atención inmediata
- ante un rebooteo de server, el server afectado y los otros servidores que sean clientes de ese server requieren atención normal.
- ante un cuelgue de aplicación, el servidor afectado requiere atención inmediata si es crítico (criticidad 1 y 2), o atención normal si el servidor no es crítico (criticidad 3 y 4). Ningún otro servidor requiere atención.

Para los dos predicados, el primer argumento es el servidor del cual se quiere saber si requiere o no atención, y el segundo es un evento; usar functores para distinguir entre distintos eventos.
Lograr para ambos predicados la inversibilidad necesaria para que pueda consultarse qué servidores requieren atención dado un evento.

Los datos de los servidores se pueden modelar así:

```
servidor(PS1, fila1, criticidad1).
servidor(PS2, fila1, criticidad2).
servidor(WAS1, fila2, criticidad1).
servidor(WAS1_2, fila2, criticidad4).
servidor(FS_X48, fila2, criticidad1).
esCliente(PS1,FS_X48).   % acá dice que el servidor PS1 es cliente de FS_X48
esCliente(WAS1,FS_X48).  % acá dice que el servidor WAS1 es cliente de FS_X48
% etc...
```

En este ejemplo:

- ante un corte de luz en la fila1, los servidores PS1 y PS2 requieren atención inmediata.
- ante un rebooteo de FS_X48, tanto ese server como PS1 y WAS1 requieren atención normal.
- ante un cuelgue en PS2, ese server requiere atención inmediata y nadie más.
- ante un cuelgue en WAS1_2, ese server requiere atención normal y nadie más.

Ejercicio 2

Se organiza un juego que consiste en ir buscando distintos objetos por el mundo. Cada participante está en un determinado nivel, cada nivel implica ciertas tareas, cada tarea consiste en buscar un objeto en una ciudad.

Representamos las tareas como functores buscar(Cosa,Ciudad), y definimos el predicado tarea/2 de esta forma:

```
tarea(basico,buscar(libro,jartum)).
tarea(basico,buscar(arbol,patras)).
tarea(basico,buscar(roca,telaviv)).
tarea(intermedio,buscar(arbol,sofia)).
tarea(intermedio,buscar(arbol,bucarest)).
tarea(avanzado,buscar(perro,bari)).
tarea(avanzado,buscar(flor,belgrado)).
```

o sea, si estoy en el nivel básico, mis tareas posibles son buscar un libro en Jartum, un árbol en Patras o una roca en Tel Aviv.

Para definir en qué nivel está cada participante se define el predicado nivelActual/2, de esta forma:

```
nivelActual(pepe,basico).
nivelActual(lucy,intermedio).
nivelActual(juancho,avanzado).
```

También vamos a necesitar saber qué idioma se habla en cada ciudad, qué idiomas habla cada persona, y el capital actual de cada persona. Esto lo representamos con los predicados idioma/2, habla/2 y capital/2:

```
idioma(alejandria,arabe).
idioma(jartum,arabe).
idioma(patras,griego).
idioma(telaviv,hebreo).
idioma(sofia,bulgaro).
idioma(bari,italiano).
idioma(bucarest,rumano).
idioma(belgrado,serbio).

habla(pepe,bulgaro).
habla(pepe,griego).
habla(pepe,italiano).
habla(juancho,arabe).
habla(juancho,griego).
habla(juancho,hebreo).
habla(lucy,griego).

capital(pepe,1200).
capital(lucy,3000).
capital(juancho,500).
```

Definir los siguientes predicados:
a. destinoPosible/2 e **idiomaUtil/2**.
destinoPosible/2 relaciona personas con ciudades; una ciudad es destino posible para un nivel si alguna tarea que tiene que hacer la persona (dado su nivel) se lleva a cabo en la ciudad. P.ej. los destinos posibles para Pepe son: Jartum, Patras y Tel Aviv.
idiomaUtil/2 relaciona niveles con idiomas: un idioma es útil para un nivel si en alguno de los destinos posibles para el nivel se habla el idioma. P.ej. los idiomas útiles para Pepe son: árabe, griego y hebreo.

b. excelenteCompaniero/2, que relaciona dos participantes. P2 es un excelente compañero para P1 si habla los idiomas de todos los destinos posibles del nivel donde está P1.
P.ej. Juancho es un excelente compañero para Pepe, porque habla todos los idiomas de los destinos posibles para el nivel de Pepe.
Asegurar que el predicado sea inversible para los dos parámetros.

c. interesante/1: un nivel es interesante si se cumple alguna de estas condiciones
- Todas las cosas posibles para buscar en ese nivel están vivas (las cosas vivas en el ejemplo son: árbol, perro y flor).
- En alguno de los destinos posibles para el nivel se habla italiano.
- Alguno de los participantes tiene un capital de al menos 2500

Asegurar que el predicado sea inversible.

d. complicado/1: un participante está complicado si: no habla ninguno de los idiomas de los destinos posibles para su nivel actual; está en un nivel distinto de básico y su capital es menor a 1500, o está en el nivel básico y su capital es menor a 500.

e. homogeneo/1: un nivel es homogéneo si en todas las opciones la cosa a buscar es la misma. En el ejemplo, el nivel intermedio es homogéneo, porque en las dos opciones el objeto a buscar es un árbol.

En general: es válido agregar los predicados necesarios para poder garantizar inversibilidad o auxiliares para resolver cada ítem, y usar en un ítem los predicados definidos para resolver ítems anteriores.

Ejercicio 3

Para un proyecto de desarrollo de software se tiene una base de conocimientos con los siguientes hechos:

tarea/3, que relaciona una tarea con su duración y un rol para realizarla.
```
tarea(login, 80, programador).
tarea(cacheDistribuida, 120, arquitecto).
tarea(pruebasPerformance, 100, tester).
tarea(tuning, 30, arquitecto).
```

predece/2, que relaciona dos tareas A y B indicando que la tarea B no puede comenzar hasta que la A no haya terminado.
```
precede(cacheDistribuida, pruebasPerformance).
precede(pruebasPerformance, tuning).
```

realizada/1, que indica las tareas que ya están terminadas.
```
realizada(login).
```

Se pide:

1. Definir el predicado anterior/2 que relaciona dos tareas A y B si estas deben realizarse en un orden específico, ya sea por una dependencia directa o indirecta. Por ejemplo:
?- anterior(T, tuning).
T = pruebasPerformance; (-> dependencia directa, porque pruebasPerformance precede a tuning)
T = cacheDistribuida; (-> dependencia indirecta, a través de pruebasPerformance)

2. Definir los predicados simple/1 y riesgo/1 de forma que:
- Las tareas que realizan los programadores y testers son consideradas simples. También se consideran simples todas las tareas que toman menos de una semana (40 horas).

- Consideramos como riesgos a todas las tareas de más de 40 horas que aún no han sido realizadas.

Por ejemplo:
```
?- simple(T).
```
`T = login;` (-> porque la realiza un programador).
`T = pruebasPerformance;` (-> porque la realiza un tester).
`T = tuning;` (-> porque tarda menos de cuarenta horas).

```
?- riesgo(T).
T = cacheDistribuida.
T = pruebasPerformance.
```

3. Usando el predicado forall definir el predicado puedoHacer/1 que indica si una tarea puede ser realizada. (Una tarea puede ser realizada cuando sus precedentes han sido realizadas). El predicado se debe poder usar de la siguiente forma:

```
?- puedoHacer(T).
T = cacheDistribuida;
```
(es la única tarea que puedo hacer en este momento, ya que login está realizada, para pruebasPerformance me falta cacheDistribuida y para tuning me falta pruebasPerformance).

Ejercicio 4 - Liga de Fútbol

Hay que desarrollar un programa en Prolog con el que podamos hacer consultas sobre los partidos desarrollados en una liga de fútbol.

Los hechos están dados por:

%fecha(NroFecha, partido(EquipoLocal,GolesLocal,EquipoVisitante,GolesVisitante)).

Por ejemplo

```
fecha(1,partido(mandiyu,0,losAndes,2)).
fecha(1,partido(yupanqui,2,claypole,1)).
fecha(1,partido(jjurquiza,1,cambaceres,4)).
fecha(2,partido(yupanqui,4,jjurquiza,2)).
fecha(2,partido(losAndes,3,claypole,1)).
fecha(2,partido(cambaceres,2,mandiyu,2)).
fecha(3,partido(jjurquiza,3,losAndes,1)).
fecha(3,partido(mandiyu,2,claypole,2)).
fecha(3,partido(cambaceres,3,yupanqui,2)).
```

Tengan en cuenta que no hay partidos que se hagan ida y vuelta en esta liga, o sea, dados dos equipos hay cargado a lo sumo un partido en el cual se enfrentan.
Suponer que la carga de los hechos es correcta, no hacer validaciones.

Desarrollar los siguientes predicados:

a) rival/3, que nos dice quién fue el rival de un equipo en una fecha. Algunas consultas de ejemplo con sus respuestas

```
?- rival(losAndes,2,claypole).
Yes

?- rival(claypole,2,losAndes).
Yes

?- rival(claypole,2,mandiyu).
No
```

```
?- rival(claypole,F,mandiyu).
F = 3;
No
```

b) jugaron/2 que relaciona dos equipos y nos responde si jugaron entre sí en alguna fecha de la liga.

c) faltaQueJueguen/2 que relaciona a dos equipos si aún no jugaron entre ellos, p.ej. Mandiyú y J.J.Urquiza, en cualquier orden.

d) gano/2 y **perdio/2** que relaciona dos equipos de acuerdo al resultado del partido jugado entre ambos. Un equipo ganó cuando la cantidad de goles que hizo es mayor que la de su rival.

e) empataron/2 que utilice gano/2 y perdio/2

f) Ver cómo va un equipo, para eso hacemos varios predicados:

- **vaBien/1** me dice si un equipo no perdió ningún partido,
- **vaMuyBien/1** me dice si ganó todos los partidos,
- **vaMuyMal/1** es verdadero si el equipo perdió todos sus partidos o bien si no ganó ninguno y perdió alguno por goleada (por más de tres goles)
- **vaMasOMenos/1** si no se cumple ninguno de los demás.

g) equipoImprevisible/1 se dice que un equipo es imprevisible si: le ganó a un equipo A, perdió con otro equipo B, pero B a su vez le ganó a A. Este predicado debe ser inversible, vale agregar hechos.

h) masCapoQue/2. Mi equipo es más capo que el tuyo si el mío le ganó al tuyo, y también les ganó a todos a los que le ganó el tuyo.