Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Inteligencia Artificial

Primer Cuatrimestre 2010

Trabajo Práctico 2

Integrantes			
		Pablo Carbajo	carbajo@gmail.com
		Diego Freijo	giga.freijo@gmail.com
		Ignacio Iacobacci	iiacobac@gmail.com
Instancia		**Corrector**	**Nota**
Entrega	02/06/2010		

1 Introducción, objetivo y alcance

El objetivo del presente trabajo es realizar prácticas de resolución lógica y representación del conocimiento.

1.1 Ejercicio 1

Representar todo el conocimiento descrito en las pistas del enigma lógico e introducirlas en un motor de inferencia a fin de resolver el mismo.

1.2 Ejercicio 2

Representar el conocimiento de las cláusulas que se presentan.

1.3 Ejercicio 3

Realizar prácticas de calculo situacional para el enunciado descrito.

2 Descripción y desarrollo de los experimentos

2.1 Ejercicio 3

"estaSobre(A,B,S)" es el unico fluent, indica si el bloque A esta sobre B en la situacion S. B puede ser otro bloque o el "piso".
La función resultado de aplicar una accion se llama "do" (en lugar de “resultado” como en la presentación dada en clase).
Las acciones son apilar(bloqueEnElPiso,bloqueQueQuedaAbajo)
desapilar(bloqueNoEnElPisoQueVaAQuedarEnElPiso).
Los axiomas se pueden ver en el código Porlog.

3 Metodologías y métricas utilizadas

3.1 Ejercicio 1
Motor de inferencia swi-prolog

3.2 Ejercicio 2
Motor de inferencia swi-prolog

3.3 Ejercicio 3
Motor de inferencia swi-prolog

4 Presentación y análisis de resultados

4.1 Ejercicio 1

La resolución se ejecuta casi instantáneamente (menos de 1 segundo).

Los resultados se presentan en 4 listas donde cada se encuentran listados los Policías, los Lugares, los Ladrones y los Países y la posición del índice define un arresto en particular.

Se llama a la función de esta forma:

?- **arrestos([],[],[],[], Policias, Lugares, Ladrones, Paises, 0).**

Y se obtiene como resultado:

Policias = [minari, kesner, frigerio, elkin, dodero],

Lugares = [cine, bar, barco, tren, avion],

Ladrones = [sombra, gato, dedos, huron, rata],

Paises = [austria, alemania, inglaterra, francia, espana].

Un arresto es el iesimo elemento de cada lista (por ej., dodero, avion, rata, españa)

4.2 Ejercicio 2

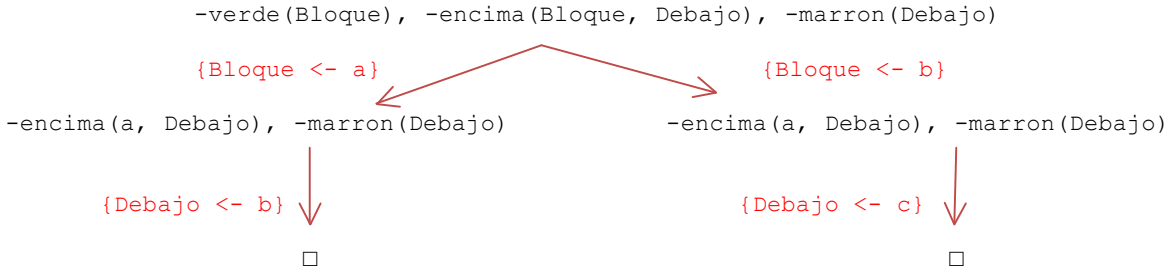
Para resolver este ejercicio primero se mostrara la solución en prolog, luego se mostrara como se soluciona el predicado manualmente y por ultimo otra solución usando negación por falla.

4.2.1 Resolución en prolog

```
bloque(a).
bloque(b).
bloque(c).
encima(a, b).
encima(b, c).
solucion(Bloque) :- verde(Bloque), encima(Bloque, Debajo),
    marron(Debajo).
```

4.2.2 Resolución manual

Se debe negar el objetivo y encontrar las sustituciones que muestren la insatisfacibilidad del mismo.



Por lo tanto, las soluciones son `Bloque = a` o `Bloque = b`.

4.2.3 Resolución por negación por falla

```
bloque(a).
bloque(b).
bloque(c).
encima(a, b).
encima(b, c).
verde(a).
marron(c).
no_marron(Bloque) :- bloque(Bloque), not(marron(Bloque)).
no_verde(Bloque) :- bloque(Bloque), not(verde(Bloque)).
solucion(Bloque) :- no_marron(Bloque), encima(Bloque, Debajo),
    no_verde(Debajo).
```

4.3 Ejercicio 3

Se incluyen como ejemplo en el código de Prolog las siguientes situaciones:

- situación inicial s0 - el bloque 2 sobre el 1, el resto en el piso.
  estaSobre(1, piso, s0).
  estaSobre(2, 1, s0).
  estaSobre(3, piso, s0).
  estaSobre(4, piso, s0).
  estaSobre(5, piso, s0).

- situacion s02: el bloque 1 sobre el 2, el resto en el piso.
  s02(do(apilar(1,2),do(desapilar(2),s0))).

Se incluyen también las siguientes consultas de ejemplo, que permiten ver cómo el motor infiere las propiedades del mundo a partir de los aximoas:

- s02(S), secPoss(S), poss(do(apilar(B1, B2), S)).
  retorna (instanciando B1 y B2) todos los bloques que se pueden apilar en la situacion s02

- s02(S), secPoss(do(desapilar(2),S)).
  retorna false porque la secuencia de acciones no es posible

- s02(S), secPoss(do(apilar(3,2),S)).
  retorna false porque la secuencia de acciones no es posible

- s02(S), estaSobre(A, B, S).
  retorna (instanciando Ay B) sobre quién esta cada bloque en s02.

5 Conclusiones

5.1 Ejercicio 1

En este ejercicio se complicó bastante el hecho de representar todo el conocimiento que proveía cada pista. Además fue costoso entender que las pistas sugeridas eran a veces disjuntas (por ejemplo que un arresto lo realizó dodero, y elkin). Fue necesario agregar en cada cláusula que representaba cada pista el concepto de utilizar la misma o no hacerlo. Por ejemplo:

La pista 4) decía: “Dodero atrapó al Rata en Espana”

Se tradujo como lo siguiente:

clausula4(Policia,Lugar,Ladron,Pais) :- Policia = dodero,Ladron = rata,Pais = espana; Policia \= dodero,Ladron \= rata,Pais \= espana.

O sea, se indicó que, existe un arresto donde el policia es dodero; el ladrón, rata; y el país, españa; o existe otro (u otros) arrestos en donde no sucede ninguna de las tres cosas a la vez. Esta misma lógica se aplicó a cada una de las pistas.

En conjunto, todas las pistas generaban 15 arrestos, a saber:

Policia = dodero, Lugar = avion, Ladrón = rata, País = espana ;	Policia = frigerio, Lugar = barco, Ladrón = dedos, País = inglaterra ;	Policia = kesner, Lugar = cine, Ladrón = gato, País = austria ;	Policia = minari, Lugar = cine, Ladrón = gato, País = francia ;
Policia = elkin, Lugar = tren, Ladrón = huron, País = francia ;	Policia = frigerio, Lugar = tren, Ladrón = dedos, País = francia ;	Policia = kesner, Lugar = cine, Ladrón = gato, País = francia ;	Policia = minari, Lugar = cine, Ladrón = sombra, País = austria ;
Policia = frigerio, Lugar = avion, Ladrón = huron, País = austria ;	Policia = frigerio, Lugar = tren, Ladrón = huron, País = francia ;	Policia = minari, Lugar = bar, Ladrón = gato, País = alemania ;	Policia = minari, Lugar = cine, Ladrón = sombra, País = francia ;
Policia = frigerio, Lugar = avion, Ladrón = huron, País = francia ;	Policia = kesner, Lugar = bar, Ladrón = gato, País = alemania ;	Policia = minari, Lugar = cine, Ladrón = gato, País = austria ;	

De estos es facil ver que el problema se basa en que los arrestos no son vistos como entidades disjuntas. Esto nos llevó a agregar una cláusula más que presentó los 5 arrestos como lo indicado anteriormente.

5.2 Ejercicio 2

Se pudo ver que la negación por falla puede ser una herramienta muy útil en ciertos escenarios. Pero en otros no. Y este fue uno de ellos, donde la solucion mas elegante y sencilla resulto ser la directa.

5.3 Ejercicio 3

Si bien el cálculo situacional es útil para inferir el estado del mundo luego de aplicar acciones, y es capaz de decir qué acciones son permitidas en cada momento, no sirve para inferir qué acciones deben ser aplicadas para ir de un estado a otro. Este último problema debe ser visto como un problema de búsqueda.

6 Anexos: dump de corridas, código y demás datos relevantes.

6.1 Ejercicio 1

```
policia(dodero).
policia(elkin).
policia(frigerio).
policia(kesner).
policia(minari).

lugar(avion).
lugar(bar).
lugar(barco).
lugar(cine).
lugar(tren).

ladron(gato).
ladron(dedos).
ladron(huron).
ladron(rata).
ladron(sombra).

pais(alemania).
pais(austria).
pais(españa).
pais(francia).
pais(inglaterra).

animal(gato).
animal(huron).
animal(rata).

transporte(avion).
transporte(barco).
transporte(tren).

esparcimiento(X):-lugar(X),not(transporte(X)).

%1_ Uno de los ladrones que tiene como alias el nombre de un animal fue atrapado en un bar
de Alemania.
clausula1(Policia,Lugar,Ladron,Pais) :-
  Pais = alemania,Lugar = bar,animal(Ladron);
  Pais \= alemania,Lugar \= bar.

%2_ Hurón no usó el barco que llegó a Inglaterra
clausula2(Policia,Lugar,Ladron,Pais) :-
  Lugar = barco, Pais = inglaterra, Ladron \= huron;
  Lugar \= barco, Pais \= inglaterra.

%3_ Quien fue detenido en el tren no estaba en España ni en Austria.
clausula3(Policia,Lugar,Ladron,Pais) :-
 Lugar = tren,Pais \= españa,Pais \= austria;
 Pais = austria,Lugar \= tren;
 Pais = españa,Lugar \=tren;
 Lugar \= tren,Pais \= españa,Pais \= austria.

%4_ Dodero atrapó al Rata en España.
clausula4(Policia,Lugar,Ladron,Pais) :-
  Policia = dodero,Ladron = rata,Pais = españa;
```

```
  Policia \= dodero,Ladron \= rata,Pais \= españa.

%5_ Ni el ladrón que fue atrapado por Elkin ni Dedos han estado en Alemania o Austria.
%   Ninguno de estos dos arrestos se realizó en un avión.
clausula5(Policia,Lugar,Ladron,Pais):-
  Policia = elkin,Ladron \= dedos,Lugar \= avion,Pais \= alemania,Pais \= austria;
  Ladron = dedos,Policia \= elkin,Lugar \= avion,Pais \= alemania,Pais \= austria;
  Policia \= elkin,Ladron \= dedos.

%6_ Sombra fue atrapado al salir de un cine pero no por Kesner o Frigerio.
clausula6(Policia,Lugar,Ladron,Pais):-
  Lugar = cine,Ladron = sombra,Policia \= kesner,Policia \= frigerio;
  Ladron \= sombra.

%7_ Dodero, Elkin y Frigerio apresaron sendos ladrones al bajar de un transporte.
%  En cambio, el Gato y Sombra fueron atrapados en lugares de esparcimiento.
clausula7(Policia,Lugar,Ladron,Pais) :-
  Policia = dodero,transporte(Lugar),Ladron \= gato,Ladron \= sombra;
  Policia = elkin,transporte(Lugar),Ladron \= gato,Ladron \= sombra;
  Policia = frigerio,transporte(Lugar),Ladron \= gato,Ladron \= sombra;
  Ladron = gato,esparcimiento(Lugar),Policia \= dodero,Policia \= elkin,Policia \=
frigerio;
  Ladron = sombra,esparcimiento(Lugar),Policia \= dodero,Policia \= elkin,Policia \=
frigerio.

arresto(Policia,Lugar,Ladron,Pais) :-
  policia(Policia),lugar(Lugar),ladron(Ladron),pais(Pais),
  clausula1(Policia,Lugar,Ladron,Pais),
  clausula2(Policia,Lugar,Ladron,Pais),
  clausula3(Policia,Lugar,Ladron,Pais),
  clausula4(Policia,Lugar,Ladron,Pais),
  clausula5(Policia,Lugar,Ladron,Pais),
  clausula6(Policia,Lugar,Ladron,Pais),
  clausula7(Policia,Lugar,Ladron,Pais).


% esto encuentra 5 arrestos que tengan todo distinto, implementando el axioma de mundo
cerrado.
% (y la idea de que no pueden repetirse cosas: si dodero atrapo a rata, nadie mas atrapo a
rata,
%  y dodero no atrapo a ningun otro ademas de rata)
arrestos( A, B, C, D, R1, R2, R3, R4, 5 ) :- R1 = A, R2 = B, R3 = C, R4 = D.
arrestos( PoL,LuL,LaL,PaL,R1,R2,R3,R4, C ) :-
  arresto(Po,Lu,La,Pa),
  not( member( Po, PoL ) ), not( member( Lu, LuL ) ), not( member( La, LaL ) ), not(
member( Pa, PaL ) ),
  D is C+1,
  arrestos( [Po|PoL],[Lu|LuL],[La|LaL],[Pa|PaL], R1,R2,R3,R4, D ),
  !.   % el cut este es para evitar las permutaciones


% para usar, copiar lo siguiente:
% arrestos( [],[],[],[], Policias, Lugares, Ladrones, Paises, 0 ).


% retorna algo asi:
% Policias = [minari, kesner, frigerio, elkin, dodero],
% Lugares = [cine, bar, barco, tren, avion],
% Ladrones = [sombra, gato, dedos, huron, rata],
% Paises = [austria, alemania, inglaterra, francia, españa].
%
% Lo cual es correcto, si se interpreta como que dodero atrapo a rata en un avion en
espana,
% elkin atrapo a huron en un tren de francia, y etc. hasta minari-cine-sombra-austria.
% ademas, funciona instantaneamente!
```

6.2 Ejercicio 3

```
% Los bloques son 1, 2, 3, 4 y 5
% El piso lo llamamos simplemente "piso"

% "estaSobre(A,B,S)" es el unico fluent, indica si el bloque A esta sobre B
% en la situacion S. B puede ser otro bloque o el "piso".
% la funcion resultado de aplicar una accion es "do".
% las acciones son apilar(bloqueEnElPiso,bloqueQueQuedaAbajo) y
% desapilar(bloqueNoEnElPisoQueVaAQuedarEnElPiso)
```

```
% situaciones para usar
% ------------------------------------------------------------------------
% s0 - inicial - el bloque 2 sobre el 1, el resto en el piso.
estaSobre( 1, piso, s0 ).
estaSobre( 2, 1, s0 ).
estaSobre( 3, piso, s0 ).
estaSobre( 4, piso, s0 ).
estaSobre( 5, piso, s0 ).

% situacion s02: el bloque 1 sobre el 2, el resto en el piso.
s02( do(apilar(1,2),do(desapilar(2),s0)) ).

% consultas ejemplo:
% s02(S), secPoss(S), poss( do( apilar(B1, B2), S ) ).
% retorna todos los bloques que se pueden apilar en la situacion s02

% s02(S), secPoss( do(desapilar(2),S) ).
% retorna false porque la secuencia de acciones no es posible

% s02(S), secPoss( do(apilar(3,2),S) ).
% retorna false porque la secuencia de acciones no es posible

% s02(S), estaSobre( A, B, S ).
% retorna sobre quien esta cada bloque en s02.


% ------------------------------------------------------------------------

% axioma de efecto para accion apilar
estaSobre( A, B, do( apilar( A, B ), S ) ).

% axioma frame accion apilar
estaSobre( A, B, do( apilar( B1, B2 ), S ) ) :-
  estaSobre( A, B, S ), A \= B1.

% axioma de efecto para accion desapilar
estaSobre( A, piso, do( desapilar( A ), S ) ).

% axioma frame accion desapilar
estaSobre( A, B, do( desapilar( B1 ), S ) ) :-
  estaSobre( A, B, S ),  A \= B1.


% precondicion accion apilar
sePuedeApilar( A, B, S ) :-
  estaSobre( A, piso, S ),
  not( estaSobre( X, A, S ) ), % A esta arriba de todo
  estaSobre( B, BB, S ), % instanciar B
  not( estaSobre( X, B, S ) ), % B esta arriba de todo
  B \= piso,
  A \= B.

% precondicion accion desapilar
sePuedeDesapilar( A, S ) :-
  estaSobre( A, B, S ), B \= piso, % A no esta en el piso
  not( estaSobre( X, A, S ) ), % A esta arriba de todo
  A \= piso.

poss( do( apilar( B1, B2 ), S ) ) :- sePuedeApilar( B1, B2, S ).
poss( do( desapilar( B ), S ) ) :- sePuedeDesapilar( B, S ).

secPoss( s0 ).
secPoss( do( A, S ) ) :- poss( do( A, S ) ), secPoss(S).
```