Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Tarea 1 – Gripper y la incertidumbre
IIC2613 - Inteligencia Artificial
Segundo Semestre, 2016
Entrega: 10 de Octubre, hasta las 23:59

1. Objetivo

El objetivo de esta tarea es que usted practique conceptos de programación en lógica, aplicado a la inteligencia artificial. En particular veremos cómo es posible representar incerteza en lenguajes lógicos como el cálculo de situaciones y veremos cómo es posible hacer algunos tipos de razonamiento, en particular, planificación, en presencia de incerteza.

2. El Mundo de Gripper

En el *mundo de gripper*, existe un robot llamado Robby, un número configurable de piezas, y un número configurable de bolitas. Robby tiene, además, un número configurable de manos robóticas. Robby puede desplazarse entre cualquier par de piezas mediante una única acción, y tomar una bolita en sus manos. Un problema de planificación en el mundo de Robby consiste en encontrar una secuencia de acciones que le indica a Robby cómo transportar bolitas hasta un lugar de destino.

En más detalle, las acciones que puede ejecutar el robot son las siguientes:

- $move(X, Y)$: Mueve al agente desde X hasta Y. Esta acción sólo es posible si el robot está en `X`.
- $pick(B, R, G)$: Recoge la bolita B en la pieza R con la mano G. Esta acción es posible si `G` está libre, y si tanto `B` como el robot están en `R`.
- $drop(B, R, G)$: Suelta la bolita B en la pieza R desde la mano G.

Por otro lado, usamos los siguientes predicados.

- $at\_robby(R)$ para decir que el robot está en la pieza R.
- $free(g)$: para decir que la mano g está libre (es decir, puede ser usada para tomar algo).
- $carry(B, G)$: B está siendo sostenida por la mano G del robot.

3. Parte 1: Conocimiento Completo

Complete el archivo base `tarea1-completo.pl` disponible en el sitio web. Específicamente agregue reglas para `poss` de cada acción y para los efectos de las acciones. Una vez terminado esto, usted debiera poder hacer las siguientes consultas.

```
?- legal(S), holds(at(b1,r4),S),holds(at(b2,r4),S).
S = do(drop(b2, r4, g2), do(drop(b1, r4, g1), do(move(r3, r4),
      do(pick(b2, r3, g2), do(move(r2, r3), do(pick(b1, r2, g1),
       do(move(r1, r2), s0)))))) .

?- legal(S), holds(at(b1,r1),S).
S = do(drop(b1, r1, g1), do(move(r2, r1),
     do(pick(b1, r2, g1), do(move(r1, r2), s0)))) .
```

4. Parte 2: Conocimiento Incompleto

En muchas aplicaciones de IA, los agentes no son capaces de observar el mundo en el que se desenvuelven, y, adicionalmente, tienen conocimiento incompleto. Para representar correctamente el conocimiento incompleto frecuentemente se usa la *semántica de mundos posibles*. Ella consiste en que el agente se considera a sí mismo en varios estados al mismo tiempo, donde un estado es un conjunto de proposiciones s. Más formalmente, entonces, el agente tiene ahora un estado de creencias b, que es un conjunto de estados, cada uno de los cuales es representado por un conjunto de proposiciones. Se dice que el agente *sabe* p *en* b si p pertenece a cada estado de b. Por otro lado, el agente *cree* p *en* b si p pertenece a algún estado de b.

El cálculo de situaciones se puede adaptar de manera sencilla para manejar conocimiento incompleto. Simplemente, en vez de suponer que estamos en una sola situación, supondremos que estamos en un conjunto de situaciones. Por ejemplo en el mundo de gripper, si es que no sabemos la ubicación de la bolita `b1`, y solo tenemos dos oficinas, `r1` y `r2`, entonces tendremos dos situaciones iniciales.

Existe, sin embargo, un detalle de representación, que se puede ilustrar con un ejemplo. Si no sabemos donde está `b1` ¿cómo representamos la acción de recoger esa bolita? ¿Cuál es la precondición? ¿Cuál es el efecto? Diremos que será posible recoger la bolita B en la pieza R usando la mano G si

- *Sabemos* que estamos en R.
- *Creemos* que la bolita B está en R.
- *Sabemos* que la mano G está libre.

Más generalmente, una precondición se evalúa sobre un estado de creencias. En nuestra implementación, entonces, deberemos usar un predicado `poss(A,Set)`, donde `A` es una acción y `Set` es un conjunto (lista) de situaciones. Note que, diferencia del caso de conocimiento completo, `poss` actúa sobre un conjunto de situaciones. Adicionalmente se debe implementar los predicados `knows(F,Set)` y `believes(F,Set)` que, respectivamente, expresan que `F` es conocido y creído en el conjunto `Set`. El predicado `holds`, por otro lado, queda funcionando igual que en el caso de certeza completa: `holds(F,S)` expresa que `F` se cumple en la situación `S`.

Para representar los efectos de las acciones en mundos inciertos debemos introducir efectos condicionales. Para ver esto, suponga que `s1` y `s2` son las situaciones iniciales del ejemplo anterior. Específicamente suponga que en `s1` la bolita `b1` está en `r1`, mientras que en `s2` la bolita `b1` está en `r2`. Suponga, además, que ejecuta la acción `pick(b1,r1)` en el estado de creencias dado por `[s1,s2]`. Las siguientes relaciones se deben cumplir.

- `holds(at(b1,r1),do(pick(b1,r1),s1))` debe ser *falso* porque la bolita ya no está en `r1` si es que inicialmente estábamos en `s1`.
- `holds(carry(b1),do(pick(b1,r1),s1))` debe ser *verdadero* porque la bolita ya es sostenida por el robot al estar inicialmente en `s1`.
- `holds(at(b1,r2),do(pick(b1,r1),s2))` debe ser *verdadero* porque la bolita sigue estando en `r2` si es que inicialmente estamos en `s2`.
- `holds(carry(b1),do(pick(b1,r1),s2))` debe ser *falso* porque la bolita no ha sido realmente recogida si inicialmente estamos en `s2`.

Para implementar correctamente este comportamiento se debe usar un efecto condicional. Específicamente, $pick(b, r, g)$ tiene el efecto de que el robot ahora tiene a b en su mano g **sólo si** b estaba en r. En caso contrario, la acción no tiene ningún efecto.

Esto se traduce en que en nuestra implementación ahora tendremos reglas del estilo:

```
conditional_positive_effect(pick(B,Room,Gripper),at(B,Room),carry(B,Gripper)).
conditional_negative_effect(pick(B,Room,Gripper),at(B,Room),free(Gripper)).
```

para expresar que, cuando `at(B,Room)` se cumple, entonces `carry(B,Gripper)` es un efecto positivo de `pick(B,Room,Gripper)`. Por otro lado, `G` deja de quedar libre si la misma condición se cumple.

Para esta parte, se espera que usted complete el archivo `tarea1-incompleto.pl`. Se espera que usted obtenga el siguiente comportamiento:

```
?- knows(F,[s1,s2]).
F = true ;
F = at_robby(r1) ;
F = at(b2, r3) ;
F = free(g1) ;
F = free(g2) ;
false.

?- believes(at(b1,r1),[s1,s2]).
true .

?- believes(at(b1,r2),[s1,s2]).
true .

?- legal(Set).
Set = [s1, s2] ;
Set = [do(move(r1, r2), s1), do(move(r1, r2), s2)] ;
Set = [do(move(r1, r3), s1), do(move(r1, r3), s2)] ;
Set = [do(pick(b1, r1, g1), s1), do(pick(b1, r1, g1), s2)] ;
Set = [do(pick(b1, r1, g2), s1), do(pick(b1, r1, g2), s2)] ;
Set = [do(move(r2, r1), do(move(r1, r2), s1)), do(move(r2, r1), do(move(r1, r2), s2))] .
```

Para encontrar planes, usted puede usar consultas como esta:

```
?- legal(Set),knows(at(b1,r3),Set).
Set = [do(drop(b1, r3, g2), do(drop(b1, r3, g1),
            do(move(r2, r3), do(pick(b1, r2, g2),
               do(move(r1, r2), do(pick(b1, r1, g1), s1))))),
       do(drop(b1, r3, g2), do(drop(b1, r3, g1),
            do(move(r2, r3), do(pick(b1, r2, g2),
               do(move(r1, r2), do(pick(b1, r1, g1), s2))))))]
```

(cuya respuesta podría tomar más tiempo del que usted considera razonable dado que, entre otros, `legal` hace una especie de BFS, de pobre rendimiento)

5. Bonus

Se dará un bonus de hasta 1 punto si usted consigue hacer que la búsqueda funcione significativamente más rápida que con la estrategia del predicado `legal`. Usar heurísticas es una posibilidad, pero hay otras opciones tal vez más sencillas de implementar.

6. Entrega

Envíe sus dos archivos por correo electrónico a `iic2613@ing.puc.cl`. El subject del correo debe ser "Tarea 1". Empaque sus archivos de manera amable.