PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
Facultad de Ingeniería
IIC2613 Inteligencia Artificial
2022-1

Tarea 1

Aspectos generales

Formato y plazo de entrega

El formato de entrega son archivos `.lp` en los directorios correspondientes a cada pregunta. El lugar de entrega es en el repositorio de la tarea, en la *branch* por defecto, hasta el **miércoles 20 de abril a las 23:59 hrs.**

Integridad Académica

Este curso se adhiere al Código de Honor establecido por la universidad, el cual tienes el deber de conocer como estudiante. Se espera que todo el trabajo hecho en esta tarea sea **totalmente individual** en cualquiera de sus aspectos. La idea es que te des el tiempo de aprender estos conceptos fundamentales, tanto para el curso, como para tu formación profesional. Las dudas se deben hacer exclusivamente al cuerpo docente a través de las *issues* en GitHub.

Por otra parte, sabemos que estás utilizando material hecho por otras personas, por lo que es importante reconocerlo de la forma apropiada. Todo lo que obtengas de internet debes citarlo de forma correcta (ya sea en APA, ICONTEC o IEEE). Cualquier falta a la ética y/o a la integridad académica será sancionada con la reprobación del curso y los antecedentes serán entregados a la Dirección de Pregrado.

Comentarios adicionales

El objetivo de esta tarea es que puedan desarrollar la capacidad de modelar problemas a partir del lenguaje natural y resolverlos utilizando ASP en Clingo. Por lo tanto, es fundamental que pongan énfasis en las explicaciones de sus respuestas, cuidando la redacción y ortografía, y manteniendo siempre el código ordenado y comentado. Aquellas respuestas que solo presenten resultados o código (sin contexto ni comentarios) no serán consideradas, mientras que tareas desordenadas pueden ser objeto de descuentos.

1. IA y un mundo mejor (Total: 1 pt.)

Lex Fridman[1], comunicador y profesor en el MIT, tiene un podcast en YouTube en el que ha entrevistado a notables investigadores en IA, como Ilya Sutskever, Judea Pearl o Ian Goodfellow; así como también a grandes referentes de la Ciencia de la Computación, como Donald Knuth, Guido van Rossum, Chris Lattner.

En Mayo de 2020, Fridman entrevistó a Kate Darling[2], quien investiga en el área de ética y robótica. En **esta entrevista**, ambos discuten sobre la aplicabilidad de la robótica para que las personas encuentren "la mejor parte de sí mismos" (minutos 17:10 - 20:30). Luego, entre los minutos 38:00 y 41:30, Darling describe cómo sería su "robot favorito".

Inspírate en esas dos secciones de este video,[3] para escribir una discusión de **media página** sobre cómo piensas que la IA podría ayudar a construir aplicaciones para sacar 'lo mejor de nosotros mismos', ya sea en la línea de la empatía, como propone Fridman, o cualquier otra que tú prefieras. Describe tu ejemplo, justifica por qué podría funcionar y ponte del lado de un crítico, tal como lo hace Darling en el video (minuto 19:50), para analizar si es que es posible que tu sistema no tenga el efecto esperado. Procura cuidar la redacción y ortografía, profundizando en tus ideas de la mejor manera posible, sintetizando lo más que puedas.

NOTA: Entrega tu respuesta en el directorio principal, en un archivo PDF llamado `respuestas.pdf`.

[1] `https://lexfridman.com/`

[2] `http://www.katedarling.org/`

[3] Puedes usar subtítulos y traducción automática si lo necesitas.

2. DCClue (Total: 2,5 pts.)

Con la presencialidad, muchos estudiantes se encuentran sumamente irritados por no poder levantarse a las 08:25 AM para ver su clase de Inteligencia Artificial a las 08:30 AM. En un arrebato de ira, un alumno escribe un graffiti expresando su descontento, por lo que el profesor Jorge Baier te solicita a ti, experto/a en programación, que desarrolles un programa en Clingo que, a partir de distintos antecedentes, pueda ayudar a esclarecer la situación, identificando al culpable y a los sospechosos para tener una profunda y sincera conversación con ellos.

Gracias a la particular firma del graffiti, sabemos que solamente una persona es la culpable material de este ilícito, mientras que todos los demás son presuntamente inocentes. Sin embargo, existe la posibilidad de que entre estas personas, algunas hayan cooperado con el culpable, por lo que todos los que encajen con ese perfil serán también considerados como sospechosos. Tu labor será generar un programa tal que en el *output* entregue una **versión valida** de cómo ocurrieron los hechos.

Para este problema, se tendrá un mapa que se trabajará como un grafo no dirigido, donde hay distintos lugares que pueden o no estar conectados por un camino. Tu programa debe ser capaz de recibir una instancia del mapa con la información de la siguiente manera:

- `lugar(L)`: Denota la existencia del lugar L.
- `camino(L1,L2)`: Denota la existencia de un camino directo que conecta $L1$ con $L2$.

Teniendo en cuenta esto, si tenemos el siguiente grafo de ejemplo (`map1.lp`):

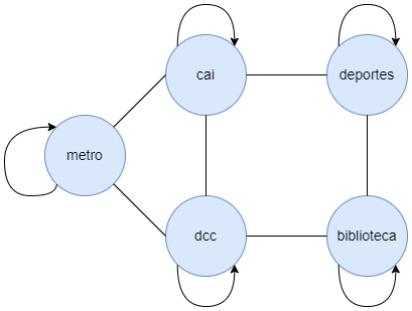

Figura 1: Ejemplo de mapa

Este mapa es descrito por un archivo de la siguiente forma:

```
% Lugares
lugar(cai).
lugar(deportes).
lugar(biblioteca).
lugar(metro).
lugar(dcc).

% Caminos
camino(metro, cai).
camino(metro, dcc).
camino(dcc, biblioteca).
camino(dcc, cai).
camino(cai, deportes).
camino(deportes, biblioteca).
```

NOTA: Considera que el grafo es no dirigido, por lo que se puede recorrer en ambas direcciones. Por otra parte, ten en cuenta que un lugar siempre estará conectado consigo mismo.

Adicionalmente, se te dará un segundo archivo con información adicional de la instancia, la cual puede contener los siguientes predicados:

- `tiempo(T)`: Indica la existencia del tiempo T como parte del período de análisis.
- `persona(P)`: Indica la existencia de la persona P.
- `personaEnLugar(P,L,T)`: Indica que la persona P estaba en el lugar L en el tiempo T.
- `afirmacionLugar(P1,P2,L,T)`: Indica que la persona $P1$ afirma que $P2$ estaba en el lugar L en el tiempo T.
- `descubrimiento(L,T)`: Indica que el crimen fue descubierto en el lugar L en el tiempo T.

El programa que desarrolles deberá generar una versión válida de cómo ocurrieron los hechos, es decir, encontrando al culpable, los sospechosos y asignando a cada persona un único lugar que no esté ocupado para cada instante de tiempo; todo esto de forma tal que toda la información sea consistente[4] entre sí. El *output* del programa debe mostrar **exclusivamente** los siguientes átomos:

- `culpable(P)`: Indica que la persona P es la culpable de haber hecho el grafitti.
- `sospechoso(P)`: Indica que la persona P es sospechosa.
- `personaEnLugar(P,L,T)`: Indica que la persona P estaba en el lugar L en el tiempo T.

El culpable

En este problema solamente **hay un único culpable**, el cual se puede descubrir teniendo en cuenta lo siguiente:

- El culpable es el único que puede mentir, así que si alguien se contradice o dice algo que sabemos que no es cierto o posible, entonces esa persona es la autora del graffiti (revisar sección de presuntos inocentes).
- El culpable debe haber estado en el lugar del crimen antes del tiempo de descubrimiento.

Presuntos inocentes

Para evitar ser descubiertos, todos **los presuntos inocentes dicen siempre la verdad**, por lo que en una versión válida de los hechos se debe cumplir lo siguiente:

- Como un presunto inocente no puede mentir, si dice que alguien estaba en un lugar en un tiempo determinado, entonces debe ser cierto y estar en ese lugar en dicho instante.
- Como un presunto inocente no puede mentir, entonces no puede afirmar que una persona estuvo en dos lugares en momentos tales que no haya suficiente tiempo como para que sea posible desplazarse. Por ejemplo, en la figura 1, no se puede estar en $T = 1$ en el metro y en $T = 2$ en la biblioteca, ya que hay que desplazarse 2 veces.

[4] Pueden existir varios modelos, lo importante es que se trate de una opción válida.

- Como un presunto inocente no puede mentir, no puede decir que hay dos personas el mismo lugar al mismo tiempo, ya que consideraremos que solo puede haber una persona en cada lugar por cada instante de tiempo.
- Como un presunto inocente no puede mentir, no puede decir que alguien estaba en un lugar si sabemos de antemano que estaba en otro.
- Como un presunto inocente no puede mentir, entonces no se pueden contradecir entre ellos. Por lo tanto, no pueden dar versiones diferentes sobre la ubicación de una persona en un instante.
- Como un presunto inocente no puede mentir, no pueden decir que alguien está en dos lugares al mismo tiempo.

Sospechosos

Los sospechosos son presuntos inocentes, pero que además han estado **al menos 3 veces** en el lugar del crimen antes del tiempo de descubrimiento.

Consideraciones adicionales

Para presentar una versión de los hechos lo más simple posible, tu programa deberá entregar un *output* con una versión válida que contenga la **menor cantidad posible de sospechosos**[5].

[5]Pueden haber varios modelos en el óptimo con menor cantidad de sospechosos

3. DCCajas (Total: 2,5 pts.)

En años anteriores, expertos del DCC desarrollaron un modelo que permitiera a los robots moverse a través de una bodega para llevar cajas a las estanterías. En dicho modelo, cada robot podía llevar una única caja asociada a él, la cual tenía que ser trasladada a una estantería determinada. Además, cada robot partía desde el inicio del programa llevando su caja y en ningún momento podía desligarse de ella, tal como se muestra en la siguiente imagen:

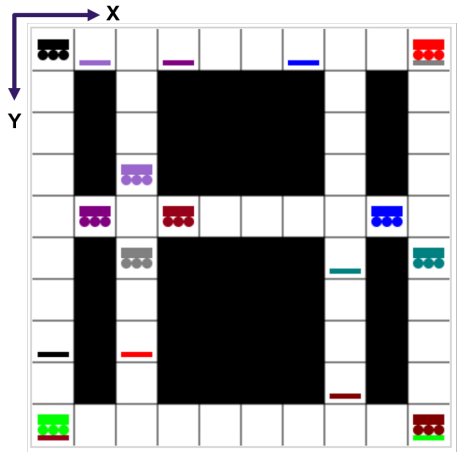

Figura 2: Sistema de robots antiguo

En este programa, el mapa se modela como una grilla, donde cada celda tiene una posición (X, Y) que aumenta según lo mostrado en la figura 1 (ej: el robot negro estaría en $(0, 0)$ y el rojo en $(9, 0)$). El tiempo se mide de forma discreta y, en cada instante de tiempo, los robots pueden realizar una acción, ya sea moverse en una dirección válida o quedarse en su lugar. El objetivo es planificar las acciones de los robots, de modo que al final del periodo de ejecución, todos los robots estén en su objetivo.

Actividad 1: El problema actual (2 pts.)

Con el paso del tiempo y buscando abaratar costos, la empresa que utilizaba esta tecnología ha estandarizado las cajas, haciéndolas todas idénticas. Lo mismo ocurre con los robots y las estanterías, por lo que ahora cualquier robot puede llevar cualquier caja hasta cualquier estantería. Además, ahora la posición de las cajas también es relevante, ya que al inicio parten en el suelo, hasta que algún robot las tome y traslade. Nuevamente, el objetivo es planificar las acciones de los robots en cada instante, para que al final lleven todas las cajas hasta una de las estanterías

La gerente te solicita a ti que actualices el programa para que sea compatible con el esquema de trabajo actual de la empresa y te da la siguiente lista de consideraciones:

- Los robots tienen 3 acciones básicas que pueden hacer en un instante de tiempo: moverse, esperar y tomar una caja.
- Los robots solo pueden moverse 1 celda a la vez por tiempo y solo pueden hacer movimientos verticales y horizontales, no diagonales.
- Los robots y las cajas no deben salir en ningún momento del mapa.
- No puede haber más de 1 robot o caja en una celda en el mismo instante de tiempo.

- Los robots no pueden traspasar obstáculos, cajas, ni otros robots.
- Los robots no pueden pasar por encima de otros robots, es decir, sus posiciones no pueden permutar entre ellas de un tiempo a otro. Ejemplo: Si en el tiempo T, el robot R_1 está en (X, Y) y el robot R_2 está en $(X+1, Y)$, no es posible que en $T+1$ el robot R_1 esté en $(X+1, Y)$ y R_2 esté en (X, Y).
- Los robots solo pueden tomar cajas que estén **en el suelo**, en celdas adyacentes no diagonales.
- Los robots solo pueden llevar 1 caja a la vez.

Por otra parte, tu programa debe recibir una instancia del problema, el cual tendrá los siguientes átomos:

- `rangeX(X)`: Indica que el valor X es parte de la dimensión horizontal del mapa.
- `rangeY(Y)`: Indica que el valor Y es parte de la dimensión vertical del mapa.
- `robot(R)`: Indica la existencia del robot R.
- `box(B)`: Indica la existencia de la caja B.
- `robotOn(R,X,Y,T)`: Indica que el robot R está en la posición (X, Y) en el tiempo T. En las instancias se pasará la posición inicial, es decir, con $T = 0$.
- `boxOn(B,X,Y,Z,T)`: Indica que la caja B está en la posición (X, Y, Z), donde Z indica si está en el suelo $(Z = 0)$ o en el aire, transportada por un robot $(Z = 1)$, en el tiempo T. En las instancias se pasará la posición inicial, es decir, con $T = Z = 0$.
- `obstacle(X,Y)`: Indica que en la posición (X, Y) hay un obstáculo.
- `goal(X,Y)`: Indica que en la posición (X, Y) hay una estantería.

Actividad 2: Robots procrastinadores (0.5 pts)

En el modelo actual, los robots cumplen a cabalidad su objetivo de llevar las cajas a la meta en un tiempo determinado. Sin embargo, a veces los robots que ya han llegado a su objetivo presentan un comportamiento ineficiente mientras otros siguen en camino a una estantería. Esto ocurre ya que no existe ninguna regla que les impida, por ejemplo, dar vueltas en circulo mientras sus compañeros terminan su trabajo, tal como se ve en el siguiente imagen:

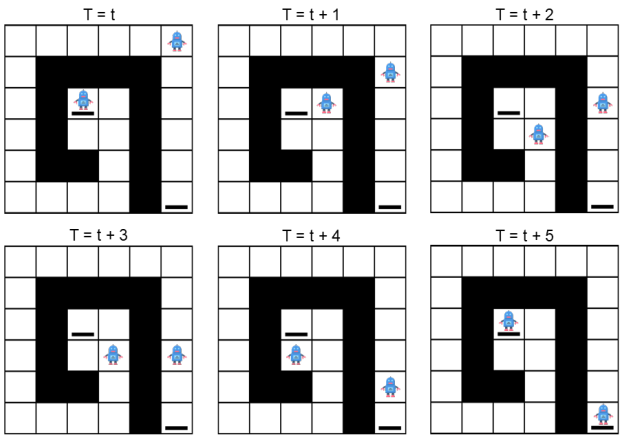

Figura 3: Ejemplo de procrastinación robótica

Tu labor será añadir lo necesario[6] a tu programa para que los robots no tengan estas conductas y que, una vez lleguen a su objetivo (y siempre que sea posible), se mantengan en su lugar[7]. Adicionalmente, deberás explicar en el PDF tu planteamiento, dando detalles de la lógica que utilizaste para dar solución al problema.

NOTA 1: Entrega la parte escrita de tu respuesta en el directorio principal, en el mismo archivo `respuestas.pdf`.

Nota 2: Se recomienda plantearlo como la maximización de la cantidad de tiempos en un estado particular.

Nota 3: Puedes añadir las reglas directamente en el archivo de la parte anterior, indicando claramente con comentarios dónde se ubica la implementación o puedes crear el archivo `optimized_model.lp`.

Bonus: Soltar cajas (0.5 pts)

Modifica el código anterior para añadir la acción de soltar una caja a los robots. Esta también se podría realizar en un instante de tiempo y debes considerar lo siguiente:

- Solo puede soltar una caja un robot que esté llevando una encima.
- Solo se puede soltar una caja en una celda adyacente que esté libre, sin considerar las diagonales.
- Una vez una caja es soltada, pasa a estar nuevamente en el suelo ($Z = 0$).

Requerimiento: Visualizador

Para poder revisar este ejercicio de forma agnóstica a cada planteamiento particular, se utilizará el visualizador del archivo `robot.html`. Para ello, deberás desarrollar un pequeño programa en Python que escriba los resultados con cierto formato en un archivo de texto (.txt), donde las primeras 2 líneas deben ser de la siguiente manera:

```
<dimensionX>,<dimensionY>
<tiempoMaximo>
```

Luego, debería continuar una serie de líneas donde se especifique la posición de cada estantería y de los obstáculos, siguiendo este formato (anteponer una 'E' para estanterías y 'O' para obstáculos):

```
E,<posX>,<posY>
O,<posX>,<posY>
```

El resto de las líneas deben seguir un formato como el de los predicados de posición de los robots y/o cajas, indicando la posición de cada robot/caja en cada instante. Para los robots debería ser algo así (anteponer una 'R'):

```
R,<id>,<posX>,<posY>,<tiempo>
```

mientras que para las cajas debería ser así (anteponer una 'C' y considerar el Z):

```
C,<id>,<posX>,<posY>,<posZ>,<tiempo>
```

Si todo esto se realiza correctamente, bastaría con cargar el archivo de texto resultante en el HTML.

[6] Puedes utilizar como código base el visto en ayudantías/clases

[7] Esto no debería cambiar el valor mínimo de bound, pero mejoraría la solución en un contexto real.