SUSTENTACIÓN PROYECTO N.1

INTRODUCCIÓN INTELIGENCIA ARTIFICIAL.

María Paula Cardona, Juan Manuel Sánchez, Matías González.

14 marzo 2024.

Pontificia Universidad Javeriana.

Departamento De Ingeniería De Sistemas.

Introducción a la Inteligencia Artificial.

CONTEXTO DEL PROYECTO:

Para este proyecto es necesario realizar un algoritmo en Prolog, que sea capaz de trasladar cajas entre habitaciones. En específico, tiene que pasar 3 cajas, de colores azul, rojo y verde, entre las habitaciones H1, H2, y H3.

El robot está en H1, con dos cajas, una azul y una roja y en H3 se encuentra una verde. El objetivo es llevar la caja azul a H3, la caja roja a H2 y la verde a H1, teniendo en cuenta que la posición final del robot es irrelevante.

El comportamiento del robot se podría simular mediante el siguiente grafo:

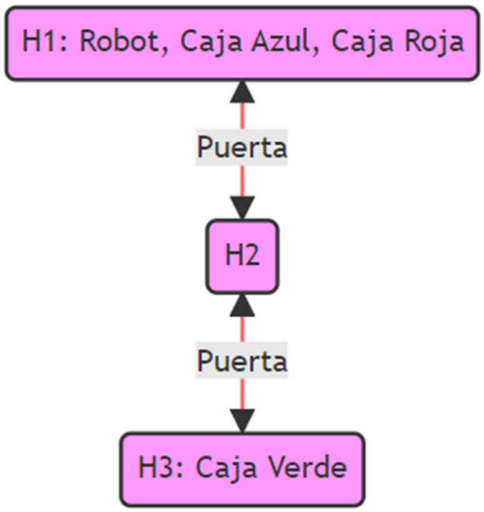

Figure 1: Diagrama que muestra las habitaciones con los estados iniciales de los elementos y los caminos que los conectan.

Donde:

- Cada nodo corresponde a una tupla en donde se encuentran las habitaciones y los objetos en ellas (ya sean cajas o el robot)
- Cada arista representa la conexión entre habitaciones.

DESCRIPCIÓN DE LA IMPLEMENTACIÓN

Para poder resolver el problema, fue necesario implementar un algoritmo en Prolog, contenido en el archivo "Sustentacion_N.pl", el cual se estructura en torno a la descripción de entidades, que incluyen las habitaciones y las conexiones entre estas, así como la especificación de los objetos, en este caso, las cajas de colores. Para gestionar dinámicamente la ubicación de las entidades durante la ejecución, se utilizan hechos dinámicos que permiten rastrear los cambios de posición del robot y las cajas a lo largo del programa. Además, se definen acciones clave como trasladar, tomar y dejar, que facultan al robot para manipular las cajas y moverlas entre las diferentes habitaciones. La resolución del problema se logra mediante un predicado que ejecuta una secuencia de acciones previamente definida, lo que permite al robot alcanzar el estado objetivo. Finalmente, para validar el estado final del problema resuelto, se emplea otro predicado que verifica la ubicación final de las cajas.

SOLUCIÓN:

Link Git: https://github.com/M4TI4SGV/PROJECT-IA

HECHOS:

Estos hechos incluyen la definición de:

- los espacios por utilizar, que en este caso corresponden a las habitaciones,
- las conexiones entre estas habitaciones
- los objetos por utilizar, representados por las cajas de colores
- las posiciones iniciales tanto del robot como de las tres cajas.

Conexiones entre espacios:

espacio(habitacion1).

espacio(habitacion2).

espacio(habitacion3).

Conexiones entre habitaciones:

entre_espacios(habitacion1, habitacion2).

entre_espacios(habitacion2, habitacion1).

entre_espacios(habitacion2, habitacion3).

entre_espacios(habitacion3, habitacion2).

Objetos disponibles:

objeto(caja_azul).

objeto(caja_roja).

objeto(caja_verde).

Posición inicial de objetos y robot:

posicion(robot, habitacion1).

posicion(caja_azul, habitacion1).

posicion(caja_roja, habitacion1).

posicion(caja_verde, habitacion3).

REGLAS:

Estas corresponden a las acciones disponibles incluyen trasladar objetos de una habitación a otra y permitir al robot tomar y dejar objetos. También, hay una regla que verifica el estado final de las cajas para asegurar que hayan sido colocadas en las posiciones deseadas.

Trasladar Un Objeto O Robot:

trasladar(Entidad, Desde, Hacia) :- ...

Esta regla permite mover una entidad (robot o caja) de una ubicación a otra, siempre que exista una conexión directa.

Tomar una caja:

tomar(Robot, Caja) :- ...

Define cómo el robot puede tomar una caja si ambos están en la misma ubicación.

Dejar una caja:

dejar(Robot, Caja) :- ...

Establece cómo el robot puede dejar una caja en su ubicación actual.

Resolver el problema:

resolver_problema :- ...

Es la regla principal que utiliza las otras reglas para mover las cajas a sus ubicaciones objetivo según los requisitos del problema.

HEURÍSTICA:

Dada la naturaleza simple del problema y las reglas claras de movimiento, la heurística se centraría en evaluar la mejor acción siguiente basándose en el estado actual de las cajas y el robot. Aunque el escenario parece más apto para un plan fijo que para una búsqueda heurística tradicional debido a sus limitaciones, es posible implementar una heurística que considere criterios como la distancia mínima a la habitación objetivo de cada caja y la minimización del número de movimientos del robot vacío. Esto permitiría hacer el código más adaptable y preparado para futuras expansiones del problema.

RESOLVER PROBLEMA:

La regla resolver_problema describe una secuencia de acciones para resolver este problema particular. Luego, se inician una serie de acciones secuenciales:

Movimiento de la Caja Roja a la Habitación 2:

El robot toma la caja roja.

La caja roja se traslada desde la habitación 1 a la habitación 2.

El robot deja la caja roja en la habitación 2.

Movimiento de la Caja Verde a la Habitación 1:

El robot se traslada desde la habitación 2 a la habitación 3.

El robot toma la caja verde.

La caja verde se traslada desde la habitación 3 a la habitación 2.

La caja verde se traslada desde la habitación 2 a la habitación 1.

El robot deja la caja verde en la habitación 1.

Movimiento de la Caja Azul a la Habitación 3:

El robot toma la caja azul.

La caja azul se traslada desde la habitación 1 a la habitación 2.

La caja azul se traslada desde la habitación 2 a la habitación 3.

El robot deja la caja azul en la habitación 3.

Después de completar estas acciones, se llama a la regla verificar_estado_final para asegurarse de que las cajas hayan sido colocadas en las posiciones deseadas.

PLAN DE PRUEBAS 1:

Se realiza una prueba de movimientos de robot junto a su caja en diferentes espacios mediante las siguientes consultas:

Conexiones espacios:

entre_espacios(habitacion1, habitacion2).

entre_espacios(habitacion2, habitacion1).

entre_espacios(habitacion2, habitacion3).

entre_espacios(habitacion3, habitacion2).

```
?- entre_espacios(habitacion1, habitacion2).
true.

?- entre_espacios(habitacion2, habitacion1).
true .

?- entre_espacios(habitacion2, habitacion3).
true.

?- entre_espacios(habitacion3, habitacion2).
true.

?- entre_espacios(habitacion1, habitacion3).
false.
```

Figure 2: Prueba mediante consultas.

Ubicación

posicion(robot, UbicacionRobot).

posicion(caja_azul, UbicacionCajaAzul).

posicion(caja_roja, UbicacionCajaRoja).

posicion(caja_verde, UbicacionCajaverde).

```
?- posicion(robot, UbicacionRobot).
UbicacionRobot = habitacion1.

?- posicion(caja_azul, UbicacionCajaAzul).
UbicacionCajaAzul = habitacion1.

?- posicion(caja_roja, UbicacionCajaRoja).
UbicacionCajaRoja = habitacion1.

?- posicion(caja_verde, UbicacionCajaverde).
UbicacionCajaverde = habitacion3.
```

Figura 3: Prueba mediante consultas.

Acciones básicas

Se realiza una prueba de acción básica de llevar la roja a la habitación 3 y dejarla en dicha habitación y desplazar el robot a la habitación 2

tomar(robot, caja_azul).

trasladar(robot, habitacion1, habitacion2).

dejar(robot, caja_azul).

```
?- posicion(robot, UbicacionRobot).
UbicacionRobot = habitacion1.

?- tomar(robot, caja_roja).
robot ha tomado la caja_roja.
true.

?- trasladar(robot, habitacion1, habitacion2).
robot movido de habitacion1 a habitacion2.
true.

?- trasladar(robot, habitacion2, habitacion3).
robot movido de habitacion2 a habitacion3.
true.

?- dejar(robot, caja_roja).
robot ha dejado la caja_roja en habitacion3.
true.

?- trasladar(robot, habitacion3, habitacion2).
robot movido de habitacion3 a habitacion2.
true.
```

Figure 4: Prueba acciones básicas

Verificamos las acciones realizadas:

```
?- posicion(robot, UbicacionRobot).
UbicacionRobot = habitacion2.

?- posicion(caja_verde, UbicacionCajaverde).
UbicacionCajaverde = habitacion3.
```

Figure 5: Verificación de la posición del robot y de la caja.

Lo cual es un resultado **EXITOSO**

RESULTADO DE PRUEBA: EXITOSO

PLAN DE PRUEBAS 2:

Se hace uso de la función resolver_problema:

La función resolver_problema inicia mostrando "Inicio de la solución" en la salida estándar. Luego, realiza una secuencia de acciones para mover las cajas a sus ubicaciones finales. Primero, el robot toma la caja roja de habitacion1 y la traslada a habitacion2. Posteriormente, traslada el robot de habitacion2 a habitacion3 para tomar la caja verde y llevarla de vuelta a habitacion2, y luego a habitacion1. Finalmente, el robot toma la caja azul de habitacion1 y la traslada a habitacion3. Después de cada acción, se verifica el estado final de las cajas.

```
?- resolver_problema.
Inicio de la soluciÃ³n.
robot ha tomado la caja_roja.
robot movido de habitacion1 a habitacion2.
robot ha dejado la caja_roja en habitacion2.
robot movido de habitacion2 a habitacion3.
robot ha tomado la caja_verde.
robot movido de habitacion3 a habitacion2.
robot movido de habitacion2 a habitacion1.
robot ha dejado la caja_verde en habitacion1.
robot ha tomado la caja_azul.
robot movido de habitacion1 a habitacion2.
robot movido de habitacion2 a habitacion3.
robot ha dejado la caja_azul en habitacion3.
Estado final: caja_azul en habitacion3, caja_roja en habitacion2, caja_verde en habitacion1.
true .
```

Figura 6: Prueba resolver_problema.

Se verifica posición de la caja verde:

```
?- posicion(caja_verde, UbicacionCajaverde).
UbicacionCajaverde = habitacion1.
```

Figura 7: Prueba ubicación final caja verde

Se verifica posición de la caja azul:

```
?- posicion(caja_azul, UbicacionCajaAzul).
UbicacionCajaAzul = habitacion3.
```

Figure 8: Prueba ubicación final caja azul.

Se verifica posición de la caja roja:

```
?- posicion(caja_roja, UbicacionCajaRoja).
UbicacionCajaRoja = habitacion2.
```

Figura 9: Prueba ubicación final caja roja

Resultado: **EXITOSO**

RESULTADO DE PRUEBA: EXITOSO