PRÁCTICA 3 TSI - HTNP

Julio A. Fresneda – 49215154F – juliofresnedag@correo.ugr.es

Ejercicio 1.

En el problema original, en el task transport-person faltaba el caso en el que el avión y la persona estén en ciudades distintas. Para solventar esto se añade el Caso3:

```
(:method Case3 ; si el avión y la persona están en distintas ciudades, y la persona no está en el destino
 :precondition (and (at ?p - person ?c1 - city)
                    (at ?a - aircraft ?c2 - city)
                    (different ?c1 - city ?c2 - city))

 :tasks (
        (mover-avion ?a ?c2 ?c1 )
        (board ?p ?a ?c1)
        (mover-avion ?a ?c1 ?c)
        (debark ?p ?a ?c)
 )

 )
)
```

Como vemos, en las precondiciones se exige que la persona y el avión estén en diferentes ciudades, y como efecto movemos el avión a la ciudad donde esté la persona, se monta a bordo, y se lleva a destino.

Ejercicio 2.

En este ejercicio, ahora contamos con que hay combustible (fuel), por lo tanto, el avión deberá repostar cuando no tenga suficiente combustible. Para llevar a cabo esta mejora, se ha modificado lo siguiente del dominio:

Se ha modificado la derivada sobre si hay fuel, ahora calcula si para viajar desde c1 hasta c2 queda fuel suficiente.

```
(:derived

  (hay-fuel ?a - aircraft ?c1 - city ?c2 - city)
  (>= (fuel ?a) (* (distance ?c1 ?c2) (slow-burn ?a) ) )


)
```

En mover-avion se han creado dos casos: Si tenemos fuel suficiente, o si no lo tenemos, usando como precondición la derivada anterior. Si hay fuel suficiente, se hace el viaje de forma normal. Si no lo hay, primero se reposta.

```
(:task mover-avion
 :parameters (?a - aircraft ?c1 - city ?c2 -city)
 (:method fuel-suficiente
  :precondition (hay-fuel ?a ?c1 ?c2)
  :tasks (
          (fly ?a ?c1 ?c2)
         )
  )

  (:method fuel-insuficiente
  :precondition (not(hay-fuel ?a ?c1 ?c2))
  :tasks (
          (refuel ?a ?c1)
          (fly ?a ?c1 ?c2)
         )
  )
)
```

Ejercicio 3.

En este problema vamos a considerar dos tipos de vuelo: Lento (fly) y rápido (zoom). El vuelo rápido tarda menos pero consume más combustible. Nuestro ejercicio consiste en modificar el dominio para que se hagan los viajes lo más rápido posible. Se ha establecido un límite de fuel.

Los cambios que se han hecho son los siguientes.

Ahora hay dos derivadas sobre calcular si habrá fuel suficiente para el viaje entre c1 y c2: Una para viaje lento, y otra para viaje rápido.

```
(:derived
  (hay-fuel ?a - aircraft ?c1 - city ?c2 - city)
  (>= (fuel ?a) (* (distance ?c1 ?c2) (slow-burn ?a) ) )
)

(:derived
  (hay-fuel-zoom ?a - aircraft ?c1 - city ?c2 - city)
  (>= (fuel ?a) (* (distance ?c1 ?c2) (fast-burn ?a) ) )
)
```

El task de mover-avion se ha modificado también. Ahora no hay acciones para fuel suficiente o insuficiente, las hay para fuel suficiente e insuficiente para zoom o fly, de forma que la primera acción es fuel-suficiente-zoom y la segunda fuel-insuficiente-zoom. Así nos aseguramos de que siempre que se pueda, se haga un viaje rápido, ya que buscamos optimizar en tiempo de vuelo.

También consideramos el límite de fuel, por si no nos podemos permitir un vuelo zoom pero sí uno lento.

```
(:task mover-avion
 :parameters (?a - aircraft ?c1 - city ?c2 -city)

 (:method fuel-suficiente-zoom
 :precondition ( and (hay-fuel-zoom ?a ?c1 ?c2)  (<= (+(*(distance ?c1 ?c2)(fast-burn ?a))(total-fuel-used))(fuel-limit)))

 :tasks (
        (zoom ?a ?c1 ?c2)
       )
 )

 (:method fuel-insuficiente-zoom
 :precondition (and (not(hay-fuel-zoom ?a ?c1 ?c2)) (<= (+(*(distance ?c1 ?c2)(fast-burn ?a))(total-fuel-used))(fuel-limit)))
 :tasks (
        (refuel ?a ?c1)
        (zoom ?a ?c1 ?c2)
       )
 )

 (:method fuel-suficiente-fly
 :precondition ( and(hay-fuel ?a ?c1 ?c2)  (<= (+(*(distance ?c1 ?c2)(slow-burn ?a))(total-fuel-used))(fuel-limit))  )

 :tasks (
        (fly ?a ?c1 ?c2)
       )
 )

 (:method fuel-insuficiente-fly
 :precondition ( and(not(hay-fuel ?a ?c1 ?c2)) (<= (+(*(distance ?c1 ?c2)(slow-burn ?a))(total-fuel-used))(fuel-limit))  )
 :tasks (
        (refuel ?a ?c1)
        (fly ?a ?c1 ?c2)
       )
 )
```

Vemos que hace viaje zoom siempre que puede, excepto en los dos últimos viajes, que hace vuelo lento porque si hiciese vuelo zoom sobrepasaría el límite.

Ejercicio 4.

En este ejercicio debemos modificar también las primitivas. Debemos representar que cada avión puede llevar a varios pasajeros a la vez (con un máximo). Los cambios hechos son los siguientes:

Se ha añadido el predicado (destino ?p – person ?c – city), y las funciones (pasajeros-a-bordo ?a - aircraft), (aforo-maximo ?a - aircraft), (tiempo-vuelo ?a - aircraft) y (tiempo-maximo ?a - aircraft).

Se ha modificado el task transport-person:

```
(:task transport-person
 :parameters (?p - person ?c - city)

 (:method Case1 ; si la persona está en la ciudad no se hace nada
  :precondition (at ?p ?c)
  :tasks ()
 )

 (:method Case2 ; si la persona está en el avion y el avión en la ciudad de destino
  :precondition (and(in ?p - person ?a - aircraft)(destino ?p ?c)(at ?a ?c))
  :tasks(
    (debark-todos ?a ?c)
  )
 )

 (:method Case3 ;si no está en la ciudad destino, pero avion y persona est@n en la misma ciudad
 :precondition (and (at ?p - person ?c1 - city)
                    (at ?a - aircraft ?c1 - city)
                    (not (= ?c1 ?c)))
 :tasks (
        (board-todos ?a ?c1)
        (transport-person ?p ?c))
 )


 (:method Case4 ; si el avión y la persona están en distintas ciudades, y la persona no está en el destino
 :precondition (and (at ?p - person ?c1 - city)
                    (at ?a - aircraft ?c2 - city)
                    (not(= ?c1 ?c2))
                    (not(= ?c1 ?c )))
 :tasks (
         (mover-avion ?a ?c2 ?c1 )
         (transport-person ?p ?c)
 )
 )

 (:method Case5 ; si no hay ninguna persona para embarcar (ya están todas embarcadas)
   :precondition (and( at ?a - aircraft ?c1 - city)(not(= ?c1 ?c)))
   :tasks(
         (mover-avion ?a ?c1 ?c)
         (debark-todos ?a ?c)
   )
 )
   |

 )
```

El primer caso se activa si la persona ya está en la ciudad, por lo que no habría que transportarla.

El segundo caso se activa si la persona está montada en el avión y el avión está en la ciudad de destino, donde se llamaría a desembarcar a todas las personas cuyo destino sea esa ciudad (no sólo desembarca la persona que estamos transportando).

El tercer caso se activa si el avión y la persona están en la misma ciudad, pero no es la de destino. En este caso sube al avión esa persona (y todas las que estén en la ciudad) y volvemos a llamar a transport-person recursivamente.

El cuarto caso se activaría si la persona no está en la ciudad donde está el avión, por lo que el avión se movería a la ciudad donde sí esté esa persona, y se volvería a llamar a transport-person recursivamente.

El quinto caso se llama si la persona ya está a bordo, por lo que se la lleva a destino y desembarca a todas las personas cuyo destino sea esa ciudad (no solo a la persona que hemos pasado como argumento).

Los tasks board-todos y debark-todos lo que hacen es llamar recursivamente a board y debark hasta que no queden personas por subir o bajar del avión (en el caso de bajar, siempre y cuando el destino de esa persona sea la ciudad donde se desembarca).

```
(:task board-todos
 :parameters (?a - aircraft ?c - city )

 (:method embarcar
  :precondition (and( at ?p - person ?c  )(not(destino ?p ?c)))
  :tasks (
    (board ?p ?a ?c)
    (board-todos ?a ?c)
  )
  )

  (:method fin
   :precondition()

   :tasks())
)
```

```
(:task debark-todos
 :parameters (?a - aircraft ?c - city )

 (:method desembarcar
  :precondition (and( in ?p - person ?a  )(destino ?p ?c))
  :tasks (
    (debark ?p ?a ?c)
    (debark-todos ?a ?c)
  )
  )

  (:method fin
   :precondition()

   :tasks())
)
```

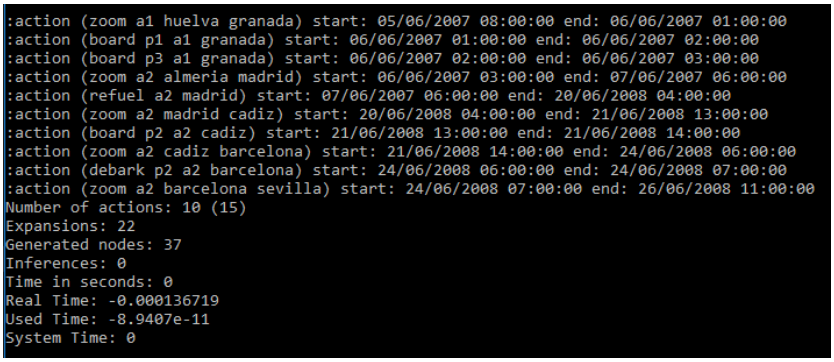
Para probar este dominio he construido tres problemas:

En el primer problema, p1 está en Granada y su destino está en Madrid, p2 está en Cádiz y su destino está en Barcelona, y p3 está en Granada y su destino está en Sevilla. El avión está en huelva. La ejecución es la siguiente:

```
:action (zoom a1 huelva granada) start: 05/06/2007 08:00:00 end: 06/06/2007 01:00:00
:action (board p1 a1 granada) start: 06/06/2007 01:00:00 end: 06/06/2007 02:00:00
:action (board p3 a1 granada) start: 06/06/2007 02:00:00 end: 06/06/2007 03:00:00
:action (refuel a1 granada) start: 06/06/2007 03:00:00 end: 05/07/2008 15:00:00
:action (zoom a1 granada madrid) start: 05/07/2008 15:00:00 end: 06/07/2008 12:00:00
:action (debark p1 a1 madrid) start: 06/07/2008 12:00:00 end: 06/07/2008 13:00:00
:action (zoom a1 madrid cadiz) start: 06/07/2008 13:00:00 end: 07/07/2008 22:00:00
:action (board p2 a1 cadiz) start: 07/07/2008 22:00:00 end: 07/07/2008 23:00:00
:action (zoom a1 cadiz barcelona) start: 07/07/2008 23:00:00 end: 10/07/2008 15:00:00
:action (debark p2 a1 barcelona) start: 10/07/2008 15:00:00 end: 10/07/2008 16:00:00
:action (zoom a1 barcelona sevilla) start: 10/07/2008 16:00:00 end: 12/07/2008 20:00:00
:action (debark p3 a1 sevilla) start: 12/07/2008 20:00:00 end: 12/07/2008 21:00:00
Number of actions: 12 (17)
Expansions: 23
Generated nodes: 40
Inferences: 0
Time in seconds: 0
Real Time: 0.000101563
Used Time: 5.0664e-10
System Time: 3.35276e-10
```

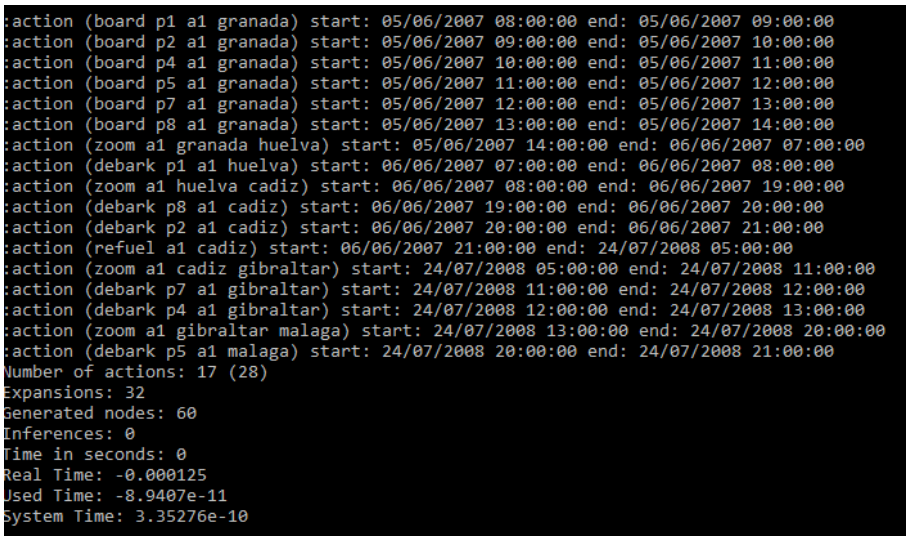
Vemos que el ejercicio se resuelve adecuadamente.

El segundo problema es como el primero, solo que esta vez tenemos otro avión en Almería. El primer avión tiene un límite de fuel de 1000, y el segundo de 20000, por lo que se tendrán que usar ambos aviones.



Vemos que en el avión a1, las personas de granada se montan en el avión, pero como éste ha alcanzado el límite de fuel, no puede despegar. Por lo tanto aparece el avión a2, que lleva a p2 a Barcelona.

El tercer problema es como el primero, solo que esta vez tenemos 10 personas en granada que tienen diferentes destinos.



Vemos que se resuelve el problema correctamente.