Memoria Relación de Ejercicios PDDL

Alejandro Alcalde, *Universidad de Granada*

31 de mayo de 2015

En ésta memoria se describe cómo se han planteado, y resuelto, los distintos problemas propuestos sobre PDDL en la asignatura Técnicas de los Sisteas Inteligentes.

1. Ejericio 1

Escribir un dominio de planificación en PDDL para que un planificador pueda encontrar planes de actuación para uno o varios robots como soluciones a problemas de distribución de paquetes entre habitaciones.

1.1. Representación de los objetos

Para representar los objetos en éste dominio, se ha usado herencia. Para ello, se definió un `ojetofisico`, extendiendo de `object`. `cosas` extiende de `objetofisico`, éstas `cosas`, serán los paquetes y los robots.

1.2. Predicados

Se han definido los siguientes predicados:

- `(at ?r - objetofisico ?h - habitacion)`: Verdadero sii `?r` es un `objetofisico`, `?h` es una `habitacion` y el objeto `?r` está en la habitación `?h`. Para poder reutilzar éste predicado para indicar que, tanto un paquete, como un robot, están en una habitación, el tipo de objeto que se requiere es de tipo `objetofisico`, ya que `paquete, robot` extienden de `cosas`, que a su vez extiende de `objetofisico`. Por tanto al predicado `at` podremos pasarle como primer parámetro tanto un paquete como un robot.
- `(conectada ?h1 - habitacion ?h2 - habitacion)`: Verdadero sii `?h1` y `?h2` son `habitaciones` y están conectadas.
- `(free ?r - robot)`: Verdadero sii `?r` es un `robot`, y el robot `?r` está libre, es decir, puede coger un paquete.
- `(carry ?p - paquete ?r - robot)`: Verdadero sii `?p` es un `paquete`, `?r` un `robot` y el robot `?r` lleva el `paquete ?p`.

1.3. Acciones

Las acciones posibles en el dominio son:

- `move`: Representa la posibilidad de que el robot se mueva entre habitaciones.
  - Parámetros: `(?r - robot ?from ?to - habitacion)`.
  - Precondiciones: `(at ?r ?from),(conectada ?from ?to)`, que el robot `?r` esté en la habitación de partida `?from`, y que ésta última esté conectada con la habitación `?to`.
  - Efectos: `(at ?r ?to), (not (at ?r ?from))`, el robot `?r` estará en la habitación `?to`, y dejará de estar en la habitación `?from`.
- `pick`: Representa la posibilidad de que el robot coja un objeto de la habitación, en éste caso un paquete.

- Parámetros: `(?obj - paquete ?h - habitacion ?r - robot)`.
- Precondiciones: `(at ?obj ?h), (at ?r ?h), (free ?r)`. El objeto `?obj` a coger está en la habitación `?h`, al igual que el robot `?r`. El robot `?r` está libre para coger el objeto.
- Efectos: `(carry ?obj ?r), (not (at ?obj ?h)), (not (free ?r))`. El robot `?r` lleva el objeto `?obj`, el objeto ya no se encuentra en la habitación `?h` y el robot `?r` ya no está libre.

▪ `drop`: Representa la posibilidad de que el robot deje un objeto en una habitación.

- Parámetros: `(?obj - paquete ?h - habitacion ?r - robot)`.
- Precondiciones: `(carry ?obj ?r), (at ?r ?h)`. El robot `?r` lleva al objeto `?obj` y el robot está en la habitación `?h`.
- Efectos: `(at ?obj ?h), (free ?r), (not (carry ?obj ?r))`. El el objeto `?obj` está en la habitación `?h`, el robot `?r` está libre y ya no lleva al objeto `?obj`.

A continuación se muestra el fichero con el código fuente.

Dominio1.pddl

```
1  (define (domain RobotDistribuidor)
2
3    (:requirements :typing)
4    (:types objetofisico - object
5            cosas - objetofisico
6            paquete robot - cosas
7            habitacion)
8
9    (:predicates
10     ;; Verdadero si ?r está en ?h
11     (at ?r - objetofisico ?h - habitacion)
12
13     ;; Verdadero si ambas son habitaciones y están conectadas
14     (conectada ?h1 - habitacion ?h2 - habitacion)
15
16     ;; El robot va vacío
17     (free ?r - robot)
18
19     ;; El robot lleva al paquete
20     (carry ?p - paquete ?r - robot)
21   )
22
23   ;; El robot se puede mover de una habitación a otra
24   (:action move
25     :parameters (?r - robot ?from ?to - habitacion)
26     :precondition (and
27                     (at ?r ?from)
```

```
28                    (conectada ?from ?to))
29     :effect (and
30               (at ?r ?to)
31               (not (at ?r ?from)))
32   )
33
34   ;; El robot puede coger ?obj en la habitación ?h
35   (:action pick
36     :parameters (?obj - paquete ?h - habitacion ?r - robot)
37     :precondition (and
38                    (at ?obj ?h)
39                    (at ?r ?h)
40                    (free ?r))
41     :effect (and
42               (carry ?obj ?r )
43               (not (at ?obj ?h))
44               (not (free ?r)))
45   )
46
47   ;; El robot puede soltar ?obj en la habitación ?h
48   (:action drop
49     :parameters (?obj - paquete ?h - habitacion ?r - robot)
50     :precondition (and
51                    (carry ?obj ?r)
52                    (at ?r ?h))
53     :effect (and
54               (at ?obj ?h)
55               (free ?r)
56               (not (carry ?obj ?r)))
57   )
58 )
```

1.4. Problemas

Para éste ejercicio se han diseñado dos tipos de problemas, el primero más sencillo, con un solo robot, dos paquetes y tres habitaciones. Las habitaciones conectadas son la primera con la segunda, y la segunda con la tercera. A continuación se muestra la definición del problema:

p0e1.pddl

```
1  (define (problem RDistribuye-1)
2  (:domain RobotDistribuidor)
3  (:objects
4    r1 - robot
5    p1 - paquete
6    p2 - paquete
7    hab0 - habitacion
8    hab1 - habitacion
9    hab2 - habitacion
10
11   )
12 (:init
13   (at r1 hab0)
```

```
  (at p1 hab0)
  (at p2 hab2)
  (free r1)
  (conectada hab0 hab1)
  (conectada hab1 hab0)
  (conectada hab2 hab1)
  (conectada hab1 hab2)
)
(:goal (and
  (at r1 hab1)
  (at p2 hab0)
  (at p1 hab2)
  ))
)
```

La ejecución de FF muestra el siguiente plan:

```
0: PICK P1 HAB0 R1
1: MOVE R1 HAB0 HAB1
2: MOVE R1 HAB1 HAB2
3: DROP P1 HAB2 R1
4: PICK P2 HAB2 R1
5: MOVE R1 HAB2 HAB1
6: MOVE R1 HAB1 HAB0
7: DROP P2 HAB0 R1
8: MOVE R1 HAB0 HAB1
```

El segundo problema tiene dos robots, dos paquetes y tres habitaciones, contectadas igual que el problema anterior.

p1e1.pddl

```
(define (problem RDistribuye-1)
(:domain RobotDistribuidor)
(:objects
  r1 r2 - robot
  p1 - paquete
  p2 - paquete
  hab0 - habitacion
  hab1 - habitacion
  hab2 - habitacion

  )
(:init
  (at r1 hab0)
  (at r2 hab2)
  (at p1 hab0)
  (at p2 hab2)
  (free r1)
  (free r2)
  (conectada hab0 hab1)
  (conectada hab1 hab0)
  (conectada hab2 hab1)
```

```
  (conectada hab1 hab2)
)
(:goal (and
  (at r1 hab1)
  (at r2 hab1)
  (at p2 hab0)
  (at p1 hab2)
  ))
)
```

El plan calculado es:

```
0: PICK P2 HAB2 R2
1: PICK P1 HAB0 R1
2: MOVE R1 HAB0 HAB1
3: MOVE R2 HAB2 HAB1
4: MOVE R2 HAB1 HAB0
5: DROP P2 HAB0 R2
6: MOVE R2 HAB0 HAB1
7: MOVE R1 HAB1 HAB2
8: DROP P1 HAB2 R1
9: MOVE R1 HAB2 HAB1
```

2. Ejercicio 2

Escribir un dominio de planificación en PDDL, modificando el dominio del anterior ejercicio, de tal manera que se tenga en cuenta que la acción de moverse de una habitación a otra consume una cantidad de batería y, por tanto, requiere que el robot tenga nivel de batería para moverse. Además, considerar que hay una nueva acción de carga de batería que permite reponer la batería. Considerar para ello que se ha definido un predicado (cambio n1 n2 – nivelbat) que representa un cambio en el nivel de batería desde un nivel n1 a un nivel n2. En En el material de esta sesión de prácticas hay un fichero ejemplo de un problema para este tipo de dominio.

2.1. Dominio

Éste dominio se ha especificado haciendo uso de funciones, (`:functions`). Para ello, se ha modificado el dominio1, añadiendo la siguiente función:

```
(:functions
  (battery-left ?r - robot)
)
```

Que devolverá la cantidad de batería disponible.

Además, la acción `move` se ha modificado para que modifique el nivel de batería de un robot, en su precondición se exige que el nivel de batería tenga un valor mínimo `(>= (battery-left ?r) 2)`, y como efecto, se decrementa el nivel de batería en un valor `(decrease (battery-left ?r) 2)`.

Cuando el nivel de batería no permite al robot moverse, éste puede recargar la batería con la acción `charge`:

```
(:action charge
  :parameters (?r - robot)
  :precondition (< (battery-left ?r ) 2)
  :effect (assign (battery-left ?r ) 10)
)
```

Ésta acción requiere que el nivel de batería sea menor que 2, y asigna un valor de 10 a la carga.

El código completo del dominio es:

Dominio2.pddl

```
1  (define (domain RobotDistribuidor)
2
3    (:requirements :typing :fluents)
4    (:types objetofisico - object
5            cosas - objetofisico
6            paquete robot - cosas
7            habitacion)
8
9    (:predicates
10     ;; Verdadero si ?r está en ?h
11     (at ?r - objetofisico ?h - habitacion)
12
13     ;; Verdadero si ambas son habitaciones y están conectadas
14     (conectada ?h1 - habitacion ?h2 - habitacion)
15
16     ;; El robot va vacío
17     (free ?r - robot)
18
19     ;; El robot lleva al paquete
20     (carry ?p - paquete ?r - robot)
21   )
22
23   (:functions
24     (battery-left ?r - robot)
25   )
26
27   ;; El robot se puede mover de una habitación a otra
28   (:action move
29     :parameters (?r - robot ?from ?to - habitacion)
30     :precondition (and
31                     (at ?r ?from)
```

```
                    (conectada ?from ?to)
                    (>= (battery-left ?r) 2)
                    )
    :effect (and
              (at ?r ?to)
              (not (at ?r ?from))
              (decrease (battery-left ?r) 2))
  )

  ;; El robot puede coger ?obj en la habitación ?h
  (:action pick
    :parameters (?obj - paquete ?h - habitacion ?r - robot)
    :precondition (and
                    (at ?obj ?h)
                    (at ?r ?h)
                    (free ?r))
    :effect (and
              (carry ?obj ?r )
              (not (at ?obj ?h))
              (not (free ?r)))
  )

  ;; El robot puede soltar ?obj en la habitación ?h
  (:action drop
    :parameters (?obj - paquete ?h - habitacion ?r - robot)
    :precondition (and
                    (carry ?obj ?r)
                    (at ?r ?h))
    :effect (and
              (at ?obj ?h)
              (free ?r)
              (not (carry ?obj ?r)))
  )

  (:action charge
    :parameters (?r - robot)
    :precondition (< (battery-left ?r ) 2)
    :effect (assign (battery-left ?r ) 10)
  )

)
```

2.2. Problema

El problema definido para éste dominio consiste en 2 robots, 10 paquetes y 4 habitaciones. Los niveles de batería de ambos robots están inicializados a 1.

p1e2.pddl

```
(define (problem RDistribuye-1)
(:domain RobotDistribuidor)
(:objects
  r1 r2 - robot
  p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 - paquete
  hab0 hab1 hab2 hab3 hab4 - habitacion
```

```
  )
(:init
  (at r1 hab0)
  (at r2 hab0)
  (at p1 hab0)
  (at p2 hab0)
  (at p3 hab0)
  (at p4 hab0)
  (at p5 hab0)
  (at p6 hab2)
  (at p7 hab2)
  (at p8 hab2)
  (at p9 hab2)
  (at p10 hab2)
  (free r1)
  (free r2)
  (= (battery-left r1 ) 1)
  (= (battery-left r2 ) 1)
  (conectada hab0 hab1)
  (conectada hab1 hab0)
  (conectada hab2 hab1)
  (conectada hab1 hab2)
  (conectada hab1 hab3)
  (conectada hab3 hab1)
  (conectada hab3 hab4)
  (conectada hab4 hab3)

)
(:goal (and
  (at p6 hab0)
  (at p7 hab0)
  (at p8 hab0)
  (at p9 hab0)
  (at p10 hab0)
  (at p1 hab2)
  (at p2 hab2)
  (at p3 hab2)
  (at p4 hab2)
  (at p5 hab2)
  (at r1 hab2)
  (at r2 hab2)
  ))

)
```

El plan generado:

```
0: CHARGE R2
1: CHARGE R1
2: MOVE R1 HAB0 HAB1
3: MOVE R2 HAB0 HAB1
4: MOVE R1 HAB1 HAB2
5: MOVE R2 HAB1 HAB0
6: MOVE R1 HAB2 HAB1
```

```
 7: PICK P1 HAB0 R2
 8: MOVE R2 HAB0 HAB1
 9: MOVE R2 HAB1 HAB2
10: DROP P1 HAB2 R2
11: MOVE R1 HAB1 HAB2
12: PICK P6 HAB2 R2
13: MOVE R2 HAB2 HAB1
14: CHARGE R2
15: MOVE R2 HAB1 HAB0
16: DROP P6 HAB0 R2
17: MOVE R2 HAB0 HAB1
18: MOVE R2 HAB1 HAB0
19: PICK P2 HAB0 R2
20: MOVE R2 HAB0 HAB1
21: MOVE R2 HAB1 HAB2
22: DROP P2 HAB2 R2
23: PICK P7 HAB2 R1
24: MOVE R1 HAB2 HAB1
25: CHARGE R1
26: MOVE R1 HAB1 HAB0
27: DROP P7 HAB0 R1
28: PICK P3 HAB0 R1
29: MOVE R1 HAB0 HAB1
30: MOVE R1 HAB1 HAB2
31: DROP P3 HAB2 R1
32: PICK P8 HAB2 R1
33: MOVE R1 HAB2 HAB1
34: MOVE R1 HAB1 HAB0
35: CHARGE R1
36: DROP P8 HAB0 R1
37: PICK P4 HAB0 R1
38: MOVE R1 HAB0 HAB1
39: MOVE R1 HAB1 HAB2
40: DROP P4 HAB2 R1
41: PICK P9 HAB2 R1
42: MOVE R1 HAB2 HAB1
43: MOVE R1 HAB1 HAB0
44: DROP P9 HAB0 R1
45: PICK P5 HAB0 R1
46: MOVE R1 HAB0 HAB1
47: CHARGE R1
48: MOVE R1 HAB1 HAB2
49: DROP P5 HAB2 R1
50: PICK P10 HAB2 R1
51: MOVE R1 HAB2 HAB1
52: MOVE R1 HAB1 HAB0
53: DROP P10 HAB0 R1
54: MOVE R1 HAB0 HAB1
55: MOVE R1 HAB1 HAB2
```

3. Ejercicio 3

Escribir un dominio de planificación en PDDL, modificando el dominio del anterior ejercicio, de manera que se puedan utilizar ahora dos acciones diferentes, moverse rápido y moverse lento tales que moverse rápido consume más unidades de fuel que moverse lento. Probarlo con varios problemas.

3.1. Dominio

Éste problema es una extensión del anterior, simplemente se ha añadido una acción `move-fast` que consume más unidades de batería:

```
(:action move-fast
  :parameters (?r - robot ?from ?to - habitacion)
  :precondition (and
                  (at ?r ?from)
                  (conectada ?from ?to)
                  (>= (battery-left ?r) 10)
                  )
  :effect (and
            (at ?r ?to)
            (not (at ?r ?from))
            (decrease (battery-left ?r) 10))
)
```

3.2. Problema

El ejemplo de problema usado en éste caso tiene 3 robots, 10 paquetes y 4 habitaciones, los niveles de batería inicialmente son 100.

p1e3.pddl

```
(define (problem RDistribuye-1)
(:domain RobotDistribuidor)
(:objects
  r1 r2 r3 - robot
  p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 - paquete
  hab0 hab1 hab2 hab3 hab4 - habitacion
  )
(:init
  (at r1 hab0)
  (at r2 hab0)
  (at r3 hab4)
  (at p1 hab0)
  (at p2 hab0)
  (at p3 hab0)
  (at p4 hab0)
  (at p5 hab0)
  (at p6 hab2)
  (at p7 hab2)
```

```
  (at p8 hab2)
  (at p9 hab4)
  (at p10 hab2)
  (free r1)
  (free r2)
  (free r3)
  (= (battery-left r1 ) 100)
  (= (battery-left r2 ) 100)
  (= (battery-left r3 ) 100)
  (conectada hab0 hab1)
  (conectada hab1 hab0)
  (conectada hab2 hab1)
  (conectada hab1 hab2)
  (conectada hab1 hab3)
  (conectada hab3 hab1)
  (conectada hab3 hab4)
  (conectada hab4 hab3)
)
(:goal (and
  (at p6 hab0)
  (at p7 hab0)
  (at p8 hab0)
  (at p9 hab0)
  (at p10 hab0)
  (at p1 hab2)
  (at p2 hab2)
  (at p3 hab2)
  (at p4 hab2)
  (at p5 hab2)
  (at r1 hab2)
  (at r2 hab2)
  (at r3 hab4)
  ))

)
```

Y el plan generado:

```
 0: PICK P9 HAB4 R3
 1: MOVE-FAST R3 HAB4 HAB3
 2: MOVE-FAST R1 HAB0 HAB1
 3: MOVE-FAST R2 HAB0 HAB1
 4: MOVE R1 HAB1 HAB2
 5: MOVE-FAST R3 HAB3 HAB1
 6: DROP P9 HAB1 R3
 7: MOVE-FAST R3 HAB1 HAB3
 8: MOVE-FAST R3 HAB3 HAB4
 9: PICK P9 HAB1 R2
10: MOVE R2 HAB1 HAB0
11: DROP P9 HAB0 R2
12: MOVE-FAST R2 HAB0 HAB1
13: MOVE R2 HAB1 HAB0
14: MOVE R1 HAB2 HAB1
15: PICK P1 HAB0 R2
```

```
16: MOVE-FAST R2 HAB0 HAB1
17: MOVE R2 HAB1 HAB2
18: DROP P1 HAB2 R2
19: MOVE R1 HAB1 HAB2
20: PICK P6 HAB2 R2
21: MOVE R2 HAB2 HAB1
22: MOVE R2 HAB1 HAB0
23: DROP P6 HAB0 R2
24: MOVE-FAST R2 HAB0 HAB1
25: MOVE R2 HAB1 HAB0
26: MOVE R1 HAB2 HAB1
27: PICK P2 HAB0 R2
28: MOVE-FAST R2 HAB0 HAB1
29: MOVE R2 HAB1 HAB2
30: DROP P2 HAB2 R2
31: MOVE R1 HAB1 HAB2
32: PICK P7 HAB2 R2
33: MOVE R2 HAB2 HAB1
34: MOVE R2 HAB1 HAB0
35: DROP P7 HAB0 R2
36: MOVE-FAST R2 HAB0 HAB1
37: MOVE R2 HAB1 HAB0
38: PICK P3 HAB0 R2
39: MOVE-FAST R2 HAB0 HAB1
40: MOVE-FAST R2 HAB1 HAB2
41: DROP P3 HAB2 R2
42: PICK P8 HAB2 R1
43: MOVE R1 HAB2 HAB1
44: MOVE R1 HAB1 HAB0
45: DROP P8 HAB0 R1
46: PICK P4 HAB0 R1
47: MOVE-FAST R1 HAB0 HAB1
48: MOVE R1 HAB1 HAB2
49: DROP P4 HAB2 R1
50: PICK P10 HAB2 R1
51: MOVE R1 HAB2 HAB1
52: MOVE R1 HAB1 HAB0
53: DROP P10 HAB0 R1
54: PICK P5 HAB0 R1
55: MOVE-FAST R1 HAB0 HAB1
56: MOVE R1 HAB1 HAB2
57: DROP P5 HAB2 R1
```