UNIVERSIDAD DE GRANADA

Práctica 2: planificación clásica en PDDL

Juan Ocaña Valenzuela

23 de mayo de 2019

Índice

1. Diario de trabajo

Domingo, 5 de mayo de 2019

Empecé la práctica hace unos días, pero no ha sido hasta hoy que he decidido comenzar a documentarla. Llevaba unos días bastante ocupado (la fiesta de la Escuela no se organiza sola), y tenía visita en casa, así que dedicaba relativamente poco espacio de mi sistema nervioso central a pensar en los apasionantes mundos de Belkan, más allá de la eterna duda: ¿es Belkan el personaje? ¿es el nombre del mundo? ¿o quizá un ser omnipresente que gobierna por aquellos lares?
Una vez mi visita se marchó, comencé a plantear el primer ejercicio de la relación, diseñando un mapa, poniendo cosas en él, definiendo las acciones y todo eso. Una vez creí haber terminado el modesto problema 1, en el cual cada personaje debe tener un objeto, ejecuté FF y...
no funcionaba.

No encontraba un plan. No lo encontraba y no sabía por qué. Estaba cansado y no me apetecía hacer nada, así que mandé a la mierda los estrafalarios mundos de Belkan y me puse a jugar a Persona 5.

Hoy es un día nuevo, y nada más levantarme y desayunar he pensado que quizás podría hacer un pequeño esfuerzo por saber qué le pasaba a mi problema, y cómo arreglarlo. Y es que nuestro pequeño bicho no puede entregar objetos a los personajes si no lo hemos definido en el problema. Qué cosas, eh.

Bautizado como Patrick Dotimas (Patrick para los amigos), ahora que está en el problema encuentra una solución. Ya tenemos un problema resuelto... O no.

Observando el plan, resulta que, en un acto de amor incondicional, Patrick se entrega a sí mismo a los personajes, y no los objetos que coge. En lugar de algo como:

```
GIVE PATRICK ROSE R20 PRINCESS
```

tenemos algo así:

```
GIVE PATRICK PATRICK R20 PRINCESS
```

A ver cómo lo soluciono... De momento voy a comer.

[Actualización - 19:30]
Después de procrastinar un buen rato (y jugar otro rato a Persona 5), me he dado cuenta de que el dominio no estaba mal del todo. Patrick cogía sus objetos y los entregaba bien, pero no sabía qué objeto tenía ni daba. Lo he solucionado añadiendo un predicado más, `on_hand`, que viene explicado en su sitio.

Acabo de caer en una cosa... Si Belkan es el personaje, y nuestro jugador se llama Patrick, ¿son estos los extraños mundos de Patrick? En fin, voy a ponerme con el generador de problemas, a ver si marcha.

Sábado, 11 de mayo de 2019

Tras la fiesta de la escuela uno se encuentra destrozado y libre a partes iguales, así que seguir con la práctica era una buena idea. Me puse con el parser de los dos primeros ejercicios, y tras darle algunas vueltas me creaba problemas que deberían funcionar pero no lo hacían. ¿Por qué? Por los paréntesis. Siempre son los paréntesis.

Miércoles, 15 de mayo de 2019

Los ejercicios tres y cuatro han sido fáciles, pero el parser del ejercicio cuatro me parece poco claro. Según el enunciado hay que leer el número de puntos a conseguir, pero introducir la tabla de puntos en el problema no es tan trivial como parecía al prinicipio. En fin, es lo que hay.

Domingo, 19 de mayo de 2019

Me he dado cuenta de que plantear los ejercicios en PDDL es más sencillo que hacer el parser en la mayoría de las ocasiones. Los ejercicios cinco y seis han consistido en modificar un par de cosas de los dominios anteriores, mientras que el parser cada vez se complica más (aunque tampoco demasiado).

Martes, 21 de mayo de 2019

Doy por terminado el trabajo que voy a entregar. Podría hacer más, pero tengo más cosas que hacer y creo que es suficiente y funciona moderadamente bien.

Miércoles, 22 de mayo de 2019

Procedo a documentar la práctica. Hay más trabajo del que parece, ya que hay que poner las cosas bonitas, dibujar un par de mapas, explicar los ejercicios, etc. Pero bueno, es echarle un rato. Lo peor ya ha pasado.

2. Leyenda

En los mapas adjuntos se representan los elementos del dominio de la siguiente forma:
Objetos: escritos en gris, sobre la zona en la que se encuentran. **Personajes:** escritos en negro, sobre la zona en la que se encuentran. **Distancia entre zonas:** representada sobre la unión de las zonas. **Tipo de zona:** representada por el color de la zona:

- **Azul:** agua.
- **Marrón claro:** arena.
- **Marrón oscuro:** piedra.
- **Gris oscuro:** precipicio.

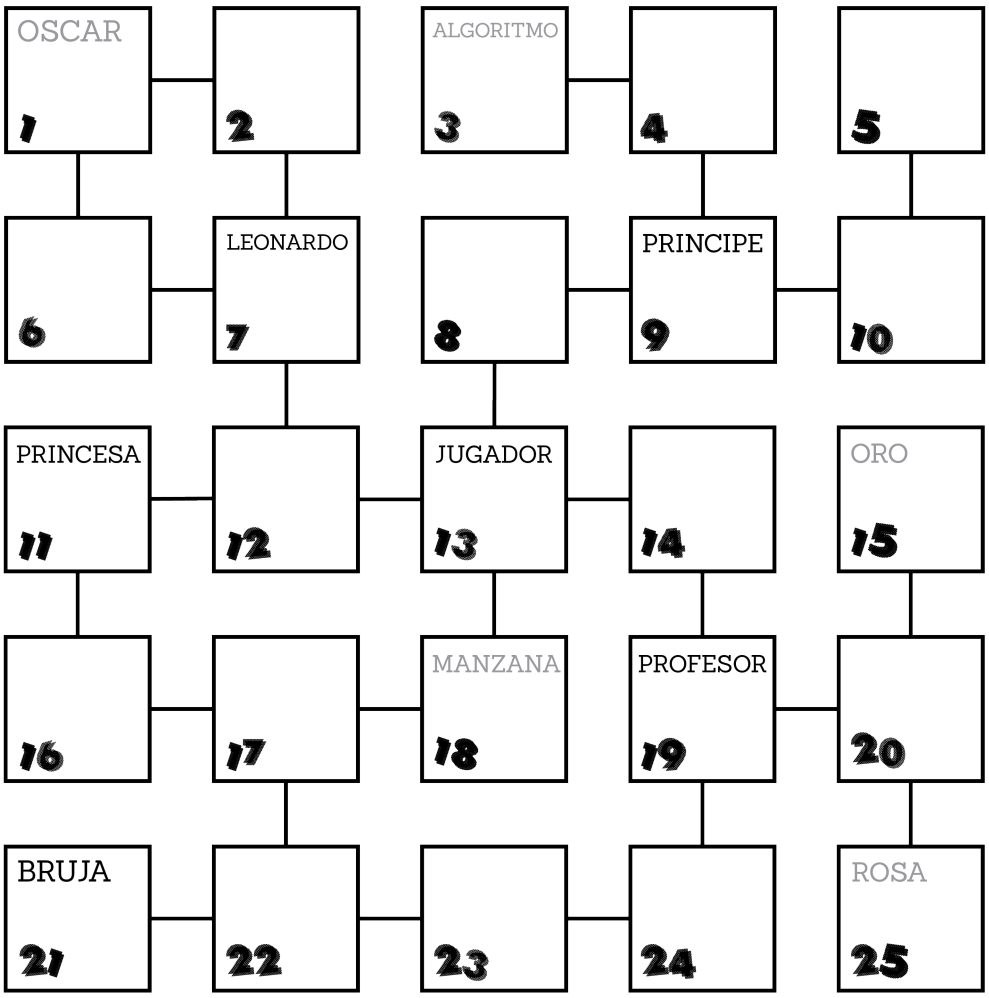
- **Verde:** bosque.

3. Ejercicios

3.1. Ejercicio 1

Definir un dominio y problema de planificación considerando que el jugador podrá estar orientado al norte, sur, este u oeste y desplazarse de una zona a otra siempre que esté correctamente orientado. Por ejemplo, podrá desplazarse a una zona al norte de su zona actual, si está orientado al norte.

3.2. Mapa problema 1



3.2.1. Apartado A

Representar en el dominio los objetos del mundo.

Para representar los objetos del mundo (jugador, personajes, objetos, habitaciones, caminos, etc.) se han establecido los siguientes tipos, con su jerarquía:

locatable: elemento que se puede localizar en una posición.
  character: personaje.
    player: jugador.
    npc: personaje no jugador.

object: objeto.
orientation: orientación de un elemento.
room: habitación del dominio.

3.2.2. Apartado B

Representar predicados que permitan describir los estados del mundo.

Se han considerado los siguientes predicados:

at
`(at ?r - room ?l - locatable)`
Un elemento `?l` se encuentra en la habitación `?r`.

on_floor
`(on_floor ?o - object)`
Un objeto `?o` se encuentra en el suelo.

compass
`(compass ?o - orientation)`
La orientación del personaje es `?o`.

path
`(path ?r1 ?r2 - room ?o - orientation)`
Existe un camino entre `?r1` y `?r2`, en el que la segunda habitación se encuentra con una orientación `?o` respecto de la primera.

has_object
`(has_object ?c - character)`
Un personaje `?c` tiene un objeto.

on_hand
`(on_hand ?o - object)`
El jugador tiene un objeto `?o` en la mano.

3.2.3. Apartado C

Representar las siguientes acciones del jugador: girar a la izquierda, girar a la derecha, ir, coger, dejar, entregar.

Girar a la izquierda
`(:action TURN_LEFT)`
Dada una orientación, el jugador mirará a aquella a su izquierda:

- N $\rightarrow$ W

- W → S
- S → E
- E → N

Girar a la derecha
`(:action TURN_RIGHT)`
Dada una orientación, el jugador mirará a aquella a su derecha:

- N → E
- W → N
- S → W
- E → S

Girar 180 grados
`(:action TURN_180)`
Dada una orientación, el jugador mirará a aquella a su espalda:

- N → S
- W → E
- S → N
- E → W

Coger un objeto
`(:action PICK)`
Si el jugador se encuentra en la misma habitación que un objeto y este se halla en el suelo, el jugador podrá cogerlo. El objeto dejará de estar en la habitación y en el suelo.

Soltar un objeto
`(:action DROP)`
Si el jugador tiene un objeto, lo soltará. El objeto pasará a estar en la misma habitación que el jugador, y en el suelo.

Dar un objeto a un personaje
`(:action GIVE)`
Si el jugador tiene un objeto, el npc no, y ambos se encuentran en la misma habitación, el jugador le dará el objeto al npc. El jugador pasará a no tener objeto, y el npc sí lo hará.

Moverse a una habitación contigua
`(:action GO)`
El jugador se moverá a la habitación hacia la que esté orientado.

3.2.4. Apartado D

Plantear un problema de planificación con un estado inicial con 25 zonas conectadas arbitrariamente en el que aparezcan situados los 5 personajes en distintas zonas y al menos 5 objetos. El objetivo de este problema consistirá en conseguir que todos los personajes tengan al menos un objeto.

El objetivo es que todos los personajes dispongan de un objeto, es decir, que se cumpla:

```
(AND
  (has_object leonardo)
  (has_object prince)
  (has_object princess)
  (has_object professor)
  (has_object witch)
)
```

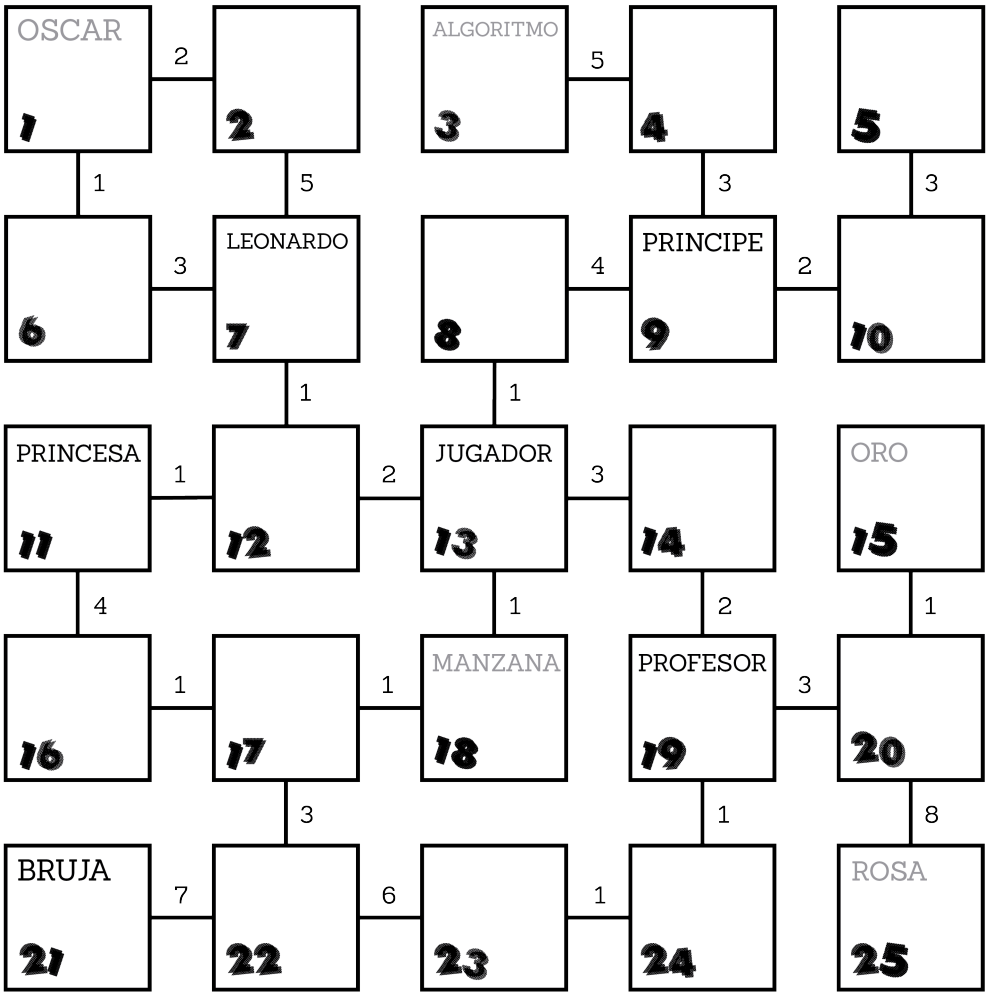
El plan generado por FF es el siguiente:

0: TURN_180 N 1: GO PATRICK R13 R18 S
2: TURN_180 S
3: PICK PATRICK APPLE R18
4: TURN_LEFT N
5: GO PATRICK R18 R17 W
6: GO PATRICK R17 R16 W
7: TURN_RIGHT W
8: TURN_RIGHT N
9: GO PATRICK R16 R17 E
10: TURN_RIGHT E
11: GO PATRICK R17 R22 S
12: TURN_LEFT S
13: GIVE PATRICK APPLE R22 WITCH
14: GO PATRICK R22 R23 E
15: GO PATRICK R23 R24 E
16: TURN_LEFT E
17: GO PATRICK R24 R19 N
18: TURN_RIGHT N
19: GO PATRICK R19 R20 E
20: TURN_LEFT E
21: GO PATRICK R20 R15 N
22: TURN_180 N
23: PICK PATRICK GOLD R15
24: GO PATRICK R15 R20 S
25: TURN_180 S
26: TURN_LEFT N
27: GO PATRICK R20 R19 W
28: TURN_180 W
29: GIVE PATRICK GOLD R19 PROFESSOR
30: GO PATRICK R19 R20 E
31: TURN_RIGHT E
32: GO PATRICK R20 R25 S
33: TURN_180 S
34: PICK PATRICK ROSE R25
35: GO PATRICK R25 R20 N
36: TURN_LEFT N
37: GO PATRICK R20 R19 W
38: TURN_RIGHT W
39: GO PATRICK R19 R14 N
40: TURN_LEFT N
41: GO PATRICK R14 R13 W
42: GO PATRICK R13 R12 W
43: GO PATRICK R12 R11 W
44: TURN_RIGHT W
45: TURN_RIGHT N
46: GIVE PATRICK ROSE R11 PRINCESS
47: GO PATRICK R11 R12 E
48: TURN_LEFT E

49: GO PATRICK R12 R7 N
50: GO PATRICK R7 R2 N
51: TURN_LEFT N
52: GO PATRICK R2 R1 W
53: TURN_LEFT W
54: PICK PATRICK OSCAR R1
55: GO PATRICK R1 R6 S
56: TURN_LEFT S
57: GO PATRICK R6 R7 E
58: TURN_RIGHT E
59: GIVE PATRICK OSCAR R7 LEONARDO
60: GO PATRICK R7 R12 S
61: TURN_LEFT S
62: GO PATRICK R12 R13 E
63: TURN_LEFT E
64: GO PATRICK R13 R8 N
65: TURN_RIGHT N
66: GO PATRICK R8 R9 E
67: TURN_LEFT E
68: GO PATRICK R9 R4 N
69: TURN_LEFT N
70: GO PATRICK R4 R3 W
71: TURN_180 W
72: PICK PATRICK ALGORITHM R3
73: GO PATRICK R3 R4 E
74: TURN_RIGHT E
75: GO PATRICK R4 R9 S
76: GIVE PATRICK ALGORITHM R9 PRINCE

3.3. Ejercicio 2

3.3.1. Mapa de problema 1



3.3.2. Apartado A

En el ejercicio 2 se pide adecuar el dominio ya definido a la nueva característica de que los caminos tienen una longitud definida.
Para ello, se ha introducido una nueva función, `(distance)`, definida en el problema. En la regla `GO`, al desplazarse se le suma al coste total del plan el valor definido de `(distance z1 z2)`.

3.3.3. Apartado B

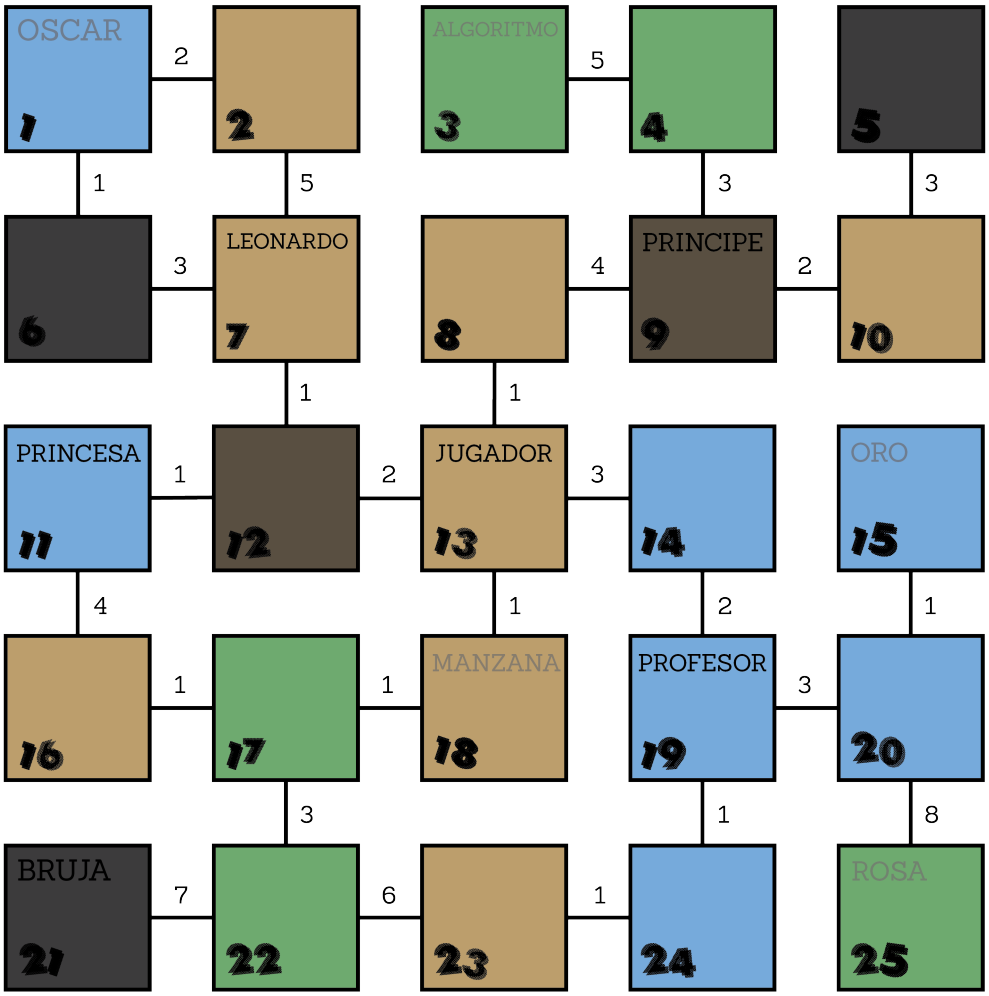
En los problemas definidos para el dominio del ejercicio 2, ahora se indicará la distancia entre dos zonas, asignando el valor de `(distance ?z1 ?z2)` junto con la declaración de los caminos. Por ejemplo, si hay un camino de distancia 3 entre z1 y z7, en el problema aparecería:
`(path z1 z7) (path z7 z1) (= (distance z1 z7) 3) (= (distance z7 z1) 3)`

3.3.4. Apartado C

El parser se ha ampliado para leer la distancia entre las zonas. Para ello, en lugar de guardar una 2-tupla con las zonas conectadas en el vector correspondiente, se guarda una 3-tupla, con las zonas y la distancia entre ellas. Posteriormente se interpreta y escribe en el fichero resultado.

3.4. Ejercicio 3

3.4.1. Mapa de problema 1



3.4.2. Apartados A y B

En el ejercicio 3 se pide introducir terrenos a las diferentes zonas, necesitando objetos especiales para pasar por algunos de ellos. Además, ahora el jugador dispondrá de una mochila, en la que podrá guardar un objeto extra.
Se ha modificado la forma en la que se representan los objetos que tiene el jugador. Primero, si tiene la mano vacía, existirá en la base de hechos (`hand_empty`), y si lo está la mochila, (`bag_empty`).

Si tiene algo en la mano, existirá en la base de hechos `(on_hand ?o - object)`, y lo mismo en la mochila con `(on_bag ?o - object)`.
Si necesitamos pasar por terrenos por los que necesitamos ropa especial, comprobará si la tenemos en la mano o en la mochila en la acción `GO`.

3.4.3. Apartado C

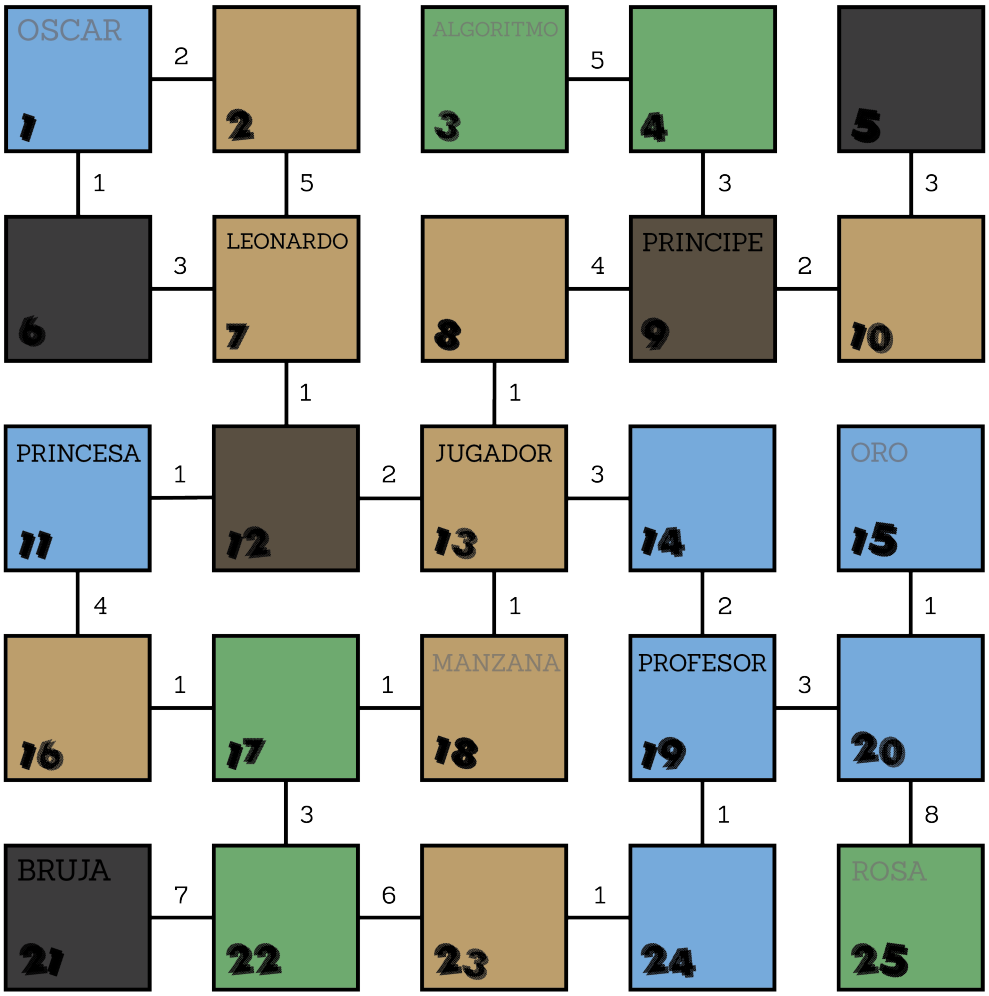
En los problemas definidos para el dominio del ejercicio 3, se indicará el terreno de cada zona con el predicado `(room_type ?z - room ?t - terrain)`.

3.4.4. Apartado D

El parser se ha ampliado para incorporar el tipo de terreno al problema. Si no se definen las zapatillas o el bikini, se situarán en una zona arbitraria.

3.5. Ejercicio 4

3.5.1. Mapa de problema 1



3.5.2. Apartado A

En el ejercicio 4 se proporciona una tabla de puntuación, con la que se define cuántos puntos gana el jugador por entregar un determinado objeto a un determinado personaje. El dominio se ha modificado añadiendo las funciones `(points_given ?ch - npc ?o - object)` y `(points_earned)`, que indican los puntos dados por un npc dado un objeto y los puntos totales acumulados, respectivamente.

3.5.3. Apartado B

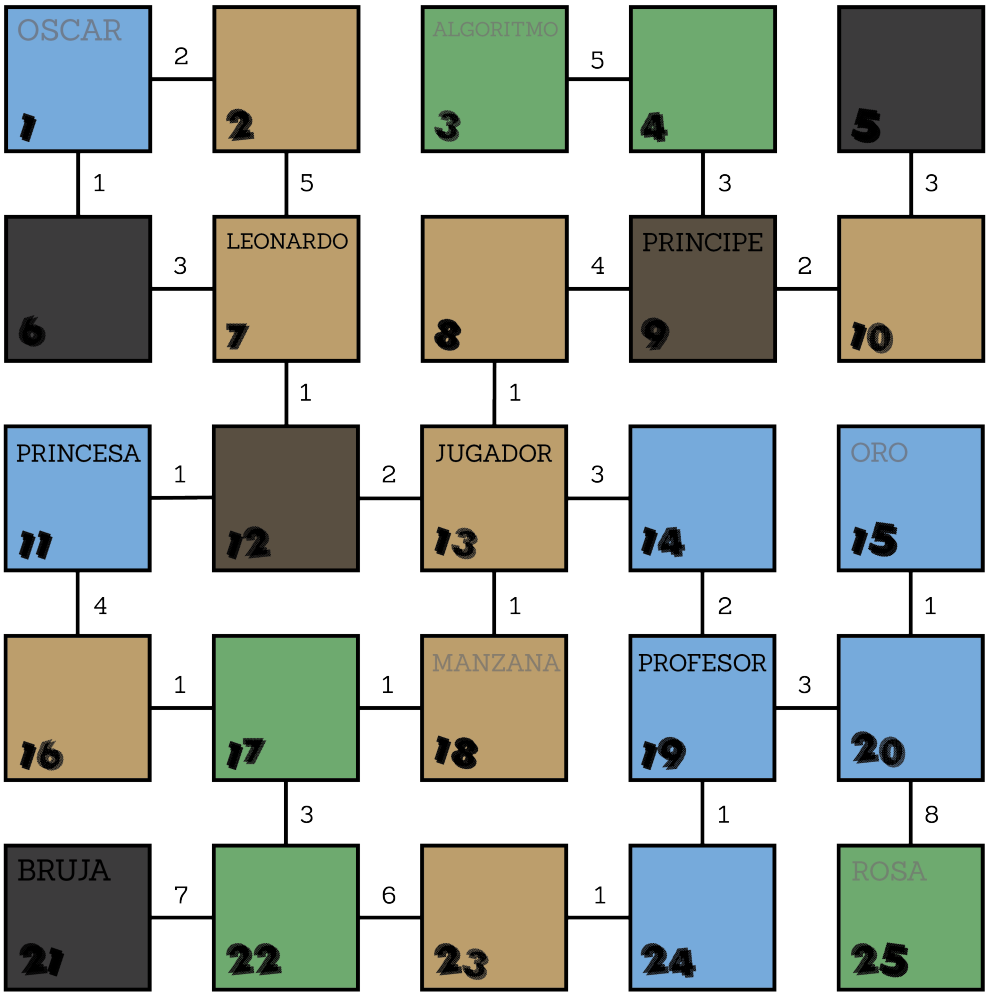
La tabla se ha representado utilizando la función (`points_given ?ch - npc ?o - object`).

3.5.4. Apartado C

El parser se ha actualizado para leer los puntos a alcanzar e incluirlos como parte de la meta.

3.6. Ejercicio 5

3.6.1. Mapa de problema 1



3.6.2. Apartado A

Se ha ampliado el dominio anterior para representar el bolsillo de cada personaje añadiendo las funciones (`stock ?ch - npc`) y (`max_stock ?ch - npc`), que representan los objetos obtenidos y la capacidad máxima, respectivamente.
En la acción `GIVE`, se ha añadido que compruebe si el personaje tiene hueco para más objetos, y en caso de tenerlo, se le entregue y se incremente en uno el total de objetos. Si no, no puede entregarlo.

3.6.3. Apartado B

El problema se ha extendido añadiendo el stock máximo de los NPC, y comenzando todos con cero objetos.

3.6.4. Apartado C

El parser se ha ampliado para representar el stock de cada personaje, añadiendo también que todos comienzan con cero objetos.

3.7. Ejercicio 6

3.7.1. Mapa de problema 1

3.7.2. Apartado A

En este ejercicio se ha introducido un nuevo jugador, y por tanto, todos los predicados que utilizábamos anteriormente para representar el estado del jugador ahora necesitan un parámetro. Se han modificado para que ambos jugadores puedan hacer uso de ellos. También se ha añadido la función `(total_points)`, para representar los puntos conseguidos en conjunto.

3.7.3. Apartado B

Se han indicado las condiciones iniciales del segundo jugador en el problema, y los puntos a conseguir.