Técnicas de los sistemas inteligentes
Práctica 2

Guillermo Gómez Trenado | 77820354-S

22 de mayo de 2018

Decisiones generales

Hay algunas decisiones que son transversales a todos los problemas: he sacrificado la versatilidad de las acciones por su legibilidad, es decir, en funciones donde la acción se realiza sobre un tipo con subtipos, como desplazarse o tirar un objeto al suelo, he definido sendas acciones para las distintas situaciones conflictivas, pues si no tenía que definir variables sólo de comprobación en los parámetros que eran posteriormente imprimidas por pantalla y no tenían ningún significado ni sentido, pero FF las necesita para construir los estados de verdad; en segundo lugar, y de forma parecida al anterior, he creado el predicado *handFull* y *bagFull* a fin de evitar tener que crear otro parámetro que será imprimido en pantalla y utilizarlo para el *for all* sobre *carriesHand* y *carriesBag* respectivamente.

Todos los resultados de los problemas están comprobados y son correctos, pero sólo comento aquellos que puedan resultar interesantes. El resto de salidas se encuentran al final del PDF con el nombre del problema para facilitar la lectura.

1. Definición general del modelo y el problema

1.1. Objetos del mundo

En este primer ejercicio la motivación es evidente, he definido un único tipo de personaje y un único tipo de regalo, y después, los tipos específicos serán instancias del tipo genérico, definiendo sus propiedades posteriormente en la definición del problema. *Orientation* por otro lado, define los cuatro puntos cardinales y se definen en el problema.

```
(:types
        character
        gift
        robot character gift - locatable
        orientation
```

```
        location
)
```

1.2. Predicados

Lo más interesante en este apartado es el predicado *link* que es direccionado y relaciona no sólo dos localizaciones sino la orientación de la segunda respecto a la primera, esto lo utilizaremos posteriormente junto a *facing* para comprobar si un robot está en la dicrreción correcta para seguir un camino. *Right* y *Left* expresan la relación de dos puntos cardinales, es decir, norte está a la izquierda de este y este a la derecha del primero. *hasGift* define si un personaje tiene un objeto o no, posteriormente lo cambiaremos por un valor numérico.

```
(:predicates
        (link ?l1 - location ?l2 - location ?o - orientation)
    (at ?x - locatable ?l - location)
    (facing ?r - robot ?o -orientation)
    (right ?o1 -orientation ?o2 -orientation)
    (left ?o1 -orientation ?o2 -orientation)
    (carries ?r -robot ?o -gift)
    (handEmpty ?r -robot)
    (hasGift ?c -character)
)
```

1.3. Acciones

Esta acción es sencilla, comprueba la posición, la orientación y un enlace con tales características, tras esto falsea la anterior posición y hace verdadera la nueva.

```
(:action GOTO
    :parameters (?r - robot ?o -orientation ?l1 -location ?l2 -location)
    :precondition (and
        (at ?r ?l1)
        (facing ?r ?o)
        (link ?l1 ?l2 ?o)
    )
    :effect (and
        (not (at ?r ?l1))
        (at ?r ?l2)
    )
)
```

Aquí las precondiciones sólo sirven para obtener los valores con los que definir la nueva orientación.

```
(:action TURN−RIGHT
    :parameters (?r −robot ?o1 −orientation ?o2 −orientation)
    :precondition (and
        (facing ?r ?o1)
        (right ?o1 ?o2)
    )
    :effect (and
        (not (facing ?r ?o1))
        (facing ?r ?o2)
    )
)

(:action TURN−LEFT
    :parameters (?r −robot ?o1 −orientation ?o2 −orientation)
    :precondition (and
        (facing ?r ?o1)
        (left ?o1 ?o2)
    )
    :effect (and
        (not (facing ?r ?o1))
        (facing ?r ?o2)
    )
)
```

Comprobamos que tenga la mano vacía, y en caso de ser así cogemos el objeto, falseando la actual posición del objeto.

```
(:action PICK
    :parameters (?r −robot ?l −location ?g −gift)
    :precondition (and
        (handEmpty ?r)
        (at ?r ?l)
        (at ?g ?l)
    )
    :effect (and
        (not (handEmpty ?r))
        (not (at ?g ?l))
        (carries ?r ?g)
    )
)
```

La única restricción es que lleve el objeto en la mano, después actualizamos el estado de la mano y la nueva posición del objeto. Nótese que no falseo *carries* porque no nos interesa, las comprobaciones siempre se realizan con *handEmpty*.

```
(:action DROP
    :parameters (?r −robot ?l −location ?g −gift)
```

```
    :precondition (and
        (not (handEmpty ?r))
        (carries ?r ?g)
        (at ?r ?l)
    )
    :effect (and
        (handEmpty ?r)
        (at ?g ?l)
    )
)
```

Si se encuentran en la misma posición y el robot lleva el objeto en la mano se actualiza el estado de la mano y el del personaje.

```
(:action GIVE
    :parameters (?r -robot ?l -location ?g -gift ?c -character)
    :precondition (and
        (at ?r ?l)
        (at ?c ?l)
        (not (handEmpty ?r))
        (carries ?r ?g)
    )
    :effect (and
        (handEmpty ?r)
        (hasGift ?c)
    )
)
```

1.4. Problema planteado

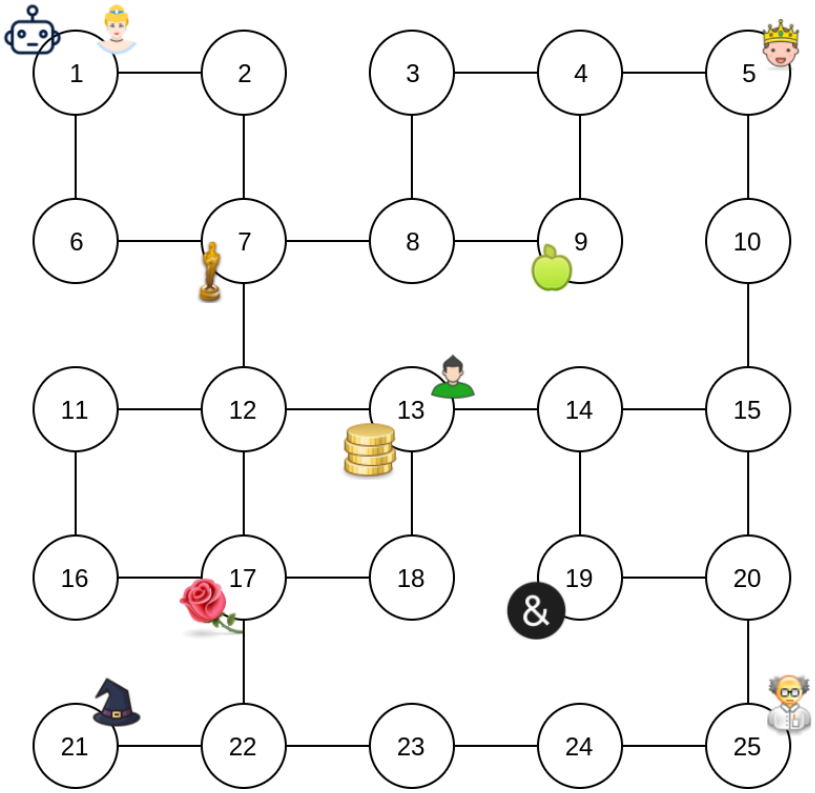
Definimos las 25 casillas, los personajes, los regalos, el robot y las orientaciones posibles.

```
(:objects
    loc1 loc2 loc3 loc4 loc5 loc6 loc7 loc8 loc9 loc10 loc11 loc12 loc13 loc14
        ↪ loc15 loc16 loc17 loc18 loc19 loc20 loc21 loc22 loc23 loc24 loc25
        ↪ - location
    princesa principe bruja profesor leonardo - character
    manzana rosa algoritmo oro oscar - gift
    robot1 - robot
    norte sur este oeste - orientation
)
```

Los caminos, y posiciones los definimos de la siguiente manera. Nótese que por cada camino aparecen dos entradas, una en un sentido y otra en el opuesto, cada uno con su correspondiente relación de orientación.

```
(:init
    ;map
    (link loc1 loc2 este)
    (link loc2 loc1 oeste)
    ...
    (at princesa loc1)
    ...
    (at algoritmo loc19)
    ...
```

Quedando de la siguiente manera



Las relaciones espaciales

```
        (right norte este)
    ...
    (left norte oeste)
    ...
```

Y por último el estado incial del robot, el predicado *handEmpty* pasará posteriormente a *handFull* y así nos ahorraremos tener que definirlo en el problema.

```
        (at robot1 loc1)
    (facing robot1 norte)
    (handEmpty robot1)
```

Finalmente el objetivo lo definimos sencillamente como:

```
(:goal (and
        (hasGift princesa)
        (hasGift principe)
        (hasGift bruja)
        (hasGift profesor)
        (hasGift leonardo)
    )
)
```

Nótese que controlo que tenga al menos un regalo pero no cuántos, posteriormente modificaremos esto en el ejercicio 5 añadiendo una función.

2. Coste de desplazamiento

2.1. Modificar el dominio

Este cambio es sencillo, sólo tenemos que añadir las funciones y modificar *GOTO*

```
(:functions
    (pathLength ?r -robot)
    (linkLength ?l1 -location ?l2 -location)
)
...
(:action GOTO
    ...
    :effect (and
        ...
        (increase (pathLength ?r) (linkLength ?l1 ?l2))
    )
)
```

2.2. Extender el problema (1)

Lo único que he hecho en este paso es añadir por cada camino una distancia, todas con longitud 1.

```
(:init
        ...
        (= (linkLength loc1 loc2) 1)
        (= (linkLength loc2 loc1) 1)
```

```
        ...
```

Tras hacer esto podemos optimizar el plan para reducir la distancia del camino

```
        (:metric minimize (pathLength robot1))
```

Ejecutando el programa de la siguiente manera, nótese que no es el óptimo pero h=1 daba unas ejecuciones demasiado largas.

```
./Metric-FF/ff -d . -o ej2_domain.pddl -f ej2_problem.pddl -O -g 1 -h
    ↪ 2
```

No dibujo el nuevo mapa con la longitud de los caminos pues es el mismo con 1s en los arcos.

2.3. Extender el problema (2)

En la segunda versión del problema actualicé el peso a 100 entre loc1 y loc6 —que era el primer camino que tomaba— para ver que efectivamente evitaba ese camino e iba por loc1-loc2, y así lo hizo.

3. Distintos tipos de zonas y mochila

3.1. Modificar el dominio para zonas

Voy a añadir un tipo *wear* que junto a *gift* anteriormente definido son de tipo *takeable*, así los métodos *PICK* y *DROP* pasan a tener como parámetro un *takeable* en vez de un *gift* y *GIVE* sólo admite un *gift*. Además añadimos un nuevo tipo para el tipo de suelo *groundKind* y predicados para gestionar el tipo de suelo *isKind*, si requiere de ropa especial *isSpecial* y qué ropa necesita *needs*. Nótese que isSpecial sólo se aplica a agua y bosque, mientras que tierra y piedra tienen *isKind* pero no *isSpecial*.

```
(:types
        ...
        gift wear - takeable
        robot character takeable - locatable
        groundKind
)

(:predicates
        ...
        (isSpecial ?l -location)
    (isKind ?l -location ?gk -groundKind)
    (needs ?gk -groundKind ?w -wear)
```

Modificamos GOTO y añadimos GOTO-SPECIAL, podríamos añadir los parámetros a GOTO y hacer una función general pero entonces FF los rellena con basura cuando hacemos el **OR** y la salida en pantalla es poco legible.

```
(:action GOTO
    :parameters (?r - robot ?o -orientation ?l1 -location ?l2 -location)
    :precondition (and
        (at ?r ?l1)
        (facing ?r ?o)
        (link ?l1 ?l2 ?o)
        (not (isSpecial ?l2))
    )
    :effect (and
        (not (at ?r ?l1))
        (at ?r ?l2)
        (increase (pathLength ?r) (linkLength ?l1 ?l2))
    )
)
(:action GOTO-SPECIAL
    :parameters (?r - robot ?o -orientation ?l1 -location ?l2 -location ?gk
        ↪ -groundKind ?w -wear)
    :precondition (and
        (at ?r ?l1)
        (facing ?r ?o)
        (link ?l1 ?l2 ?o)
        (isSpecial ?l2)
        (isKind ?l2 ?gk)
        (needs ?gk ?w)
        (carries ?r ?w)
    )
    :effect (and
        (not (at ?r ?l1))
        (at ?r ?l2)
        (increase (pathLength ?r) (linkLength ?l1 ?l2))
    )
)
```

Además ahora en *DROP* controlamos que no sea un terreno especial, de igual forma que en el caso anterior lo dividimos en dos acciones distintas por el mismo motivo. Hay dos pequeños errores en este ejercicio que no se solucionarán hasta el 6, el primero es que si sólo hay un objeto de cada tipo el programa funciona perfectamente, pero si hay varios *wear* de un mismo tipo en el mapa sucede que no puede soltar el elemento *wear* que está usando aunque tenga otro igual en la mochila, es un problema menor pero limita el espacio solución y no me di cuenta hasta 3 ejercicios más adelante al releer esta parte del código. El segundo es que por los parámetros de *DROP-SPECIAL* no se pueden soltar en el bosque

objeto de tipo gift, en el ejercicio se añadirá una tercera función para dejar los mensajes de salida significativos y la lógica correcta.

```
(:action DROP
    :parameters (?r −robot ?l −location ?g −takeable)
    :precondition (and
        (handFull ?r)
        (carries ?r ?g)
        (not (carriesBag ?r ?g))
        (at ?r ?l)
        (not (isSpecial ?l))
    )
    :effect (and
        (not (handFull ?r))
        (not (carries ?r ?g))
        (at ?g ?l)
    )
)

(:action DROP−SPECIAL
    :parameters (?r −robot ?l −location ?g −wear ?gk −groundKind)
    :precondition (and
        (handFull ?r)
        (carries ?r ?g)
        (not (carriesBag ?r ?g))
        (at ?r ?l)
        (isKind ?l ?gk)
        (not (needs ?gk ?g))
    )
    :effect (and
        (not (handFull ?r))
        (not (carries ?r ?g))
        (at ?g ?l)
    )
)
```

3.2. Modificar el dominio para mochila

Aunque hemos adelantado algunas modificaciones del problema para la mochila vamos a verlas aquí en profundidad.

Definimos nuevos predicados. Como sucedía con el error del apartado anterior, aquí dejar implícitamente definida $carriesHand = carries \wedge \neg carriesBag$ puede generar problemas cuando hay varios objetos *wear* del mismo tipo, también lo soluciono en el ejercicio 6, de momento no general ninguna situación limitante porque los problemas que defino sólo tienen un objeto de cada tipo en el mapa.

```
(:predicates
    ...
    (carries ?r -robot ?o -takeable)
    ;carriesHand -> carries and !carriesBag
    (carriesBag ?r -robot ?o -takeable)
    (handFull ?r -robot)
    (bagFull ?r -robot)
    ...
```

También actualizo todas las acciones donde comprobaba que llevara el objeto cambiando *(carries ?r ?g)* por *(AND (carries ?r ?g) (not (carriesBag ?r ?g))* para comprobar que efectivamente lo lleva en la mano. Esta situación genera que posteriormente, en el problema del ejercicio 4, no pueda dar un regalo si tiene otro del mismo tipo en la mochila, pero como ya he dicho se solucionará más adelante, conservo los errores por ahorrar tiempo por algo que no tendría interés didáctico —ya lo soluciono más adelante— y por reflejar la evolución del razonamiento incluso el equivocado.

Añadimos también los métodos para gestionar la mochila. Son dos métodos complementarios, y donde uno comprueba tener algo en la mano y la mochila vacía el otro lo contrario. Nótese que carries no hace referencia a llevar en la mano si no a llevar en general, comprobando con *handFull* y *bagFull* que nunca lleve más de dos objetos, tanto aquí como en *PICK* y *DROP*.

```
(:action PUSH-BAG
    :parameters (?r -robot ?o -takeable)
    :precondition (and
        (handFull ?r)
        (not (bagFull ?r))
        (carries ?r ?o)
    )
    :effect (and
        (not (handFull ?r))
        (bagFull ?r)
        (carriesBag ?r ?o)
    )
)

(:action PULL-BAG
    :parameters (?r -robot ?o -takeable)
    :precondition (and
        (not (handFull ?r))
        (bagFull ?r)
        (carries ?r ?o)
    )
    :effect (and
        (handFull ?r)
```

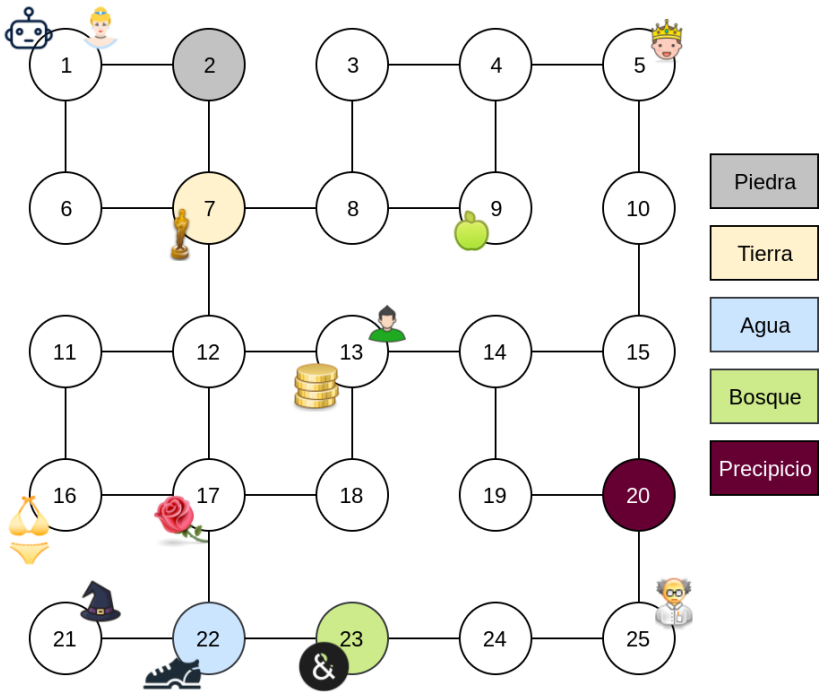
```
        (not (bagFull ?r))
        (not (carriesBag ?r ?o))
    )
)
```

3.3. Extender el problema

Añadimos los nuevos objetos y sus predicados, nótese que mientras que tanto agua como bosque como precipicio son especial el último no tiene objeto que permita recorrerlo. Por el contrario, arena y piedra tienen tipo pero no son especiales y no requieren por tanto de objeto para pasar por él.

```
(:objects
        ...
        bosque agua precipicio arena piedra - groundKind
    zapatilla bikini - wear
)
(:init
        ...
        ;#land conditions
    (needs agua bikini)
    (needs bosque zapatilla)

        ;#special locs
    (isSpecial loc22)
    (isKind loc22 agua)
    (isSpecial loc23)
    (isKind loc23 bosque)

    (isSpecial loc20)
    (isKind loc20 precipicio)

    ;#not so special locs
    (isKind loc7 arena)
    (isKind loc2 piedra)

    ;#wear position
    (at bikini loc16)
    (at zapatilla loc22)

    (at algoritmo loc23)
```

Quedando así el mapa



El único objetivo es que el profesor consiga un regalo, y es bonito ver la solución porque no puede tirar las zapatillas para coger el algoritmo, entonces tiene que coger el bikini, las zapatillas, volver a la casilla 21, cambiar el orden de los objetos soltar el bikini en la casilla 23, coger el algoritmo y dárselo al profesor. Veamos un fragmeno de la salida.

```
7: GOTO ROBOT1 OESTE LOC17 LOC16
8: TURN-LEFT ROBOT1 OESTE SUR
9: TURN-LEFT ROBOT1 SUR ESTE
10: PICK ROBOT1 LOC16 BIKINI
11: GOTO ROBOT1 ESTE LOC16 LOC17
12: PUSH-BAG ROBOT1 BIKINI
13: TURN-RIGHT ROBOT1 ESTE SUR
14: GOTO-SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
15: PICK ROBOT1 LOC22 ZAPATILLA
16: TURN-RIGHT ROBOT1 SUR OESTE
17: GOTO ROBOT1 OESTE LOC22 LOC21
18: TURN-RIGHT ROBOT1 OESTE NORTE
19: DROP ROBOT1 LOC21 ZAPATILLA
20: PULL-BAG ROBOT1 BIKINI
21: DROP ROBOT1 LOC21 BIKINI
22: TURN-RIGHT ROBOT1 NORTE ESTE
23: PICK ROBOT1 LOC21 ZAPATILLA
24: PUSH-BAG ROBOT1 ZAPATILLA
```

```
25: PICK ROBOT1 LOC21 BIKINI
26: GOTO−SPECIAL ROBOT1 ESTE LOC21 LOC22 AGUA BIKINI
27: GOTO−SPECIAL ROBOT1 ESTE LOC22 LOC23 BOSQUE
    ↪ ZAPATILLA
28: DROP−SPECIAL ROBOT1 LOC23 BIKINI BOSQUE
29: PICK ROBOT1 LOC23 ALGORITMO
30: GOTO ROBOT1 ESTE LOC23 LOC24
31: GOTO ROBOT1 ESTE LOC24 LOC25
32: GIVE ROBOT1 LOC25 ALGORITMO PROFESOR
```

4. Puntos por regalos

4.1. Modificar el dominio

Este cambio es más sencillo, sólo debemos añadir algunas funciones y modificar *GIVE*.

```
(:functions
    ...
    (pointsEarned ?r −robot)
    (pointsFor ?c −character ?g −gift)
)

(:action GIVE
    :parameters (?r −robot ?l −location ?g −gift ?c −character)
    :precondition (and
        (at ?r ?l)
        (at ?c ?l)
        (handFull ?r)
        (carries ?r ?g)
        (not (carriesBag ?r ?g))
    )
    :effect (and
        (not (handFull ?r))
        (not (carries ?r ?g))
        (hasGift ?c)
        (increase (pointsEarned ?r) (pointsFor ?c ?g))
    )
)
```

4.2. Modificar el problema

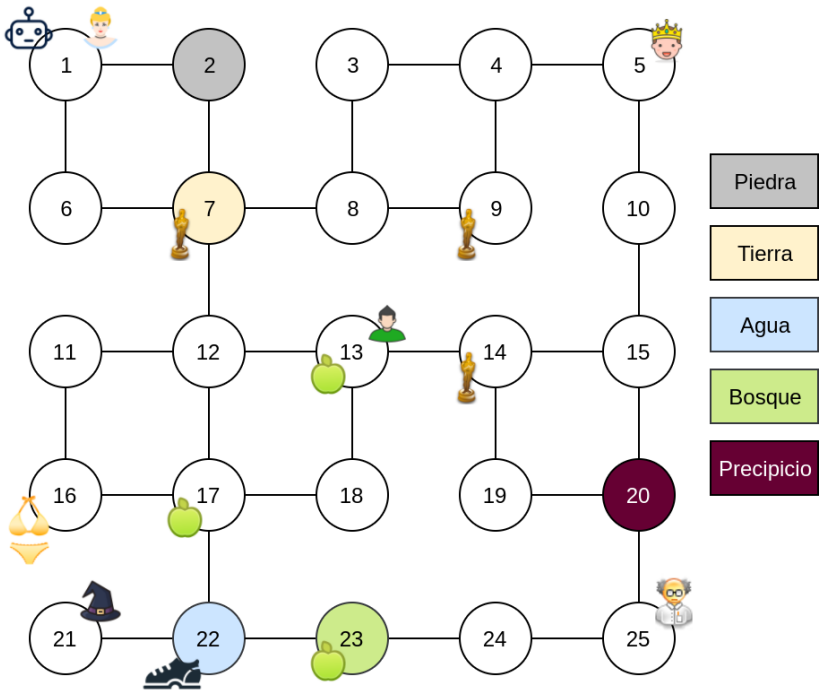
Añadimos la matriz de puntos al problema

```
(:init
```

```
    ...
    ;points For
    (= (pointsFor leonardo oscar) 10)
    (= (pointsFor leonardo rosa) 1)
    (= (pointsFor leonardo manzana) 3)
    (= (pointsFor leonardo algoritmo) 4)
    (= (pointsFor leonardo oro) 5)
    ...
```

Modificamos los regalos y los objetivos. Ahora buscamos que todos los personajes tengan al menos un regalo, y la combinación de puntos sea de al menos 40, el espacio solución se reduce y como lo tengo configurado para optimizar el número de instrucciones con $g = 1, h = 5$ ya tarda 17.4 segundos en encontrar una solución. Al final del documento dejo constancia de las opciones de ejecución de cada problema.

```
(:init
    ...
    ;gifts
    (at oscar loc7)
    (at oscar loc9)
    (at oscar loc14)
    (at manzana loc17)
    (at manzana loc23)
    (at manzana loc13)
    ...
)
(:goal
        (and
        (hasGift princesa)
        (hasGift principe)
        (hasGift bruja)
        (hasGift profesor)
        (hasGift leonardo)
        (> (pointsEarned robot1) 40)
    )
)
```



Finalmente llega a la conclusión de que el reparto que satisface la restricción es la manzana a la bruja, dos oscars a leonardo, el oscar a la princesa y sendas manzanas al profesor y al príncipe, sumando en total **44**.

5. Bolsillo mágico de los personajes

5.1. Modificar el dominio

Sólo tenemos que añadir una función que se inicialice con el número máximo de regalos para cada personaje y modificar *GIVE* para comprobar que queda espacio en el bolsillo y actualizarlo. En este problema conservo *hasGift*, para el ejercicio 7 ya lo elimino y trabajo únicamente con el número restante de regalos que caben.

```
(:functions ;todo: define numeric functions here
        ...
    (maxGifts ?c -character)
)

(:action GIVE
    :parameters (?r -robot ?l -location ?g -gift ?c -character)
    :precondition (and
        (at ?r ?l)
```

```
        (at ?c ?l)
        (handFull ?r)
        (carries ?r ?g)
        (not (carriesBag ?r ?g))
        (> (maxGifts ?c) 0)
    )
    :effect (and
        (not (handFull ?r))
        (not (carries ?r ?g))
        (hasGift ?c)
        (increase (pointsEarned ?r) (pointsFor ?c ?g))
        (decrease (maxGifts ?c) 1)
    )
)
```

5.2. Modificar el problema

En este caso retomo el problema anterior eliminando el requisito de tener un regalo el príncipe y prohibiendo la solución que nos daba el robot —Leonardo sólo puede coger un regalo—. Definimos el tamaño máximo del bolsillo y el nuevo objetivo. Este problema es suficientemente complejo como para evaluar correctamente el functionamiento del programa, para los dos siguientes ejercicios sí habrá dos problemas distintos para poder evaluarlo de forma exhaustiva.

```
(:init
        ...
        ;characters max
    (= (maxGifts princesa) 2)
    (= (maxGifts principe) 2)
    (= (maxGifts bruja) 2)
    (= (maxGifts profesor) 2)
    (= (maxGifts leonardo) 1)
        ...
)
(:goal (and
        (hasGift princesa)
        (hasGift bruja)
        (hasGift profesor)
        (hasGift leonardo)
        (>= (pointsEarned robot1) 45)
    )
)
```

Llega a la solución donde le da dos manzanas a la bruja, dos óscars a la princesa, otro óscar a Leonardo y la manzana restante al profesor, así consigue llegar a la única solución de exactamente **45** puntos.

6. Dos jugadores cooperantes

6.1. Modificar el dominio

Aquí sólo tenemos que añadir una nueva función para controlar el número total de puntos —*pointEarned* se queda como estaba—, así como modificar las acciones para actualizar este valor al entregar un regalo.

```
(:functions ;todo: define numeric functions here
    ...
    (pointsEarned ?r −robot)
    (totalPoints)
    ...
)

(:action GIVE
    :parameters (?r −robot ?l −location ?g −gift ?c −character)
    :precondition (and
        (at ?r ?l)
        (at ?c ?l)
        (handFull ?r)
        (carries ?r ?g)
        (not (carriesBag ?r ?g))
        (> (maxGifts ?c) 0)
    )
    :effect (and
        (not (handFull ?r))
        (not (carries ?r ?g))
        ;(hasGift ?c)
        (increase (pointsEarned ?r) (pointsFor ?c ?g))
        (increase (totalPoints) (pointsFor ?c ?g))
        (decrease (maxGifts ?c) 1)
    )
)
```

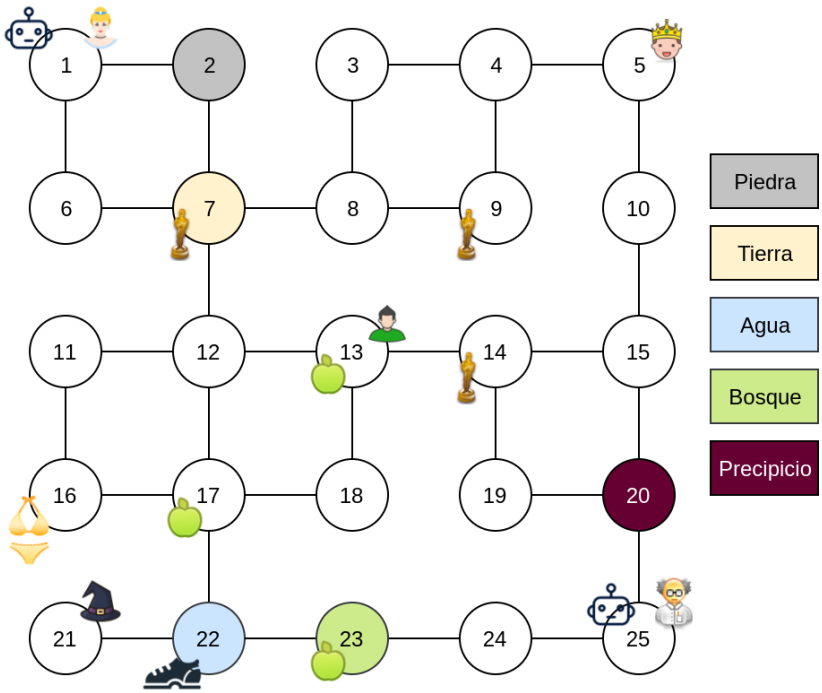
6.2. Modificar el problema (1)

Para el problema, inicializamos el contador total a 0, añadimos un nuevo robot y definimos los puntos mínimos para cada uno.

```
(:objects
    ...
    robot1 robot2 − robot
    ...
)
(:init
```

```
        ...
        ;#points
    (= (totalPoints) 0)
    ;#robot1
    (at robot1 loc1)
    (facing robot1 norte)
    (= (pathLength robot1) 0)
    (= (pointsEarned robot1) 0)
    ;#robot2
    (at robot2 loc25)
    (facing robot2 norte)
    (= (pathLength robot2) 0)
    (= (pointsEarned robot2) 0)
)
(:goal (and
        (>= (pointsEarned robot1) 10)
        (>= (pointsEarned robot2) 10)
    )
)
```

Nos basamos en el problema anterior pero añadiendo un plus de dificultad, y es que para poder satisfacer el robot dos su cometido necesita la colaboración del primero, pues tiene que llevarle las zapatillas para poder coger el regalo del bosque, o las zapatillas y el bikini para salir de su islote y satisfacer su objetivo.



Efectivamente así lo hace, veamos un fragmento de la salida.

```
25: GOTO_SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
26: TURN_LEFT ROBOT1 SUR ESTE
27: PICK ROBOT1 LOC22 ZAPATILLA
28: GOTO_SPECIAL ROBOT1 ESTE LOC22 LOC23 BOSQUE
    ↪ ZAPATILLA
29: GOTO ROBOT1 ESTE LOC23 LOC24
30: DROP ROBOT1 LOC24 ZAPATILLA
31: GOTO ROBOT2 OESTE LOC25 LOC24
32: PICK ROBOT2 LOC24 ZAPATILLA
33: PUSH_BAG ROBOT2 ZAPATILLA
34: GOTO_SPECIAL ROBOT2 OESTE LOC24 LOC23 BOSQUE
    ↪ ZAPATILLA
35: TURN_LEFT ROBOT2 OESTE SUR
36: TURN_LEFT ROBOT2 SUR ESTE
37: PICK ROBOT2 LOC23 MANZANA
38: GOTO ROBOT2 ESTE LOC23 LOC24
39: GOTO ROBOT2 ESTE LOC24 LOC25
40: GIVE ROBOT2 LOC25 MANZANA PROFESOR
41: PULL_BAG ROBOT1 BIKINI
42: TURN_RIGHT ROBOT2 ESTE SUR
43: TURN_RIGHT ROBOT2 SUR OESTE
44: GOTO ROBOT2 OESTE LOC25 LOC24
```

```
45: DROP ROBOT1 LOC24 BIKINI
46: PICK ROBOT2 LOC24 BIKINI
47: GOTO-SPECIAL ROBOT2 OESTE LOC24 LOC23 BOSQUE
    ↪ ZAPATILLA
48: GOTO-SPECIAL ROBOT2 OESTE LOC23 LOC22 AGUA BIKINI
49: TURN-RIGHT ROBOT2 OESTE NORTE
50: GOTO ROBOT2 NORTE LOC22 LOC17
51: GOTO ROBOT2 NORTE LOC17 LOC12
52: TURN-RIGHT ROBOT2 NORTE ESTE
53: DROP ROBOT2 LOC12 BIKINI
54: GOTO ROBOT2 ESTE LOC12 LOC13
55: GOTO ROBOT2 ESTE LOC13 LOC14
56: TURN-RIGHT ROBOT2 ESTE SUR
57: TURN-RIGHT ROBOT2 SUR OESTE
58: PICK ROBOT2 LOC14 OSCAR
59: GOTO ROBOT2 OESTE LOC14 LOC13
60: GIVE ROBOT2 LOC13 OSCAR LEONARDO
```

Tras conseguir ROBOT1 sus 10 puntos le lleva las zapatillas y el bikini a ROBOT2, primero le da las zapatillas, coge la manzana, se la da al profesor y después le da el ROBOT1 el bikini para poder salir de la isla y conseguir otros 10 puntos dándole el óscar a leonardo. Es prácticamente mágico.

6.3. Modificar el problema (2)

En este caso vamos a repetir un problema muy parecido al anterior pero cambiando los términos del objetivo.

```
(:goal (and
        (>= (pointsEarned robot1) 5)
        (<= (pointsEarned robot1) 10)
        (>= (totalPoints) 15)
    )
)
```

Aquí podemos ver un situación muy interesante, aunque la solución del problema anterior —que tardó 0.3 segundos en obtenerla— satisface los requisitos de este objetivo, para este problema tarda 45 minutos porque no es tan fácil ver que tiene que llevarle las zapatillas y el bikini el ROBOT1 al ROBOT2 para poder satisfacer los objetivos. Por el contrario, si sólo pongo como objetivo *(>= (totalPoints) 15)* lo resuelve de forma inmediata.

7. Robots diferenciados

7.1. Corrigiendo los errores anteriores

Aquí ya están solucionados todos los problemas que hemos ido comentando en ejercicios anteriores. He eliminado el carriesHand deducido por uno explícitamente definido y en las acciones que lo requieran se comprueba si está en alguno de los dos sitios.

```
(:predicates ;todo: define predicates here
    ...
    (carriesHand ?r -robot ?o -takeable)
    (carriesBag ?r -robot ?o -takeable)
    (handFull ?r -robot)
    (bagFull ?r -robot)
        ...
)

(:action GOTO-SPECIAL
    :parameters (?r - robot ?o -orientation ?l1 -location ?l2 -location ?gk
        ↪ -groundKind ?w -wear)
    :precondition (and
        (at ?r ?l1)
        (facing ?r ?o)
        (link ?l1 ?l2 ?o)
        (isSpecial ?l2)
        (isKind ?l2 ?gk)
        (needs ?gk ?w)
        (or
            (carriesHand ?r ?w)
            (carriesBag ?r ?w)
        )
    )
    :effect (and
        (not (at ?r ?l1))
        (at ?r ?l2)
        (increase (pathLength ?r) (linkLength ?l1 ?l2))
    )
)
```

Como ya avanzábamos hemos modificado la lógica del drop para permitir soltar correctamente *takeables* en cualquier zona dependiendo de la legalidad de la acción y conservar los mensajes de salida significativos.

```
(:action DROP
    :parameters (?r -robot ?l -location ?g -takeable)
    :precondition (and
```

```
        (handFull ?r)
        (carriesHand ?r ?g)
        (at ?r ?l)
        (not (isSpecial ?l))
    )
    :effect (and
        (not (handFull ?r))
        (not (carriesHand ?r ?g))
        (at ?g ?l)
    )
)

(:action DROP-SPECIAL-GIFT
    :parameters (?r -robot ?l -location ?g -gift ?gk -groundKind)
    :precondition (and
        (handFull ?r)
        (carriesHand ?r ?g)
        (at ?r ?l)
        (isSpecial ?l)
    )
    :effect (and
        (not (handFull ?r))
        (not (carriesHand ?r ?g))
        (at ?g ?l)
    )
)

(:action DROP-SPECIAL-WEAR
    :parameters (?r -robot ?l -location ?g -wear ?gk -groundKind)
    :precondition (and
        (handFull ?r)
        (carriesHand ?r ?g)
        (at ?r ?l)
        (isKind ?l ?gk)
        (or
            (carriesBag ?r ?g)
            (not (needs ?gk ?g))
        )
    )
    :effect (and
        (not (handFull ?r))
        (not (carriesHand ?r ?g))
        (at ?g ?l)
    )
)
```

7.2. Modificar el modelo

Ahora, para añadir al modelo la lógica de los robots con capacidades diferenciadas añadimos dos nuevos subtipos de robot, *carrier* y *giver*, el primero coge objetos del suelo y se los da a *giver*, mientras que éste último sólo puede interactuar con humanos.

```
(:types
        carrier giver − robot
    robot character takeable − locatable
)
```

Ahora tenemos que modificar *PICK y GIVE*, y añadir un nuevo método, *GIVE-COOP*. No modificamos ninguno de los tres *DROP*, ambos robots pueden tirar algo al suelo, pero solo *carrier* puede recogerlo.

```
(:action PICK
    :parameters (?r − carrier ?l −location ?g −takeable)
    :precondition (and
        (not (handFull ?r))
        (at ?r ?l)
        (at ?g ?l)
    )
    :effect (and
        (handFull ?r)
        (not (at ?g ?l))
        (carriesHand ?r ?g)
    )
)
(:action GIVE
    :parameters (?r −giver ?l −location ?g −gift ?c −character)
    :precondition (and
        (at ?r ?l)
        (at ?c ?l)
        (handFull ?r)
        (carriesHand ?r ?g)
        (> (maxGifts ?c) 0)
    )
    :effect (and
        (not (handFull ?r))
        (not (carriesHand ?r ?g))
        (increase (pointsEarned ?r) (pointsFor ?c ?g))
        (decrease (maxGifts ?c) 1)
    )
)

(:action GIVE−COOP
```

```
    :parameters (?c −carrier ?g −giver ?l −location ?t −takeable)
    :precondition (and
        (at ?c ?l)
        (at ?g ?l)
        (handFull ?c)
        (carriesHand ?c ?t)
        (not (handFull ?g))
    )
    :effect (and
        (not (handFull ?c))
        (not (carriesHand ?c ?t))
        (handFull ?g)
        (carriesHand ?g ?t)
    )
)
```

7.3. Modificar el problema (1)

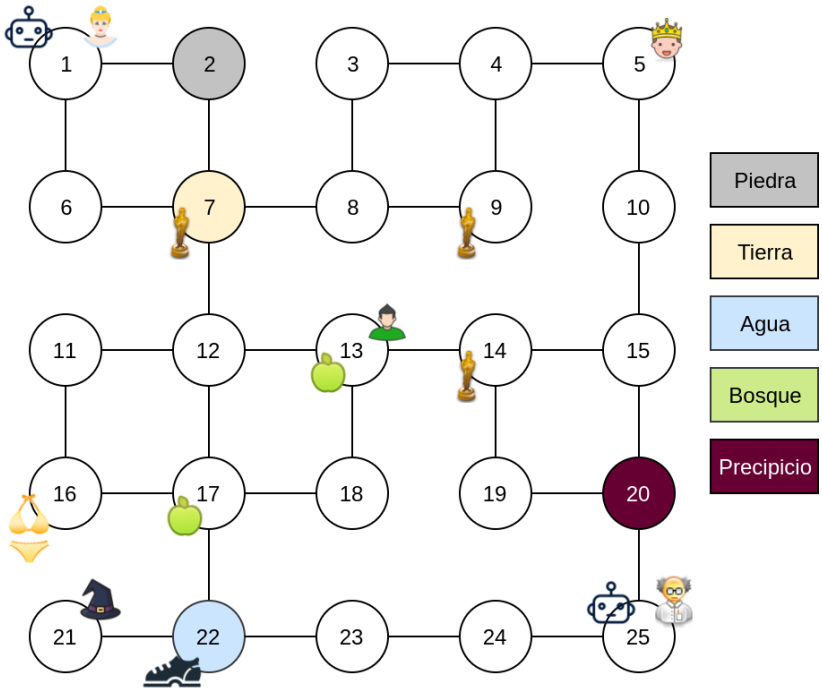
Para el correcto funcionamiento sólo tenemos que definir ROBOT1 y ROBOT2 como tipos diferenciados.

```
(:objects
    robot1 − carrier
    robot2 − giver
)
```

Para este problema, ya que ROBOT1 no puede recoger ni las zapatillas ni el bikini del suelo para poder salir de la isla, eliminamos el bosque y planteamos como objetivo que ROBOT2 le de un regalo a la bruja, es prácticamente como el problema anterior pero ahora el intercambio del bikini y las zapatillas debe ser con *GIVE-COOP*.

```
(:init
        ;#special locs
    (isSpecial loc22)
    (isKind loc22 agua)

    (isSpecial loc20)
    (isKind loc20 precipicio)

    ;#not so special locs
    (isKind loc7 arena)
    (isKind loc2 piedra)

    ...
```

```
    ;#gifts
    (at oscar loc7)
    (at oscar loc9)
    (at oscar loc14)
    (at manzana loc17)
    (at manzana loc13)

        ...
)
(:goal
    (and
        (= (maxGifts bruja) 1)
    )
)
```



Veamos un fragmento de la salida.

```
12: GOTO ROBOT1 OESTE LOC17 LOC16
13: TURN-LEFT ROBOT1 OESTE SUR
14: TURN-LEFT ROBOT1 SUR ESTE
15: PICK ROBOT1 LOC16 BIKINI
16: GOTO ROBOT1 ESTE LOC16 LOC17
17: PUSH-BAG ROBOT1 BIKINI
18: PICK ROBOT1 LOC17 MANZANA
```

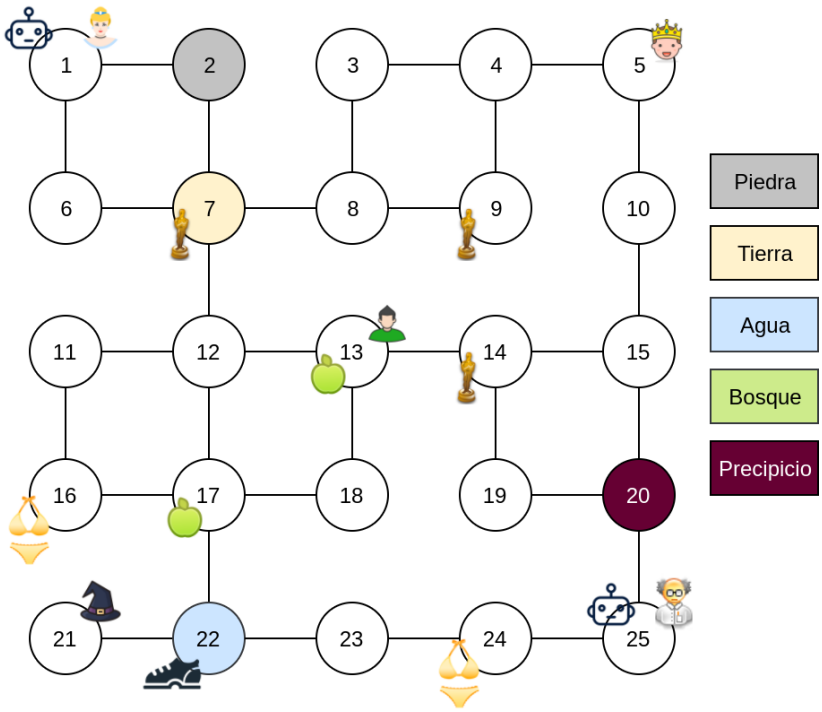
```
19: TURN−RIGHT ROBOT1 ESTE SUR
20: GOTO−SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
21: TURN−LEFT ROBOT1 SUR ESTE
22: GOTO ROBOT1 ESTE LOC22 LOC23
23: GIVE−COOP ROBOT1 ROBOT2 LOC23 BIKINI
24: PULL−BAG ROBOT1 BIKINI
25: PUSH−BAG ROBOT2 BIKINI
26: GIVE−COOP ROBOT1 ROBOT2 LOC23 MANZANA
27: GOTO−SPECIAL ROBOT2 OESTE LOC23 LOC22 AGUA BIKINI
28: GOTO ROBOT2 OESTE LOC22 LOC21
29: GIVE ROBOT2 LOC21 MANZANA BRUJA
```

Efectivamente ROBOT1 coge la manza y el bikini y se lo lleva a ROBOT2, y éste último le entrega la manzana a la bruja.

7.4. Modificar el problema (2)

En esta versión, sobre el mapa anterior, en las mismas condiciones cambiamos el objetivo para que todos los personajes reciban exactamente 1 regalo. Tengo que añadir otro bikini dentro del islote para que ROBOT1 pueda salir con ROBOT2 y ayudarle a entregar los regalos.

```
(:init
        ....
        ;wears
    (at bikini loc16)
    (at bikini loc24)
    (at zapatilla loc22)
    ...
)
```



Efectivamente traza el plan de 124 pasos que adjunto al final.

7.5. Modificar el problema (3)

Finalmente, y para regodearnos en nuestro éxito vamos a definir un problema combinando varias cosas de las ya implementadas.

```
(:init
        ...
        ;gifts
    (at oscar loc7)
    (at manzana loc9)
    (at rosa loc17)
    (at algoritmo loc23)
    (at oro loc13)
    ...
)

(:goal (and
        (= (maxGifts princesa) 1)
        (= (maxGifts principe) 1)
        (= (maxGifts bruja) 1)
        (= (maxGifts profesor) 1)
        (= (maxGifts leonardo) 1)
```

```
        (= (totalPoints) 50)
    )
)
```

No ha sido precisamente rápido pero lo ha obtenido.

8. Salidas

1.1

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is best-first on 1*g(s) + 5*h(s) where
    metric is plan length

advancing to distance: 23
                       22
                       20
                       19
                       18
                       17
                       16
                       14
                       13
                       12
                       11
                       10
                        9
                        8
                        7
                        6
                        5
                        4
                        3
                        2
```

```
                    1
                    0

ff: found legal plan as follows

step 0: PICK ROBOT1 LOC1 PRINCESA
        1: TURN-RIGHT ROBOT1 NORTE ESTE
        2: GOTO ROBOT1 ESTE LOC1 LOC2
        3: TURN-RIGHT ROBOT1 ESTE SUR
        4: GOTO ROBOT1 SUR LOC2 LOC7
        5: GOTO ROBOT1 SUR LOC7 LOC12
        6: TURN-LEFT ROBOT1 SUR ESTE
        7: GOTO ROBOT1 ESTE LOC12 LOC13
        8: TURN-RIGHT ROBOT1 ESTE SUR
        9: DROP ROBOT1 LOC13 PRINCESA
       10: PICK ROBOT1 LOC13 ORO
       11: GIVE ROBOT1 LOC13 PRINCESA PRINCESA
       12: PICK ROBOT1 LOC13 PRINCESA
       13: GIVE ROBOT1 LOC13 ORO LEONARDO
       14: PICK ROBOT1 LOC13 LEONARDO
       15: TURN-LEFT ROBOT1 SUR ESTE
       16: GOTO ROBOT1 ESTE LOC13 LOC14
       17: GOTO ROBOT1 ESTE LOC14 LOC15
       18: TURN-RIGHT ROBOT1 ESTE SUR
       19: GOTO ROBOT1 SUR LOC15 LOC20
       20: GOTO ROBOT1 SUR LOC20 LOC25
       21: TURN-RIGHT ROBOT1 SUR OESTE
       22: GIVE ROBOT1 LOC25 ORO PROFESOR
       23: PICK ROBOT1 LOC25 PROFESOR
       24: TURN-RIGHT ROBOT1 OESTE NORTE
       25: GOTO ROBOT1 NORTE LOC25 LOC20
       26: GOTO ROBOT1 NORTE LOC20 LOC15
       27: GOTO ROBOT1 NORTE LOC15 LOC10
       28: GOTO ROBOT1 NORTE LOC10 LOC5
       29: TURN-LEFT ROBOT1 NORTE OESTE
       30: GIVE ROBOT1 LOC5 PROFESOR PRINCIPE
       31: PICK ROBOT1 LOC5 PRINCIPE
       32: GOTO ROBOT1 OESTE LOC5 LOC4
       33: GOTO ROBOT1 OESTE LOC4 LOC3
       34: TURN-LEFT ROBOT1 OESTE SUR
       35: GOTO ROBOT1 SUR LOC3 LOC8
       36: TURN-RIGHT ROBOT1 SUR OESTE
       37: GOTO ROBOT1 OESTE LOC8 LOC7
       38: TURN-LEFT ROBOT1 OESTE SUR
       39: GOTO ROBOT1 SUR LOC7 LOC12
       40: GOTO ROBOT1 SUR LOC12 LOC17
```

```
        41: GOTO ROBOT1 SUR LOC17 LOC22
        42: TURN-RIGHT ROBOT1 SUR OESTE
        43: GOTO ROBOT1 OESTE LOC22 LOC21
        44: GIVE ROBOT1 LOC21 PROFESOR BRUJA


time spent: 0.00 seconds instantiating 1820 easy, 0 hard action templates
                    0.00 seconds reachability analysis, yielding 296 facts and 1820
                        ↪ actions
                    0.00 seconds creating final representation with 296 relevant
                        ↪ facts, 0 relevant fluents
                    0.00 seconds computing LNF
                    0.00 seconds building connectivity graph
                    0.02 seconds searching, evaluating 367 states, to a max depth
                        ↪ of 0
                    0.02 seconds total time
```

2.1

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


metric established (normalized to minimize): ((1.00*[RF0](
    ↪ PATHLENGTH_ROBOT1)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is best-first on 1*g(s) + 2*h(s) where
    metric is ((1.00*[RF0](PATHLENGTH_ROBOT1)) - () + 0.00)

advancing to distance: 23
                       22
                       21
                       19
                       18
                       17
                       16
                       15
                       14
```

```
                13
                12
                11
                10
                 9
                 8
                 7
                 6
                 5
                 4
                 3
                 2
                 1
                 0

ff: found legal plan as follows

step 0: TURN-RIGHT ROBOT1 NORTE ESTE
        1: TURN-RIGHT ROBOT1 ESTE SUR
        2: GOTO ROBOT1 SUR LOC1 LOC6
        3: TURN-LEFT ROBOT1 SUR ESTE
        4: GOTO ROBOT1 ESTE LOC6 LOC7
        5: PICK ROBOT1 LOC7 OSCAR1
        6: TURN-LEFT ROBOT1 ESTE NORTE
        7: TURN-LEFT ROBOT1 NORTE OESTE
        8: GOTO ROBOT1 OESTE LOC7 LOC6
        9: TURN-RIGHT ROBOT1 OESTE NORTE
       10: GOTO ROBOT1 NORTE LOC6 LOC1
       11: GIVE ROBOT1 LOC1 OSCAR1 PRINCESA1
       12: TURN-RIGHT ROBOT1 NORTE ESTE
       13: GOTO ROBOT1 ESTE LOC1 LOC2
       14: TURN-RIGHT ROBOT1 ESTE SUR
       15: GOTO ROBOT1 SUR LOC2 LOC7
       16: GOTO ROBOT1 SUR LOC7 LOC12
       17: GOTO ROBOT1 SUR LOC12 LOC17
       18: PICK ROBOT1 LOC17 ROSA1
       19: GOTO ROBOT1 SUR LOC17 LOC22
       20: TURN-RIGHT ROBOT1 SUR OESTE
       21: GOTO ROBOT1 OESTE LOC22 LOC21
       22: GIVE ROBOT1 LOC21 ROSA1 BRUJA1
       23: TURN-RIGHT ROBOT1 OESTE NORTE
       24: TURN-RIGHT ROBOT1 NORTE ESTE
       25: GOTO ROBOT1 ESTE LOC21 LOC22
       26: TURN-LEFT ROBOT1 ESTE NORTE
       27: GOTO ROBOT1 NORTE LOC22 LOC17
       28: TURN-RIGHT ROBOT1 NORTE ESTE
```

29: GOTO ROBOT1 ESTE LOC17 LOC18
30: TURN−LEFT ROBOT1 ESTE NORTE
31: GOTO ROBOT1 NORTE LOC18 LOC13
32: TURN−LEFT ROBOT1 NORTE OESTE
33: PICK ROBOT1 LOC13 ORO1
34: GIVE ROBOT1 LOC13 ROSA1 LEONARDO1
35: TURN−RIGHT ROBOT1 OESTE NORTE
36: TURN−RIGHT ROBOT1 NORTE ESTE
37: GOTO ROBOT1 ESTE LOC13 LOC14
38: TURN−RIGHT ROBOT1 ESTE SUR
39: GOTO ROBOT1 SUR LOC14 LOC19
40: PICK ROBOT1 LOC19 ALGORITMO1
41: TURN−LEFT ROBOT1 SUR ESTE
42: GOTO ROBOT1 ESTE LOC19 LOC20
43: TURN−RIGHT ROBOT1 ESTE SUR
44: GOTO ROBOT1 SUR LOC20 LOC25
45: DROP ROBOT1 LOC25 OSCAR1
46: TURN−LEFT ROBOT1 SUR ESTE
47: TURN−LEFT ROBOT1 ESTE NORTE
48: PICK ROBOT1 LOC25 OSCAR1
49: GIVE ROBOT1 LOC25 ORO1 PROFESOR1
50: GOTO ROBOT1 NORTE LOC25 LOC20
51: GOTO ROBOT1 NORTE LOC20 LOC15
52: GOTO ROBOT1 NORTE LOC15 LOC10
53: GOTO ROBOT1 NORTE LOC10 LOC5
54: TURN−LEFT ROBOT1 NORTE OESTE
55: GOTO ROBOT1 OESTE LOC5 LOC4
56: TURN−LEFT ROBOT1 OESTE SUR
57: GOTO ROBOT1 SUR LOC4 LOC9
58: PICK ROBOT1 LOC9 MANZANA1
59: DROP ROBOT1 LOC9 MANZANA1
60: TURN−LEFT ROBOT1 SUR ESTE
61: TURN−LEFT ROBOT1 ESTE NORTE
62: PICK ROBOT1 LOC9 MANZANA1
63: GOTO ROBOT1 NORTE LOC9 LOC4
64: TURN−RIGHT ROBOT1 NORTE ESTE
65: GOTO ROBOT1 ESTE LOC4 LOC5
66: DROP ROBOT1 LOC5 OSCAR1
67: TURN−LEFT ROBOT1 ESTE NORTE
68: TURN−LEFT ROBOT1 NORTE OESTE
69: PICK ROBOT1 LOC5 OSCAR1
70: GIVE ROBOT1 LOC5 ALGORITMO1 PRINCIPE1

time spent: 0.00 seconds instantiating 945 easy, 0 hard **action** templates

```
0.00 seconds reachability analysis, yielding 171 facts and 345
     ↪ actions
0.00 seconds creating final representation with 166 relevant
     ↪ facts, 1 relevant fluents
0.00 seconds computing LNF
0.00 seconds building connectivity graph
1.72 seconds searching, evaluating 51090 states, to a max
     ↪ depth of 0
1.72 seconds total time
```

2.2

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


metric established (normalized to minimize): ((1.00*[RF0](
     ↪ PATHLENGTH_ROBOT1)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is best-first on 1*g(s) + 2*h(s) where
     metric is ((1.00*[RF0](PATHLENGTH_ROBOT1)) - () + 0.00)

advancing to distance: 23
                       22
                       21
                       19
                       18
                       17
                       16
                       15
                       14
                       13
                       12
                       11
                       10
                        9
                        8
                        7
                        6
```

```
                    5
                    4
                    3
                    2
                    1
                    0

ff: found legal plan as follows

step 0: TURN-RIGHT ROBOT1 NORTE ESTE
        1: GOTO ROBOT1 ESTE LOC1 LOC2
        2: TURN-RIGHT ROBOT1 ESTE SUR
        3: GOTO ROBOT1 SUR LOC2 LOC7
        4: TURN-LEFT ROBOT1 SUR ESTE
        5: TURN-LEFT ROBOT1 ESTE NORTE
        6: PICK ROBOT1 LOC7 OSCAR1
        7: TURN-LEFT ROBOT1 NORTE OESTE
        8: GOTO ROBOT1 OESTE LOC7 LOC6
        9: TURN-RIGHT ROBOT1 OESTE NORTE
       10: GOTO ROBOT1 NORTE LOC6 LOC1
       11: GIVE ROBOT1 LOC1 OSCAR1 PRINCESA1
       12: TURN-RIGHT ROBOT1 NORTE ESTE
       13: GOTO ROBOT1 ESTE LOC1 LOC2
       14: TURN-RIGHT ROBOT1 ESTE SUR
       15: GOTO ROBOT1 SUR LOC2 LOC7
       16: GOTO ROBOT1 SUR LOC7 LOC12
       17: GOTO ROBOT1 SUR LOC12 LOC17
       18: PICK ROBOT1 LOC17 ROSA1
       19: GOTO ROBOT1 SUR LOC17 LOC22
       20: TURN-RIGHT ROBOT1 SUR OESTE
       21: GOTO ROBOT1 OESTE LOC22 LOC21
       22: GIVE ROBOT1 LOC21 ROSA1 BRUJA1
       23: TURN-RIGHT ROBOT1 OESTE NORTE
       24: TURN-RIGHT ROBOT1 NORTE ESTE
       25: GOTO ROBOT1 ESTE LOC21 LOC22
       26: TURN-LEFT ROBOT1 ESTE NORTE
       27: GOTO ROBOT1 NORTE LOC22 LOC17
       28: TURN-RIGHT ROBOT1 NORTE ESTE
       29: GOTO ROBOT1 ESTE LOC17 LOC18
       30: TURN-LEFT ROBOT1 ESTE NORTE
       31: GOTO ROBOT1 NORTE LOC18 LOC13
       32: TURN-LEFT ROBOT1 NORTE OESTE
       33: PICK ROBOT1 LOC13 ORO1
       34: GIVE ROBOT1 LOC13 ROSA1 LEONARDO1
       35: TURN-RIGHT ROBOT1 OESTE NORTE
       36: TURN-RIGHT ROBOT1 NORTE ESTE
```

```
37: GOTO ROBOT1 ESTE LOC13 LOC14
38: TURN-RIGHT ROBOT1 ESTE SUR
39: GOTO ROBOT1 SUR LOC14 LOC19
40: PICK ROBOT1 LOC19 ALGORITMO1
41: TURN-LEFT ROBOT1 SUR ESTE
42: GOTO ROBOT1 ESTE LOC19 LOC20
43: TURN-RIGHT ROBOT1 ESTE SUR
44: GOTO ROBOT1 SUR LOC20 LOC25
45: DROP ROBOT1 LOC25 OSCAR1
46: TURN-LEFT ROBOT1 SUR ESTE
47: TURN-LEFT ROBOT1 ESTE NORTE
48: PICK ROBOT1 LOC25 OSCAR1
49: GIVE ROBOT1 LOC25 ORO1 PROFESOR1
50: GOTO ROBOT1 NORTE LOC25 LOC20
51: GOTO ROBOT1 NORTE LOC20 LOC15
52: GOTO ROBOT1 NORTE LOC15 LOC10
53: GOTO ROBOT1 NORTE LOC10 LOC5
54: TURN-LEFT ROBOT1 NORTE OESTE
55: GOTO ROBOT1 OESTE LOC5 LOC4
56: TURN-LEFT ROBOT1 OESTE SUR
57: GOTO ROBOT1 SUR LOC4 LOC9
58: PICK ROBOT1 LOC9 MANZANA1
59: DROP ROBOT1 LOC9 MANZANA1
60: TURN-LEFT ROBOT1 SUR ESTE
61: TURN-LEFT ROBOT1 ESTE NORTE
62: PICK ROBOT1 LOC9 MANZANA1
63: GOTO ROBOT1 NORTE LOC9 LOC4
64: TURN-RIGHT ROBOT1 NORTE ESTE
65: GOTO ROBOT1 ESTE LOC4 LOC5
66: DROP ROBOT1 LOC5 OSCAR1
67: TURN-LEFT ROBOT1 ESTE NORTE
68: TURN-LEFT ROBOT1 NORTE OESTE
69: PICK ROBOT1 LOC5 OSCAR1
70: GIVE ROBOT1 LOC5 ALGORITMO1 PRINCIPE1


time spent: 0.00 seconds instantiating 945 easy, 0 hard action templates
            0.00 seconds reachability analysis, yielding 171 facts and 345
                 ↪ actions
            0.00 seconds creating final representation with 166 relevant
                 ↪ facts, 1 relevant fluents
            0.00 seconds computing LNF
            0.00 seconds building connectivity graph
            1.66 seconds searching, evaluating 51516 states, to a max
                 ↪ depth of 0
            1.66 seconds total time
```

3.1

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is best-first on 1*g(s) + 2*h(s) where
    metric is plan length

advancing to distance: 17
                            16
                            15
                            14
                            12
                            11
                            10
                             9
                             8
                             7
                             6
                             5
                             4
                             3
                             2
                             1
                             0

ff: found legal plan as follows

step 0: TURN-RIGHT ROBOT1 NORTE ESTE
        1: GOTO ROBOT1 ESTE LOC1 LOC2
        2: TURN-RIGHT ROBOT1 ESTE SUR
        3: GOTO ROBOT1 SUR LOC2 LOC7
        4: GOTO ROBOT1 SUR LOC7 LOC12
        5: GOTO ROBOT1 SUR LOC12 LOC17
```

```
 6: TURN-RIGHT ROBOT1 SUR OESTE
 7: GOTO ROBOT1 OESTE LOC17 LOC16
 8: TURN-LEFT ROBOT1 OESTE SUR
 9: TURN-LEFT ROBOT1 SUR ESTE
10: PICK ROBOT1 LOC16 BIKINI
11: GOTO ROBOT1 ESTE LOC16 LOC17
12: PUSH-BAG ROBOT1 BIKINI
13: TURN-RIGHT ROBOT1 ESTE SUR
14: GOTO-SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
15: PICK ROBOT1 LOC22 ZAPATILLA
16: TURN-RIGHT ROBOT1 SUR OESTE
17: GOTO ROBOT1 OESTE LOC22 LOC21
18: TURN-RIGHT ROBOT1 OESTE NORTE
19: DROP ROBOT1 LOC21 ZAPATILLA
20: PULL-BAG ROBOT1 BIKINI
21: DROP ROBOT1 LOC21 BIKINI
22: TURN-RIGHT ROBOT1 NORTE ESTE
23: PICK ROBOT1 LOC21 ZAPATILLA
24: PUSH-BAG ROBOT1 ZAPATILLA
25: PICK ROBOT1 LOC21 BIKINI
26: GOTO-SPECIAL ROBOT1 ESTE LOC21 LOC22 AGUA BIKINI
27: GOTO-SPECIAL ROBOT1 ESTE LOC22 LOC23 BOSQUE
    ↪ ZAPATILLA
28: DROP-SPECIAL ROBOT1 LOC23 BIKINI BOSQUE
29: PICK ROBOT1 LOC23 ALGORITMO
30: GOTO ROBOT1 ESTE LOC23 LOC24
31: GOTO ROBOT1 ESTE LOC24 LOC25
32: GIVE ROBOT1 LOC25 ALGORITMO PROFESOR


time spent: 0.00 seconds instantiating 1288 easy, 0 hard action templates
            0.00 seconds reachability analysis, yielding 340 facts and 1175
                 ↪ actions
            0.00 seconds creating final representation with 340 relevant
                 ↪ facts, 1 relevant fluents
            0.00 seconds computing LNF
            0.00 seconds building connectivity graph
            1.05 seconds searching, evaluating 14709 states, to a max
                 ↪ depth of 0
            1.05 seconds total time
```

4.1

```
ff: parsing domain file
domain 'BELKAN' defined
```

```
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is best-first on 1*g(s) + 5*h(s) where
     metric is plan length

advancing to distance: 30
                             29
                             28
                             27
                             24
                             23
                             22
                             21
                             20
                             19
                             18
                             17
                             16
                             15
                             14
                             13
                             11
                             10
                              9
                              5
                              4
                              3
                              2
                              1
                              0

ff: found legal plan as follows

step 0: TURN-RIGHT ROBOT1 NORTE ESTE
        1: GOTO ROBOT1 ESTE LOC1 LOC2
        2: TURN-RIGHT ROBOT1 ESTE SUR
        3: GOTO ROBOT1 SUR LOC2 LOC7
        4: GOTO ROBOT1 SUR LOC7 LOC12
```

5: TURN−RIGHT ROBOT1 SUR OESTE
6: GOTO ROBOT1 OESTE LOC12 LOC11
7: TURN−LEFT ROBOT1 OESTE SUR
8: GOTO ROBOT1 SUR LOC11 LOC16
9: PICK ROBOT1 LOC16 BIKINI
10: PUSH−BAG ROBOT1 BIKINI
11: TURN−LEFT ROBOT1 SUR ESTE
12: GOTO ROBOT1 ESTE LOC16 LOC17
13: TURN−RIGHT ROBOT1 ESTE SUR
14: PICK ROBOT1 LOC17 MANZANA
15: GOTO−SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
16: TURN−RIGHT ROBOT1 SUR OESTE
17: GOTO ROBOT1 OESTE LOC22 LOC21
18: TURN−RIGHT ROBOT1 OESTE NORTE
19: TURN−RIGHT ROBOT1 NORTE ESTE
20: GIVE ROBOT1 LOC21 MANZANA BRUJA
21: GOTO−SPECIAL ROBOT1 ESTE LOC21 LOC22 AGUA BIKINI
22: TURN−LEFT ROBOT1 ESTE NORTE
23: GOTO ROBOT1 NORTE LOC22 LOC17
24: GOTO ROBOT1 NORTE LOC17 LOC12
25: GOTO ROBOT1 NORTE LOC12 LOC7
26: PICK ROBOT1 LOC7 OSCAR
27: TURN−RIGHT ROBOT1 NORTE ESTE
28: TURN−RIGHT ROBOT1 ESTE SUR
29: GOTO ROBOT1 SUR LOC7 LOC12
30: TURN−LEFT ROBOT1 SUR ESTE
31: GOTO ROBOT1 ESTE LOC12 LOC13
32: GIVE ROBOT1 LOC13 OSCAR LEONARDO
33: GOTO ROBOT1 ESTE LOC13 LOC14
34: PICK ROBOT1 LOC14 OSCAR
35: TURN−RIGHT ROBOT1 ESTE SUR
36: TURN−RIGHT ROBOT1 SUR OESTE
37: GOTO ROBOT1 OESTE LOC14 LOC13
38: GIVE ROBOT1 LOC13 OSCAR LEONARDO
39: PICK ROBOT1 LOC13 MANZANA
40: GOTO ROBOT1 OESTE LOC13 LOC12
41: TURN−RIGHT ROBOT1 OESTE NORTE
42: GOTO ROBOT1 NORTE LOC12 LOC7
43: DROP ROBOT1 LOC7 MANZANA
44: TURN−RIGHT ROBOT1 NORTE ESTE
45: GOTO ROBOT1 ESTE LOC7 LOC8
46: GOTO ROBOT1 ESTE LOC8 LOC9
47: PICK ROBOT1 LOC9 OSCAR
48: TURN−LEFT ROBOT1 ESTE NORTE
49: TURN−LEFT ROBOT1 NORTE OESTE
50: GOTO ROBOT1 OESTE LOC9 LOC8

51: GOTO ROBOT1 OESTE LOC8 LOC7
52: GOTO ROBOT1 OESTE LOC7 LOC6
53: TURN−RIGHT ROBOT1 OESTE NORTE
54: GOTO ROBOT1 NORTE LOC6 LOC1
55: TURN−RIGHT ROBOT1 NORTE ESTE
56: GIVE ROBOT1 LOC1 OSCAR PRINCESA
57: GOTO ROBOT1 ESTE LOC1 LOC2
58: TURN−RIGHT ROBOT1 ESTE SUR
59: GOTO ROBOT1 SUR LOC2 LOC7
60: TURN−LEFT ROBOT1 SUR ESTE
61: PICK ROBOT1 LOC7 MANZANA
62: GOTO ROBOT1 ESTE LOC7 LOC8
63: TURN−LEFT ROBOT1 ESTE NORTE
64: GOTO ROBOT1 NORTE LOC8 LOC3
65: TURN−RIGHT ROBOT1 NORTE ESTE
66: GOTO ROBOT1 ESTE LOC3 LOC4
67: GOTO ROBOT1 ESTE LOC4 LOC5
68: TURN−RIGHT ROBOT1 ESTE SUR
69: GIVE ROBOT1 LOC5 MANZANA PRINCIPE
70: GOTO ROBOT1 SUR LOC5 LOC10
71: GOTO ROBOT1 SUR LOC10 LOC15
72: TURN−RIGHT ROBOT1 SUR OESTE
73: GOTO ROBOT1 OESTE LOC15 LOC14
74: GOTO ROBOT1 OESTE LOC14 LOC13
75: GOTO ROBOT1 OESTE LOC13 LOC12
76: TURN−LEFT ROBOT1 OESTE SUR
77: GOTO ROBOT1 SUR LOC12 LOC17
78: GOTO−SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
79: PICK ROBOT1 LOC22 ZAPATILLA
80: TURN−LEFT ROBOT1 SUR ESTE
81: GOTO−SPECIAL ROBOT1 ESTE LOC22 LOC23 BOSQUE
↪ ZAPATILLA
82: GOTO ROBOT1 ESTE LOC23 LOC24
83: GOTO ROBOT1 ESTE LOC24 LOC25
84: TURN−LEFT ROBOT1 ESTE NORTE
85: TURN−LEFT ROBOT1 NORTE OESTE
86: DROP ROBOT1 LOC25 ZAPATILLA
87: PULL−BAG ROBOT1 BIKINI
88: DROP ROBOT1 LOC25 BIKINI
89: PICK ROBOT1 LOC25 ZAPATILLA
90: GOTO ROBOT1 OESTE LOC25 LOC24
91: GOTO−SPECIAL ROBOT1 OESTE LOC24 LOC23 BOSQUE
↪ ZAPATILLA
92: PUSH−BAG ROBOT1 ZAPATILLA
93: PICK ROBOT1 LOC23 MANZANA
94: TURN−RIGHT ROBOT1 OESTE NORTE

```
        95: TURN-RIGHT ROBOT1 NORTE ESTE
        96: GOTO ROBOT1 ESTE LOC23 LOC24
        97: GOTO ROBOT1 ESTE LOC24 LOC25
        98: GIVE ROBOT1 LOC25 MANZANA PROFESOR


time spent: 0.00 seconds instantiating 1043 easy, 0 hard action templates
                 0.00 seconds reachability analysis, yielding 148 facts and 267
                      ↪ actions
                 0.00 seconds creating final representation with 140 relevant
                      ↪ facts, 2 relevant fluents
                 0.00 seconds computing LNF
                 0.00 seconds building connectivity graph
                31.46 seconds searching, evaluating 142745 states, to a max
                      ↪ depth of 0
                31.46 seconds total time
```

5.1

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is best-first on 1*g(s) + 5*h(s) where
    metric is plan length

advancing to distance: 25
                            24
                            22
                            21
                            20
                            19
                            18
                            17
                            16
                            14
                            13
```

```
12
11
10
 9
 8
 7
 6
 5
 4
 3
 2
 1
 0

ff: found legal plan as follows

step 0: TURN-RIGHT ROBOT1 NORTE ESTE
        1: GOTO ROBOT1 ESTE LOC1 LOC2
        2: TURN-RIGHT ROBOT1 ESTE SUR
        3: GOTO ROBOT1 SUR LOC2 LOC7
        4: GOTO ROBOT1 SUR LOC7 LOC12
        5: TURN-RIGHT ROBOT1 SUR OESTE
        6: GOTO ROBOT1 OESTE LOC12 LOC11
        7: TURN-LEFT ROBOT1 OESTE SUR
        8: GOTO ROBOT1 SUR LOC11 LOC16
        9: PICK ROBOT1 LOC16 BIKINI
       10: PUSH-BAG ROBOT1 BIKINI
       11: TURN-LEFT ROBOT1 SUR ESTE
       12: GOTO ROBOT1 ESTE LOC16 LOC17
       13: TURN-RIGHT ROBOT1 ESTE SUR
       14: PICK ROBOT1 LOC17 MANZANA
       15: GOTO-SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
       16: TURN-RIGHT ROBOT1 SUR OESTE
       17: GOTO ROBOT1 OESTE LOC22 LOC21
       18: TURN-RIGHT ROBOT1 OESTE NORTE
       19: TURN-RIGHT ROBOT1 NORTE ESTE
       20: GIVE ROBOT1 LOC21 MANZANA BRUJA
       21: GOTO-SPECIAL ROBOT1 ESTE LOC21 LOC22 AGUA BIKINI
       22: TURN-LEFT ROBOT1 ESTE NORTE
       23: GOTO ROBOT1 NORTE LOC22 LOC17
       24: GOTO ROBOT1 NORTE LOC17 LOC12
       25: GOTO ROBOT1 NORTE LOC12 LOC7
       26: PICK ROBOT1 LOC7 OSCAR
       27: GOTO ROBOT1 NORTE LOC7 LOC2
       28: TURN-LEFT ROBOT1 NORTE OESTE
       29: GOTO ROBOT1 OESTE LOC2 LOC1
```

30: TURN−LEFT ROBOT1 OESTE SUR
31: GIVE ROBOT1 LOC1 OSCAR PRINCESA
32: GOTO ROBOT1 SUR LOC1 LOC6
33: TURN−LEFT ROBOT1 SUR ESTE
34: GOTO ROBOT1 ESTE LOC6 LOC7
35: GOTO ROBOT1 ESTE LOC7 LOC8
36: GOTO ROBOT1 ESTE LOC8 LOC9
37: TURN−LEFT ROBOT1 ESTE NORTE
38: TURN−LEFT ROBOT1 NORTE OESTE
39: PICK ROBOT1 LOC9 OSCAR
40: GOTO ROBOT1 OESTE LOC9 LOC8
41: GOTO ROBOT1 OESTE LOC8 LOC7
42: TURN−LEFT ROBOT1 OESTE SUR
43: DROP ROBOT1 LOC7 OSCAR
44: GOTO ROBOT1 SUR LOC7 LOC12
45: TURN−LEFT ROBOT1 SUR ESTE
46: GOTO ROBOT1 ESTE LOC12 LOC13
47: GOTO ROBOT1 ESTE LOC13 LOC14
48: TURN−RIGHT ROBOT1 ESTE SUR
49: TURN−RIGHT ROBOT1 SUR OESTE
50: PICK ROBOT1 LOC14 OSCAR
51: GOTO ROBOT1 OESTE LOC14 LOC13
52: TURN−LEFT ROBOT1 OESTE SUR
53: GIVE ROBOT1 LOC13 OSCAR LEONARDO
54: PICK ROBOT1 LOC13 MANZANA
55: GOTO ROBOT1 SUR LOC13 LOC18
56: TURN−RIGHT ROBOT1 SUR OESTE
57: GOTO ROBOT1 OESTE LOC18 LOC17
58: TURN−LEFT ROBOT1 OESTE SUR
59: GOTO−SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
60: TURN−RIGHT ROBOT1 SUR OESTE
61: GOTO ROBOT1 OESTE LOC22 LOC21
62: TURN−RIGHT ROBOT1 OESTE NORTE
63: TURN−RIGHT ROBOT1 NORTE ESTE
64: GIVE ROBOT1 LOC21 MANZANA BRUJA
65: GOTO−SPECIAL ROBOT1 ESTE LOC21 LOC22 AGUA BIKINI
66: PICK ROBOT1 LOC22 ZAPATILLA
67: GOTO−SPECIAL ROBOT1 ESTE LOC22 LOC23 BOSQUE
↪ ZAPATILLA
68: GOTO ROBOT1 ESTE LOC23 LOC24
69: GOTO ROBOT1 ESTE LOC24 LOC25
70: TURN−LEFT ROBOT1 ESTE NORTE
71: TURN−LEFT ROBOT1 NORTE OESTE
72: DROP ROBOT1 LOC25 ZAPATILLA
73: PULL−BAG ROBOT1 BIKINI
74: DROP ROBOT1 LOC25 BIKINI

```
 75: PICK ROBOT1 LOC25 ZAPATILLA
 76: GOTO ROBOT1 OESTE LOC25 LOC24
 77: GOTO-SPECIAL ROBOT1 OESTE LOC24 LOC23 BOSQUE
     ↪ ZAPATILLA
 78: PUSH-BAG ROBOT1 ZAPATILLA
 79: PICK ROBOT1 LOC23 MANZANA
 80: TURN-RIGHT ROBOT1 OESTE NORTE
 81: TURN-RIGHT ROBOT1 NORTE ESTE
 82: GOTO ROBOT1 ESTE LOC23 LOC24
 83: GOTO ROBOT1 ESTE LOC24 LOC25
 84: TURN-LEFT ROBOT1 ESTE NORTE
 85: GIVE ROBOT1 LOC25 MANZANA PROFESOR
 86: PICK ROBOT1 LOC25 BIKINI
 87: TURN-LEFT ROBOT1 NORTE OESTE
 88: GOTO ROBOT1 OESTE LOC25 LOC24
 89: GOTO-SPECIAL ROBOT1 OESTE LOC24 LOC23 BOSQUE
     ↪ ZAPATILLA
 90: GOTO-SPECIAL ROBOT1 OESTE LOC23 LOC22 AGUA
     ↪ BIKINI
 91: TURN-RIGHT ROBOT1 OESTE NORTE
 92: GOTO ROBOT1 NORTE LOC22 LOC17
 93: GOTO ROBOT1 NORTE LOC17 LOC12
 94: GOTO ROBOT1 NORTE LOC12 LOC7
 95: DROP-SPECIAL ROBOT1 LOC7 BIKINI ARENA
 96: PICK ROBOT1 LOC7 OSCAR
 97: GOTO ROBOT1 NORTE LOC7 LOC2
 98: TURN-LEFT ROBOT1 NORTE OESTE
 99: GOTO ROBOT1 OESTE LOC2 LOC1
100: GIVE ROBOT1 LOC1 OSCAR PRINCESA


time spent: 0.00 seconds instantiating 1043 easy, 0 hard action templates
            0.00 seconds reachability analysis, yielding 148 facts and 267
                 ↪ actions
            0.00 seconds creating final representation with 140 relevant
                 ↪ facts, 7 relevant fluents
            0.00 seconds computing LNF
            0.00 seconds building connectivity graph
           32.36 seconds searching, evaluating 186591 states, to a max
                 ↪ depth of 0
           32.36 seconds total time
```

6.1

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is EHC, if that fails then best-first on 1*g(s) + 5*h(s
    ↪ ) where
    metric is plan length

Cueing down from goal distance: 23 into depth [1]
                                22 [1]
                                21 [1][2][3]
                                20 [1][2][3][4][5]
                                19 [1][2][3]
                                18 [1][2]
                                17 [1]
                                16 [1][2]
                                15 [1][2]
                                14 [1][2]
                                13 [1][2][3]
                                12 [1]
                                11 [1][2]
                                10 [1]
                                 9 [1][2]
                                 8 [1]
                                 7 [1][2][3][4][5][6]
                                 6 [1]
                                 5 [1]
                                 4 [1]
                                 3 [1]
                                 2
                                     ↪ [1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20]
                                     ↪
                                 1 [1]
                                 0

ff: found legal plan as follows

step 0: TURN-LEFT ROBOT2 NORTE OESTE
```

1: TURN−LEFT ROBOT1 NORTE OESTE
2: TURN−LEFT ROBOT1 OESTE SUR
3: GOTO ROBOT1 SUR LOC1 LOC6
4: TURN−LEFT ROBOT1 SUR ESTE
5: GOTO ROBOT1 ESTE LOC6 LOC7
6: TURN−RIGHT ROBOT1 ESTE SUR
7: TURN−RIGHT ROBOT1 SUR OESTE
8: PICK ROBOT1 LOC7 OSCAR
9: TURN−LEFT ROBOT1 OESTE SUR
10: GOTO ROBOT1 SUR LOC7 LOC12
11: TURN−LEFT ROBOT1 SUR ESTE
12: GOTO ROBOT1 ESTE LOC12 LOC13
13: GIVE ROBOT1 LOC13 OSCAR LEONARDO
14: TURN−RIGHT ROBOT1 ESTE SUR
15: GOTO ROBOT1 SUR LOC13 LOC18
16: TURN−RIGHT ROBOT1 SUR OESTE
17: GOTO ROBOT1 OESTE LOC18 LOC17
18: GOTO ROBOT1 OESTE LOC17 LOC16
19: TURN−LEFT ROBOT1 OESTE SUR
20: PICK ROBOT1 LOC16 BIKINI
21: TURN−LEFT ROBOT1 SUR ESTE
22: GOTO ROBOT1 ESTE LOC16 LOC17
23: PUSH−BAG ROBOT1 BIKINI
24: TURN−RIGHT ROBOT1 ESTE SUR
25: GOTO−SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
26: TURN−LEFT ROBOT1 SUR ESTE
27: PICK ROBOT1 LOC22 ZAPATILLA
28: GOTO−SPECIAL ROBOT1 ESTE LOC22 LOC23 BOSQUE
↪ ZAPATILLA
29: GOTO ROBOT1 ESTE LOC23 LOC24
30: DROP ROBOT1 LOC24 ZAPATILLA
31: GOTO ROBOT2 OESTE LOC25 LOC24
32: PICK ROBOT2 LOC24 ZAPATILLA
33: PUSH−BAG ROBOT2 ZAPATILLA
34: GOTO−SPECIAL ROBOT2 OESTE LOC24 LOC23 BOSQUE
↪ ZAPATILLA
35: TURN−LEFT ROBOT2 OESTE SUR
36: TURN−LEFT ROBOT2 SUR ESTE
37: PICK ROBOT2 LOC23 MANZANA
38: GOTO ROBOT2 ESTE LOC23 LOC24
39: GOTO ROBOT2 ESTE LOC24 LOC25
40: GIVE ROBOT2 LOC25 MANZANA PROFESOR
41: PULL−BAG ROBOT1 BIKINI
42: TURN−RIGHT ROBOT2 ESTE SUR
43: TURN−RIGHT ROBOT2 SUR OESTE
44: GOTO ROBOT2 OESTE LOC25 LOC24

```
        45: DROP ROBOT1 LOC24 BIKINI
        46: PICK ROBOT2 LOC24 BIKINI
        47: GOTO-SPECIAL ROBOT2 OESTE LOC24 LOC23 BOSQUE
            ↪ ZAPATILLA
        48: GOTO-SPECIAL ROBOT2 OESTE LOC23 LOC22 AGUA
            ↪ BIKINI
        49: TURN-RIGHT ROBOT2 OESTE NORTE
        50: GOTO ROBOT2 NORTE LOC22 LOC17
        51: GOTO ROBOT2 NORTE LOC17 LOC12
        52: TURN-RIGHT ROBOT2 NORTE ESTE
        53: DROP ROBOT2 LOC12 BIKINI
        54: GOTO ROBOT2 ESTE LOC12 LOC13
        55: GOTO ROBOT2 ESTE LOC13 LOC14
        56: TURN-RIGHT ROBOT2 ESTE SUR
        57: TURN-RIGHT ROBOT2 SUR OESTE
        58: PICK ROBOT2 LOC14 OSCAR
        59: GOTO ROBOT2 OESTE LOC14 LOC13
        60: GIVE ROBOT2 LOC13 OSCAR LEONARDO


time spent: 0.00 seconds instantiating 2086 easy, 0 hard action templates
                    0.00 seconds reachability analysis, yielding 190 facts and 534
                        ↪ actions
                    0.00 seconds creating final representation with 179 relevant
                        ↪ facts, 10 relevant fluents
                    0.00 seconds computing LNF
                    0.00 seconds building connectivity graph
                    0.04 seconds searching, evaluating 1835 states, to a max depth
                        ↪ of 20
                    0.04 seconds total time
```

6.2

```
in.pddl -f ej6_problem.2.pddl

ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.
```

```
checking for cyclic := effects --- OK.

ff: search configuration is EHC, if that fails then best-first on 1*g(s) + 5*h(s
    ↪ ) where
    metric is plan length

Cueing down from goal distance: 10 into depth [1]
                                    9 [1]
                                    8 [1][2][3][4]
                                    6 [1]
                                    5 [1]
                                    4 [1]
                                    3 [1]
                                    2 [1][2][3][4][5][6]
                                    1 [1] --- pruning stopped ---
                                        ↪ [1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20][21][2
                                        ↪
                                    0

ff: found legal plan as follows

step 0: TURN-LEFT ROBOT1 NORTE OESTE
        1: TURN-LEFT ROBOT1 OESTE SUR
        2: GOTO ROBOT1 SUR LOC1 LOC6
        3: TURN-LEFT ROBOT1 SUR ESTE
        4: GOTO ROBOT1 ESTE LOC6 LOC7
        5: TURN-RIGHT ROBOT1 ESTE SUR
        6: PICK ROBOT1 LOC7 OSCAR
        7: GOTO ROBOT1 SUR LOC7 LOC12
        8: TURN-LEFT ROBOT1 SUR ESTE
        9: GOTO ROBOT1 ESTE LOC12 LOC13
       10: GIVE ROBOT1 LOC13 OSCAR LEONARDO
       11: GOTO ROBOT1 ESTE LOC13 LOC14
       12: TURN-RIGHT ROBOT1 ESTE SUR
       13: TURN-RIGHT ROBOT1 SUR OESTE
       14: PICK ROBOT1 LOC14 OSCAR
       15: GOTO ROBOT1 OESTE LOC14 LOC13
       16: TURN-LEFT ROBOT2 NORTE OESTE
       17: GOTO ROBOT1 OESTE LOC13 LOC12
       18: GOTO ROBOT2 OESTE LOC25 LOC24
       19: GOTO ROBOT1 OESTE LOC12 LOC11
       20: TURN-LEFT ROBOT1 OESTE SUR
       21: DROP ROBOT1 LOC11 OSCAR
       22: GOTO ROBOT1 SUR LOC11 LOC16
       23: TURN-LEFT ROBOT1 SUR ESTE
       24: PICK ROBOT1 LOC16 BIKINI
```

```
        25: GOTO ROBOT1 ESTE LOC16 LOC17
        26: TURN-RIGHT ROBOT1 ESTE SUR
        27: PUSH-BAG ROBOT1 BIKINI
        28: GOTO-SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
        29: TURN-LEFT ROBOT1 SUR ESTE
        30: PICK ROBOT1 LOC22 ZAPATILLA
        31: GOTO-SPECIAL ROBOT1 ESTE LOC22 LOC23 BOSQUE
            ↪ ZAPATILLA
        32: GOTO ROBOT1 ESTE LOC23 LOC24
        33: DROP ROBOT1 LOC24 ZAPATILLA
        34: PICK ROBOT2 LOC24 ZAPATILLA
        35: GOTO-SPECIAL ROBOT2 OESTE LOC24 LOC23 BOSQUE
            ↪ ZAPATILLA
        36: TURN-LEFT ROBOT2 OESTE SUR
        37: PUSH-BAG ROBOT2 ZAPATILLA
        38: TURN-LEFT ROBOT2 SUR ESTE
        39: PICK ROBOT2 LOC23 MANZANA
        40: GOTO ROBOT2 ESTE LOC23 LOC24
        41: GOTO ROBOT2 ESTE LOC24 LOC25
        42: GIVE ROBOT2 LOC25 MANZANA PROFESOR


time spent: 0.00 seconds instantiating 2086 easy, 0 hard action templates
                0.00 seconds reachability analysis, yielding 190 facts and 534
                    ↪ actions
                0.00 seconds creating final representation with 179 relevant
                    ↪ facts, 11 relevant fluents
                0.00 seconds computing LNF
                0.00 seconds building connectivity graph
             2467.12 seconds searching, evaluating 5517244 states, to a max
                 ↪ depth of 27
             2467.12 seconds total time
```

7.1

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.
```

checking **for** cyclic := effects −−− OK.

ff: search configuration is EHC, **if** that fails **then** best−first on 1∗g(s) + 5∗h(s ↪) where
metric is plan length

Cueing down from **goal** distance: 21 into depth [1]
20 [1]
19 [1]
18 [1]
17 [1][2][3]
16 [1][2][3][4][5]
15 [1][2]
13 [1]
12 [1][2][3]
11 [1][2][3]
10 [1]
9 [1]
8 [1][2]
7 [1]
6 [1]
5 [1][2][3]
4 [1]
3 [1]
2 [1]
1 [1]
0

ff: found legal plan as follows

step 0: TURN−LEFT ROBOT1 NORTE OESTE
1: TURN−LEFT ROBOT2 NORTE OESTE
2: GOTO ROBOT2 OESTE LOC25 LOC24
3: GOTO ROBOT2 OESTE LOC24 LOC23
4: TURN−LEFT ROBOT1 OESTE SUR
5: GOTO ROBOT1 SUR LOC1 LOC6
6: TURN−LEFT ROBOT1 SUR ESTE
7: GOTO ROBOT1 ESTE LOC6 LOC7
8: TURN−RIGHT ROBOT1 ESTE SUR
9: GOTO ROBOT1 SUR LOC7 LOC12
10: GOTO ROBOT1 SUR LOC12 LOC17
11: TURN−RIGHT ROBOT1 SUR OESTE
12: GOTO ROBOT1 OESTE LOC17 LOC16
13: TURN−LEFT ROBOT1 OESTE SUR
14: TURN−LEFT ROBOT1 SUR ESTE
15: PICK ROBOT1 LOC16 BIKINI

```
        16: GOTO ROBOT1 ESTE LOC16 LOC17
        17: PUSH-BAG ROBOT1 BIKINI
        18: PICK ROBOT1 LOC17 MANZANA
        19: TURN-RIGHT ROBOT1 ESTE SUR
        20: GOTO-SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
        21: TURN-LEFT ROBOT1 SUR ESTE
        22: GOTO ROBOT1 ESTE LOC22 LOC23
        23: GIVE-COOP ROBOT1 ROBOT2 LOC23 BIKINI
        24: PULL-BAG ROBOT1 BIKINI
        25: PUSH-BAG ROBOT2 BIKINI
        26: GIVE-COOP ROBOT1 ROBOT2 LOC23 MANZANA
        27: GOTO-SPECIAL ROBOT2 OESTE LOC23 LOC22 AGUA
            ↪ BIKINI
        28: GOTO ROBOT2 OESTE LOC22 LOC21
        29: GIVE ROBOT2 LOC21 MANZANA BRUJA


time spent: 0.02 seconds instantiating 1553 easy, 28 hard action templates
                0.00 seconds reachability analysis, yielding 181 facts and 568
                    ↪ actions
                0.00 seconds creating final representation with 176 relevant
                    ↪ facts, 11 relevant fluents
                0.00 seconds computing LNF
                0.00 seconds building connectivity graph
                0.00 seconds searching, evaluating 70 states, to a max depth of
                    ↪  5
                0.02 seconds total time
```

7.2

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is EHC, if that fails then best-first on 1*g(s) + 5*h(s
    ↪ ) where
    metric is plan length
```

Cueing down from **goal** distance: 37 into depth [1]
36 [1][2][3]
35 [1][2][3][4][5]
34 [1][2]
32 [1]
31 [1][2][3]
30 [1][2][3]
29 [1]
28 [1]
27 [1][2][3]
26 [1][2]
25 [1]
24 [1][2]
23 [1][2]
22 [1][2]
21 [1][2][3]
20 [1]
19 [1]
18 [1][2][3][4][5][6][7][8][9]
17
↪ [1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16]
↪
16 [1]
15 [1][2][3][4][5][6]
14 [1][2][3][4][5][6][7][8][9][10][11]
13 [1][2][3][4][5][6][7]
12
↪ [1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20][21]
↪
11 [1]
10 [1]
9 [1][2]
8 [1][2][3]
7 [1][2][3][4][5][6][7][8]
6 [1]
5 [1][2]
4 [1][2]
3 [1]
2 [1]
1 [1]
0

ff: found legal plan as follows

step 0: TURN−LEFT ROBOT1 NORTE OESTE

```
 1: TURN−LEFT ROBOT1 OESTE SUR
 2: GOTO ROBOT1 SUR LOC1 LOC6
 3: TURN−LEFT ROBOT1 SUR ESTE
 4: GOTO ROBOT1 ESTE LOC6 LOC7
 5: TURN−RIGHT ROBOT1 ESTE SUR
 6: GOTO ROBOT1 SUR LOC7 LOC12
 7: GOTO ROBOT1 SUR LOC12 LOC17
 8: TURN−RIGHT ROBOT1 SUR OESTE
 9: GOTO ROBOT1 OESTE LOC17 LOC16
10: TURN−LEFT ROBOT1 OESTE SUR
11: TURN−LEFT ROBOT1 SUR ESTE
12: PICK ROBOT1 LOC16 BIKINI
13: GOTO ROBOT1 ESTE LOC16 LOC17
14: PUSH−BAG ROBOT1 BIKINI
15: PICK ROBOT1 LOC17 MANZANA
16: TURN−RIGHT ROBOT1 ESTE SUR
17: GOTO−SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
18: TURN−LEFT ROBOT1 SUR ESTE
19: GOTO ROBOT1 ESTE LOC22 LOC23
20: GOTO ROBOT1 ESTE LOC23 LOC24
21: GOTO ROBOT1 ESTE LOC24 LOC25
22: GIVE−COOP ROBOT1 ROBOT2 LOC25 MANZANA
23: PULL−BAG ROBOT1 BIKINI
24: PUSH−BAG ROBOT2 MANZANA
25: GIVE−COOP ROBOT1 ROBOT2 LOC25 BIKINI
26: GIVE ROBOT2 LOC25 MANZANA PROFESOR
27: PULL−BAG ROBOT2 MANZANA
28: TURN−LEFT ROBOT2 NORTE OESTE
29: GOTO ROBOT2 OESTE LOC25 LOC24
30: GOTO ROBOT2 OESTE LOC24 LOC23
31: GOTO−SPECIAL ROBOT2 OESTE LOC23 LOC22 AGUA
    ↪ BIKINI
32: TURN−RIGHT ROBOT2 OESTE NORTE
33: GOTO ROBOT2 NORTE LOC22 LOC17
34: GOTO ROBOT2 NORTE LOC17 LOC12
35: TURN−RIGHT ROBOT2 NORTE ESTE
36: GOTO ROBOT2 ESTE LOC12 LOC13
37: GOTO ROBOT2 ESTE LOC13 LOC14
38: GOTO ROBOT2 ESTE LOC14 LOC15
39: TURN−RIGHT ROBOT2 ESTE SUR
40: TURN−RIGHT ROBOT2 SUR OESTE
41: TURN−RIGHT ROBOT2 OESTE NORTE
42: GOTO ROBOT2 NORTE LOC15 LOC10
43: GOTO ROBOT2 NORTE LOC10 LOC5
44: TURN−LEFT ROBOT2 NORTE OESTE
45: TURN−LEFT ROBOT2 OESTE SUR
```

46: GIVE ROBOT2 LOC5 MANZANA PRINCIPE
47: GOTO ROBOT2 SUR LOC5 LOC10
48: GOTO ROBOT2 SUR LOC10 LOC15
49: TURN−RIGHT ROBOT1 ESTE SUR
50: TURN−RIGHT ROBOT1 SUR OESTE
51: GOTO ROBOT1 OESTE LOC25 LOC24
52: GOTO ROBOT1 OESTE LOC24 LOC23
53: TURN−RIGHT ROBOT2 SUR OESTE
54: GOTO ROBOT2 OESTE LOC15 LOC14
55: GOTO ROBOT2 OESTE LOC14 LOC13
56: PICK ROBOT1 LOC23 MANZANA
57: GOTO ROBOT2 OESTE LOC13 LOC12
58: TURN−RIGHT ROBOT2 OESTE NORTE
59: GOTO ROBOT2 NORTE LOC12 LOC7
60: GOTO ROBOT2 NORTE LOC7 LOC2
61: TURN−RIGHT ROBOT2 NORTE ESTE
62: TURN−RIGHT ROBOT2 ESTE SUR
63: GOTO ROBOT2 SUR LOC2 LOC7
64: GOTO ROBOT2 SUR LOC7 LOC12
65: GOTO ROBOT2 SUR LOC12 LOC17
66: GOTO−SPECIAL ROBOT2 SUR LOC17 LOC22 AGUA BIKINI
67: TURN−LEFT ROBOT2 SUR ESTE
68: GOTO ROBOT2 ESTE LOC22 LOC23
69: TURN−RIGHT ROBOT2 ESTE SUR
70: TURN−RIGHT ROBOT2 SUR OESTE
71: GIVE−COOP ROBOT1 ROBOT2 LOC23 MANZANA
72: GOTO−SPECIAL ROBOT2 OESTE LOC23 LOC22 AGUA
↪ BIKINI
73: TURN−LEFT ROBOT2 OESTE SUR
74: TURN−LEFT ROBOT2 SUR ESTE
75: TURN−LEFT ROBOT2 ESTE NORTE
76: GOTO ROBOT2 NORTE LOC22 LOC17
77: GOTO ROBOT2 NORTE LOC17 LOC12
78: GOTO ROBOT2 NORTE LOC12 LOC7
79: GOTO ROBOT2 NORTE LOC7 LOC2
80: TURN−RIGHT ROBOT2 NORTE ESTE
81: TURN−RIGHT ROBOT2 ESTE SUR
82: TURN−RIGHT ROBOT2 SUR OESTE
83: GOTO ROBOT2 OESTE LOC2 LOC1
84: TURN−LEFT ROBOT2 OESTE SUR
85: GIVE ROBOT2 LOC1 MANZANA PRINCESA
86: GOTO ROBOT2 SUR LOC1 LOC6
87: TURN−LEFT ROBOT1 OESTE SUR
88: TURN−LEFT ROBOT2 SUR ESTE
89: GOTO ROBOT2 ESTE LOC6 LOC7
90: TURN−LEFT ROBOT1 SUR ESTE

```
 91: GOTO ROBOT1 ESTE LOC23 LOC24
 92: TURN-RIGHT ROBOT1 ESTE SUR
 93: TURN-RIGHT ROBOT1 SUR OESTE
 94: PICK ROBOT1 LOC24 BIKINI
 95: GOTO ROBOT1 OESTE LOC24 LOC23
 96: GOTO-SPECIAL ROBOT1 OESTE LOC23 LOC22 AGUA
     ↪ BIKINI
 97: TURN-RIGHT ROBOT1 OESTE NORTE
 98: GOTO ROBOT1 NORTE LOC22 LOC17
 99: GOTO ROBOT1 NORTE LOC17 LOC12
100: PUSH-BAG ROBOT1 BIKINI
101: GOTO ROBOT1 NORTE LOC12 LOC7
102: PICK ROBOT1 LOC7 OSCAR
103: GIVE-COOP ROBOT1 ROBOT2 LOC7 OSCAR
104: TURN-RIGHT ROBOT2 ESTE SUR
105: GOTO ROBOT2 SUR LOC7 LOC12
106: TURN-LEFT ROBOT2 SUR ESTE
107: GOTO ROBOT2 ESTE LOC12 LOC13
108: TURN-RIGHT ROBOT2 ESTE SUR
109: GIVE ROBOT2 LOC13 OSCAR LEONARDO
110: TURN-RIGHT ROBOT1 NORTE ESTE
111: TURN-RIGHT ROBOT1 ESTE SUR
112: GOTO ROBOT1 SUR LOC7 LOC12
113: TURN-LEFT ROBOT1 SUR ESTE
114: GOTO ROBOT1 ESTE LOC12 LOC13
115: PICK ROBOT1 LOC13 MANZANA
116: GIVE-COOP ROBOT1 ROBOT2 LOC13 MANZANA
117: GOTO ROBOT2 SUR LOC13 LOC18
118: TURN-RIGHT ROBOT2 SUR OESTE
119: GOTO ROBOT2 OESTE LOC18 LOC17
120: TURN-LEFT ROBOT2 OESTE SUR
121: GOTO-SPECIAL ROBOT2 SUR LOC17 LOC22 AGUA BIKINI
122: TURN-RIGHT ROBOT2 SUR OESTE
123: GOTO ROBOT2 OESTE LOC22 LOC21
124: GIVE ROBOT2 LOC21 MANZANA BRUJA


time spent: 0.02 seconds instantiating 1553 easy, 28 hard action templates
            0.00 seconds reachability analysis, yielding 181 facts and 568
                 ↪ actions
            0.00 seconds creating final representation with 176 relevant
                 ↪ facts, 15 relevant fluents
            0.01 seconds computing LNF
            0.00 seconds building connectivity graph
            0.46 seconds searching, evaluating 10897 states, to a max
                 ↪ depth of 21
```

```
                0.49 seconds total time
```

7.3

```
ff: parsing domain file
domain 'BELKAN' defined
 ... done.
ff: parsing problem file
problem 'EJ1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is EHC, if that fails then best-first on 1*g(s) + 5*h(s
    ↪ ) where
    metric is plan length

Cueing down from goal distance: 10 into depth [1]
                                9 [1]
                                8 [1][2][3][4]
                                6 [1]
                                5 [1]
                                4 [1]
                                3 [1]
                                2 [1][2][3][4][5][6]
                                1 [1] --- pruning stopped ---
                                    ↪ [1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20][21][2
                                    ↪
                                0

ff: found legal plan as follows

step 0: TURN-LEFT ROBOT1 NORTE OESTE
        1: TURN-LEFT ROBOT1 OESTE SUR
        2: GOTO ROBOT1 SUR LOC1 LOC6
        3: TURN-LEFT ROBOT1 SUR ESTE
        4: GOTO ROBOT1 ESTE LOC6 LOC7
        5: TURN-RIGHT ROBOT1 ESTE SUR
        6: PICK ROBOT1 LOC7 OSCAR
        7: GOTO ROBOT1 SUR LOC7 LOC12
        8: TURN-LEFT ROBOT1 SUR ESTE
        9: GOTO ROBOT1 ESTE LOC12 LOC13
```

```
10: GIVE ROBOT1 LOC13 OSCAR LEONARDO
11: GOTO ROBOT1 ESTE LOC13 LOC14
12: TURN−RIGHT ROBOT1 ESTE SUR
13: TURN−RIGHT ROBOT1 SUR OESTE
14: PICK ROBOT1 LOC14 OSCAR
15: GOTO ROBOT1 OESTE LOC14 LOC13
16: TURN−LEFT ROBOT2 NORTE OESTE
17: GOTO ROBOT1 OESTE LOC13 LOC12
18: GOTO ROBOT2 OESTE LOC25 LOC24
19: GOTO ROBOT1 OESTE LOC12 LOC11
20: TURN−LEFT ROBOT1 OESTE SUR
21: DROP ROBOT1 LOC11 OSCAR
22: GOTO ROBOT1 SUR LOC11 LOC16
23: TURN−LEFT ROBOT1 SUR ESTE
24: PICK ROBOT1 LOC16 BIKINI
25: GOTO ROBOT1 ESTE LOC16 LOC17
26: TURN−RIGHT ROBOT1 ESTE SUR
27: PUSH−BAG ROBOT1 BIKINI
28: GOTO−SPECIAL ROBOT1 SUR LOC17 LOC22 AGUA BIKINI
29: TURN−LEFT ROBOT1 SUR ESTE
30: PICK ROBOT1 LOC22 ZAPATILLA
31: GOTO−SPECIAL ROBOT1 ESTE LOC22 LOC23 BOSQUE
    ↪ ZAPATILLA
32: GOTO ROBOT1 ESTE LOC23 LOC24
33: DROP ROBOT1 LOC24 ZAPATILLA
34: PICK ROBOT2 LOC24 ZAPATILLA
35: GOTO−SPECIAL ROBOT2 OESTE LOC24 LOC23 BOSQUE
    ↪ ZAPATILLA
36: TURN−LEFT ROBOT2 OESTE SUR
37: PUSH−BAG ROBOT2 ZAPATILLA
38: TURN−LEFT ROBOT2 SUR ESTE
39: PICK ROBOT2 LOC23 MANZANA
40: GOTO ROBOT2 ESTE LOC23 LOC24
41: GOTO ROBOT2 ESTE LOC24 LOC25
42: GIVE ROBOT2 LOC25 MANZANA PROFESOR


time spent: 0.00 seconds instantiating 2086 easy, 0 hard action templates
            0.00 seconds reachability analysis, yielding 190 facts and 534
                 ↪ actions
            0.00 seconds creating final representation with 179 relevant
                 ↪ facts, 11 relevant fluents
            0.00 seconds computing LNF
            0.00 seconds building connectivity graph
         2467.12 seconds searching, evaluating 5517244 states, to a max
                 ↪ depth of 27
```

2467.12 seconds total time