Temas Tratados en el Trabajo Práctico 6

- Modelado de problemas en espacios de estado.
- Algoritmos de planificación hacia adelante y hacia atrás.
- Representación y solución de problemas descritos en lenguaje STRIPS.
- Algoritmo GRAPHPLAN.
- Planificación con restricciones de tiempo y recursos.
- Caminos críticos y tiempos de relajación.

Ejercicios Teóricos

1. ¿En qué tipo de algoritmos se basa un planificador para encontrar el mejor camino a un estado solución?

Se utilizan algoritmos de busqueda, estos algoritmos pueden ser de busqueda local o total.

1. ¿Qué tres elementos se encuentran dentro de una acción formulada en lenguaje STRIPS? Describa brevemente qué función cumple cada uno.

En el lenguaje STRIPS (Stanford Research Institute Problem Solver), una acción se define con tres elementos principales: - Nombre de la acción y parámetros Es la “etiqueta” que identifica la acción y las variables que intervienen en ella. Ejemplo: Volar(p, desde, hasta) → significa que el avión p se mueve de un aeropuerto desde a otro hasta. - Precondiciones Son las condiciones que deben cumplirse en el estado actual para que la acción pueda ejecutarse. Funcionan como “requisitos previos”. Ejemplo: para volar, el avión debe estar en el aeropuerto de salida y tanto origen como destino deben ser aeropuertos válidos. - Efectos Describen cómo cambia el mundo después de ejecutar la acción. Pueden ser efectos positivos (lo que pasa a ser verdadero) o negativos (lo que deja de ser verdadero). Ejemplo: después de volar, el avión ya no está en el aeropuerto de origen (efecto negativo), pero sí en el aeropuerto destino (efecto positivo).

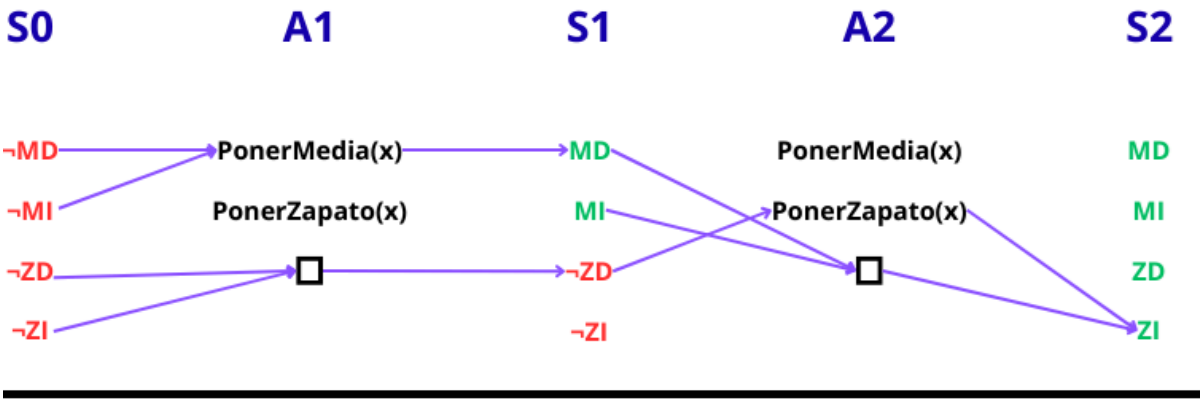
1. Describa las ventajas y desventajas de desarrollar un algoritmo de planificación hacia adelante y hacia atrás en el espacio de estados.

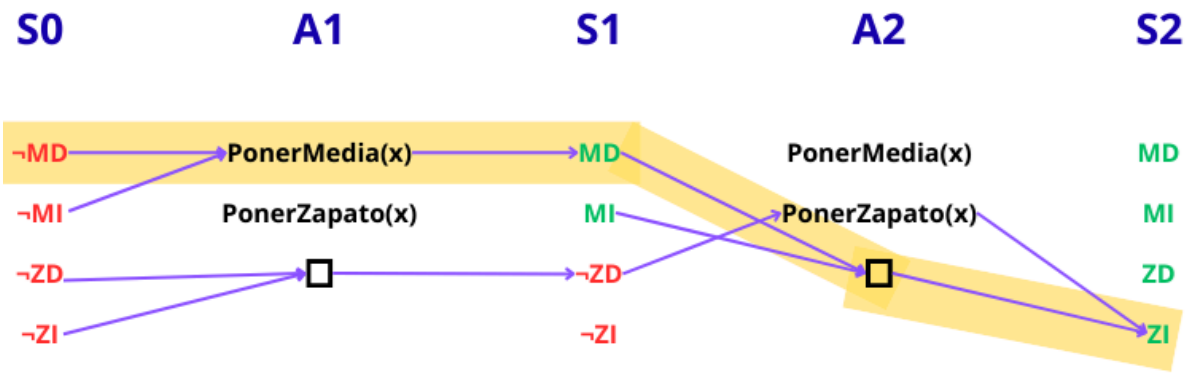
1. Planificación hacia adelante (Forward Search) Cómo funciona: Se parte del estado inicial y se aplican acciones que son posibles (cuyas precondiciones se cumplen). Cada acción genera un nuevo estado, y se sigue expandiendo hasta encontrar uno que cumpla el objetivo. Ventajas: - Es más intuitiva (se parece a como pensamos: “desde lo que tengo, qué puedo hacer”). - Es directa: siempre se trabaja con estados que son alcanzables de verdad desde la situación inicial. - Se adapta bien cuando el número de estados objetivos es grande o poco definido. Desventajas: - Puede explorar muchos caminos irrelevantes, porque no siempre guía bien hacia el objetivo. - El espacio de búsqueda puede crecer muchísimo si no hay una buena heurística. 2. Planificación hacia atrás (Backward Search) Cómo funciona: Se parte del objetivo y se busca qué acciones podrían producirlo (mirando sus efectos). Luego se agregan como nuevos subobjetivos las precondiciones de esas acciones, y así se sigue hasta llegar al estado inicial. Ventajas: - Está más

enfocada en el objetivo, porque sólo considera acciones que realmente ayudan a alcanzarlo. - Puede reducir mucho el espacio de búsqueda cuando los objetivos son pocos o muy específicos. Desventajas: - Puede generar subobjetivos difíciles de alcanzar o incluso inconsistentes (que no se derivan del estado inicial). - A veces es menos natural de implementar que la búsqueda hacia adelante.

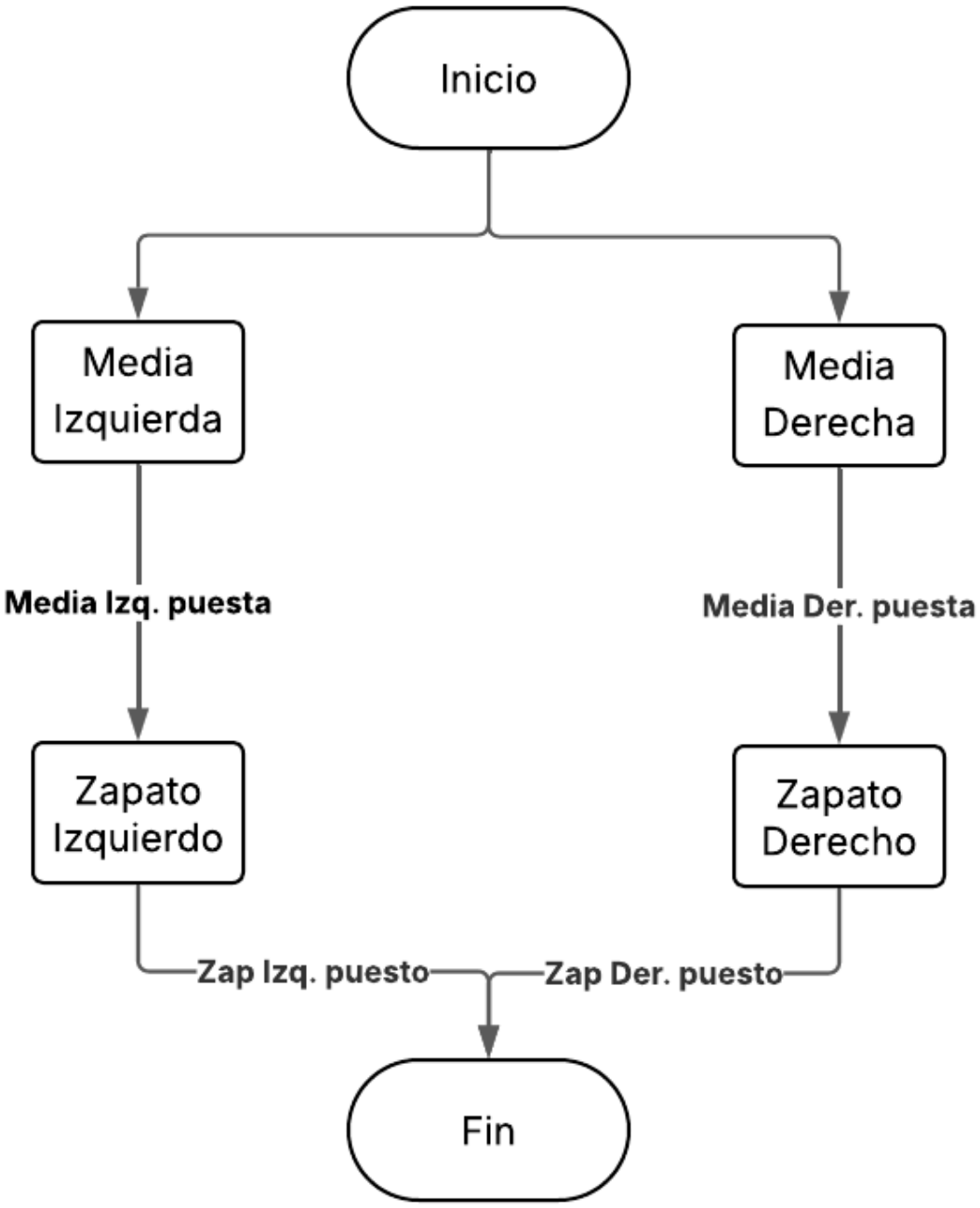
1. Considere el problema de ponerse uno mismo zapatos y medias. Aplique GRAPHPLAN a este problema y muestre la solución obtenida. Muestre el plan de orden parcial que es solución e indique cuántas linealizaciones diferentes existen para el plan de orden parcial.

Graphplan





Plan de Orden Parcial



Existen 6 linealizaciones posibles: MI → MD → ZI → ZD
MI → MD → ZD → ZI
MI → ZI → MD → ZD
MD → MI → ZI → ZD
MD → MI → ZD → ZI
MD → ZD → MI → ZI

1. Se requiere ensamblar una máquina cuyas piezas están identificadas con las letras A, B, C, D y E. El tiempo que se tarda en ensamblar cada pieza es:

- A: 2 semanas
- B: 1 semana
- C: 4 semanas

- D: 3 semanas
- E: 5 semanas

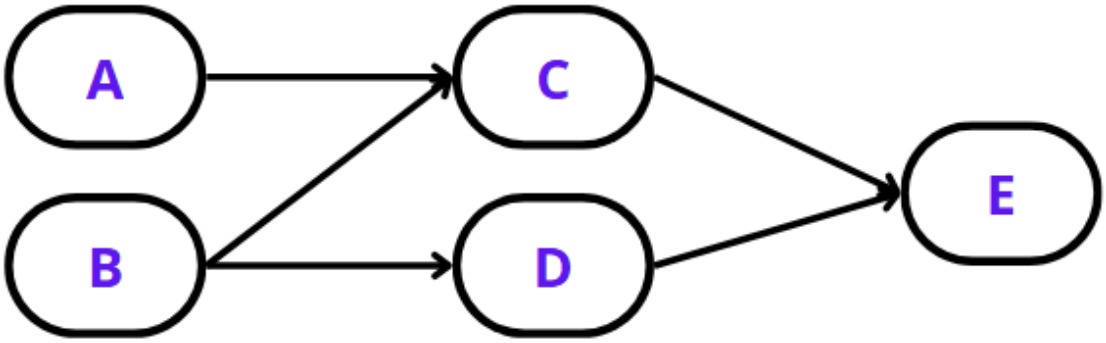
El orden de ensamblaje de cada pieza requiere que:

- A esté realizado antes que C
- B esté realizado antes que C
- B esté realizado antes que D
- C esté realizado antes que E
- D esté realizado antes que E

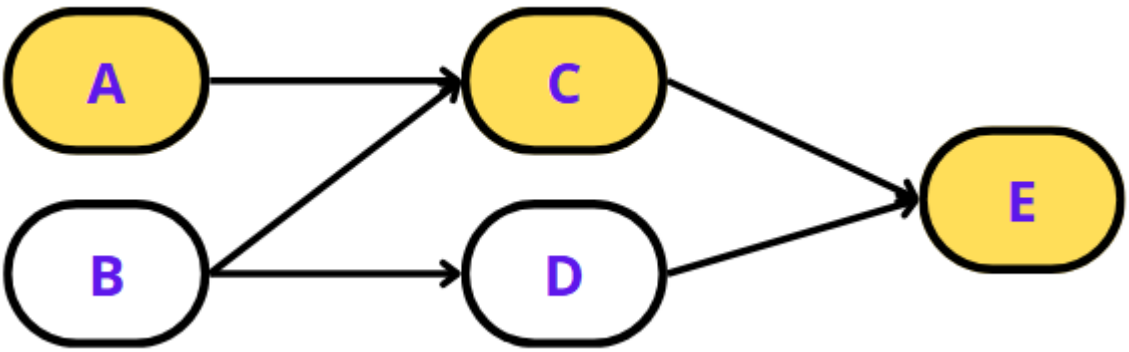
Con esta información:

```
5.1 Arme el Plan de Orden Parcial.

5.2 Encuentre el Camino Crítico.

5.3 Encuentre los tiempos de relajación.

5.4 Dibuje un diagrama temporal indicando las tareas y los tiempos de
relajación encontrados.
```

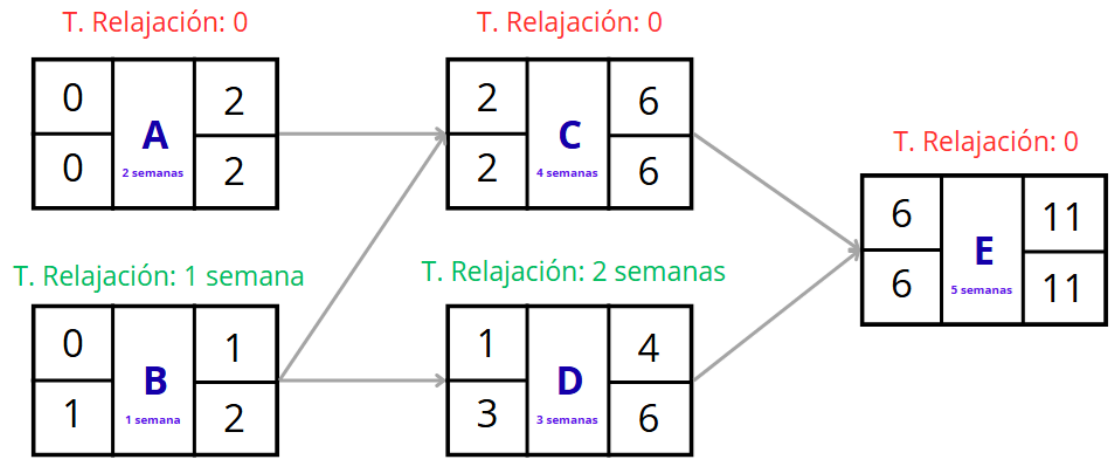
5.1 Plan de Orden Parcial



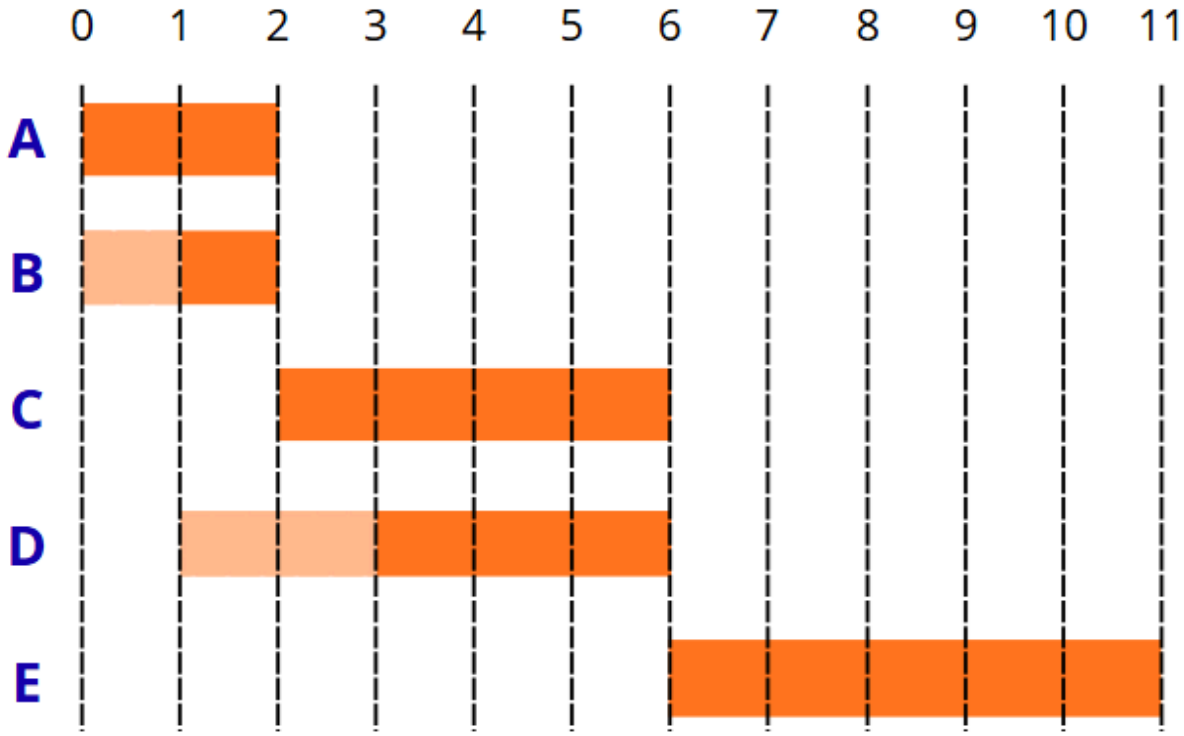
5.2 Camino Crítico



5.3 Tiempos de relajación



5.4 Diagrama Temporal



Ejercicios de Implementación

1. Suponga que tiene un robot de oficina capaz de moverse y tomar y depositar objetos. El robot solo puede tener un objeto a la vez, pero puede conseguir una *caja* en la que depositar varios objetos. Suponga que programa al robot para *ir a la tienda* a comprarle un *café* y en el camino de vuelta tome una *carta* del *buzón* de la oficina para para que se la traiga junto con el café. Describa en lenguaje STRIPS:

   6.1 El dominio del robot (nombre, predicados y acciones que puede hacer el robot).

   6.2 El problema que se quiere resolver (estado inicial, estado objetivo y objetos del mundo representados).

   6.3 Introduzca el código desarrollado en los puntos anteriores en el

[planificador online](http://lcas.lincoln.ac.uk/fast-downward/) y obtenga el plan de acción que tomará el robot para cumplir lo solicitado.

6.1 - Domimnio del robot

Nombre del dominio: robot-cafe-carta

Predicados: (Hechos que describen el estado del mundo)

- en(objeto, lugar) — El objeto está en un lugar.
- en-robot(lugar) — El robot está en un lugar.
- tiene(objeto) — El robot tiene el objeto en la mano.
- mano-libre — El robot no tiene nada en la mano.
- tiene-caja — El robot está llevando una caja.
- en-caja(objeto) — El objeto está dentro de la caja.

Acciones que puede hacer el robot:

- Moverse de un lugar a otro.
- Agarrar un objeto (si está en el mismo lugar y tiene la mano libre).
- Soltar un objeto (lo deja en el lugar actual).
- Agarrar una caja (si está en el mismo lugar).
- Guardar un objeto en la caja (si lo tiene en la mano y está llevando la caja)

6.2 - Problema

Estado inicial:

- El robot está en la oficina, con la mano libre.
- El café está en la tienda.
- La carta está en el buzón.
- Hay una caja en la oficina.

Objetivo (estado final deseado):

- El café está en la oficina.
- La carta está en la oficina.

Objetos del mundo:

- robot

- cafe, carta, caja
- oficina, tienda, buzon (lugares)

6.3 - Codigo y pasos

In [ ]:

```
;;DEFINICION DE DOMINIO

(define (domain robot-cafe-carta)
  ;; Se define el nombre del dominio
  (:requirements :strips)
  ;; Se indica el uso del modelo STRIPS (acciones con precondiciones y efectos)

  (:predicates
    ;; Predicados que representan el estado del mundo

    (en ?obj ?lugar)
    ;; El objeto ?obj está en el lugar ?lugar

    (en-robot ?lugar)
    ;; El robot está en el lugar ?lugar

    (tiene ?obj)
    ;; El robot tiene el objeto ?obj en la mano

    (en-caja ?obj)
    ;; El objeto ?obj está dentro de la caja

    (tiene-caja)
    ;; El robot está transportando la caja

    (mano-libre)
    ;; El robot tiene la mano libre (no lleva nada)
  )

  ;; Acción: mover el robot de un lugar a otro
  (:action mover
    :parameters (?from ?to)
    :precondition (en-robot ?from)
    ;; El robot debe estar en el lugar de origen
    :effect (and (not (en-robot ?from)) (en-robot ?to))
    ;; El robot ya no está en el origen y ahora está en el destino
  )

  ;; Acción: agarrar un objeto si está en el mismo lugar y la mano está libre
  (:action agarrar
    :parameters (?obj ?lugar)
    :precondition (and (en ?obj ?lugar) (en-robot ?lugar) (mano-libre))
    ;; El objeto debe estar en el mismo lugar que el robot y la mano debe estar libr
    :effect (and (not (en ?obj ?lugar)) (not (mano-libre)) (tiene ?obj))
    ;; El objeto ya no está en el lugar, la mano ya no está libre y el robot lo tien
  )

  ;; Acción: soltar un objeto en el lugar donde está el robot
  (:action soltar
    :parameters (?obj ?lugar)
    :precondition (and (tiene ?obj) (en-robot ?lugar))
    ;; El robot debe tener el objeto y estar en el lugar donde quiere soltarlo
    :effect (and (en ?obj ?lugar) (not (tiene ?obj)) (mano-libre))
    ;; El objeto ahora está en el lugar, el robot ya no lo tiene y la mano queda lib
```

```
  )

  ;; Acción: agarrar la caja si está en el mismo lugar que el robot
  (:action agarrar-caja
    :parameters (?lugar)
    :precondition (and (en caja ?lugar) (en-robot ?lugar))
    ;; La caja debe estar en el lugar y el robot también
    :effect (and (tiene-caja) (not (en caja ?lugar)))
    ;; El robot ahora lleva la caja y la caja ya no está en el lugar
  )

  ;; Acción: guardar un objeto en la caja si se tiene la caja y el objeto
  (:action guardar-en-caja
    :parameters (?obj)
    :precondition (and (tiene ?obj) (tiene-caja))
    ;; El robot debe tener el objeto en la mano y también estar llevando la caja
    :effect (and (en-caja ?obj) (not (tiene ?obj)) (mano-libre))
    ;; El objeto está ahora en la caja, el robot ya no lo tiene y la mano queda libr
  )
)
```

In [ ]:

```
;; DEFINICION DE PROBLEMA

(define (problem llevar-cafe-carta)
  ;; Definimos el nombre del problema: "llevar-cafe-carta"

  (:domain robot-cafe-carta)
  ;; Este problema usa el dominio llamado "robot-cafe-carta"

  (:objects
    robot cafe carta caja tienda buzon oficina
  )
  ;; Lista de objetos que existen en este mundo:
  ;; - robot: el agente que se mueve y actúa
  ;; - cafe, carta: los objetos que debe transportar
  ;; - caja: recipiente opcional para transportar más de un objeto
  ;; - tienda, buzon, oficina: los lugares del mapa

  (:init
    (en cafe tienda)
    ;; El café está inicialmente en la tienda

    (en carta buzon)
    ;; La carta está inicialmente en el buzón

    (en caja oficina)
    ;; La caja está inicialmente en la oficina

    (en-robot oficina)
    ;; El robot comienza en la oficina

    (mano-libre)
    ;; El robot tiene la mano libre al inicio
  )

  (:goal
    (and
      (en cafe oficina)
      ;; Queremos que el café termine en la oficina

      (en carta oficina)
      ;; Y también que la carta termine en la oficina
    )
```

```
    )
)
```

Plan segun planificador web

0: (mover oficina buzon)
1: (agarrar carta buzon)
2: (mover buzon oficina)
3: (soltar carta oficina)
4: (mover oficina tienda)
5: (agarrar cafe tienda)
6: (mover tienda oficina)
7: (soltar cafe oficina)
; cost = 8 (unit cost)
1. Va al buzón → busca primero la carta.
2. Agarra la carta.
3. Vuelve a la oficina.
4. Deja la carta.
5. Va a la tienda.
6. Agarra el café.
7. Vuelve a la oficina.
8. Deja el café.

Bibliografía

Russell, S. & Norvig, P. (2004) *Inteligencia Artificial: Un Enfoque Moderno*. Pearson Educación S.A. (2a Ed.) Madrid, España

Poole, D. & Mackworth, A. (2023) *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press (3a Ed.) Vancouver, Canada