

## TALLER SESION No. 6

## PLANIFICACIÓN AUTOMÁTICA

**Profesor:** Jaime Alberto Guzmán Luna

---

Contenido del taller:

1. Lenguaje de definición de Dominios de Planificación (Planning Domain Definition Language PDDL)
2. STRIPS en práctica de PDDL
3. Actividad propuesta

Introducir al estudiante en las metodologías y herramientas para la solución de problemas a través de técnicas de planificación en IA (PDDL y STRIPS)

Identificar la lógica de la planificación basada en estados.

Familiarizarse con la sintaxis de PDDL.

---

## INTRODUCCION

### Sintaxis y Elementos a tener en cuenta

Un problema de planificación en PDDL, está compuesto por dos archivos:

- Un archivo de dominio: Con la especificación de predicados y acciones del dominio.
- Un archivo del problema: Con la especificación de los objetos, estado inicial y objetivo.

### Archivo de Dominio

#### **+Definición del nombre:**

```
(define (domain <domain name>)
    ....
)
```

<domain name> Nombre para identificar el dominio que se está planteando

#### **+Definición de requerimientos:**

```
(:requirements :typing)
```

Definición de elementos necesarios que debe tener el planificador para correr ese dominio.

#### **+ Definición de predicados:**

```
(:predicates <pred-name> ?<variables>)
```

Propiedades de los objetos que nos interesan; puede ser verdadera o falsa. Se definen como: Nombre del predicado y la variable que lo identifica antecedido por "?". Ejemplos: (ROOM ?x) (BALL ?x) (carry ?x ?y).

**+ Definición de Acciones**

```
(:action <nombre> :parameters (<parámetros(variables)>)
:precondition (<precondiciones(predicados)>)
:effect (<efectos (predicados)>)
)
```

Para las precondiciones y efectos, es posible usar operadores lógicos con el fin de componer una proposición más compleja:

Conjunción ( $\wedge$ ): AND,

Ejemplo: (and (ROOM ?x) (ROOM ?y))

Negación ( $\neg$ ): NOT

Ejemplo: (not (ROOM ?x))

\* El uso del operador NOT es restringido solo para la especificación de efectos de supresión.

*Ejemplo de una acción en pddl*

```
(:action move :parameters (?x ?y)
:precondition (and (ROOM ?x) (ROOM ?y) (at-robby ?x))
:effect (and (at-robby ?y) (not (at-robby ?x))))
```

**Archivo de Problema**

```
(define (problem <problem name>)
(:domain <domain name>)
(:objects <objetos (valores (no variables))>)
(:init <predicados estado inicial (valores(no variables))>)
(:goal <predicados del estado objetivo>)
)
```

**Actividad 1: Notación STRIPS en PDDL**

Analice el dominio y el problema, citados a continuación. En este dominio se utiliza uno o más robots móviles para repartir el correo en una planta de despachos. Los robots pueden desplazarse por una serie de puntos de referencia dentro de la planta, y son capaces de recoger y entregar cartas y de abrir y cerrar las puertas de los despachos. Los robots, además, pueden comprobar el estado de las puertas (abiertas o cerradas) y verificar si una persona o una carta se encuentra en un determinado despacho.

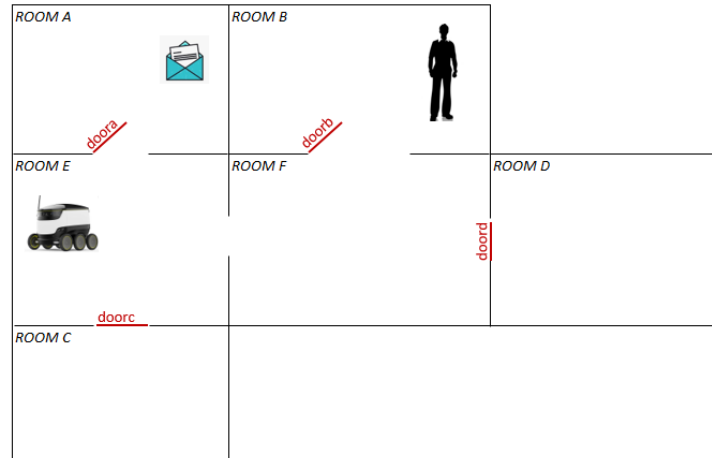


Figura 1. Robot Delivery estado inicial

**Archivo de Dominio (RobotDomain.pddl)**

```
(define (domain RobotDelivery)
```

```
(:requirements :strips)
```

```
(:predicates
```

```
(at ?x ?y)
```

```
(atl ?x ?y)
```

```
(atr ?x ?y)
```

```
(in ?l ?r)
```

```
(open ?d)
```

```
(delivered ?l ?p)
```

```
(connected ?p1 ?p2)
```

```
(conn-door ?p1 ?p2 ?d)
```

```
(next ?p ?d))
```

```
(:action Open-door
```

```
  :parameters (?r ?d ?p)
```

```
  :precondition (and (not (open ?d)) (atr ?r ?p) (next ?p ?d))
```

```
  :effect (and (open ?d))
```

```
)
```

```
(:action Travel
```

```
  :parameters (?r ?p1 ?p2)
```

```
  :precondition (and (atr ?r ?p1) (connected ?p1 ?p2))
```

```
  :effect (and (not (atr ?r ?p1)) (atr ?r ?p2))
```

```
)
```

```
(:action Travel-door
```

```
  :parameters (?r ?p1 ?p2 ?d)
```

```
  :precondition (and (atr ?r ?p1) (conn-door ?p1 ?p2 ?d) (open ?d))
```

```
  :effect (and (not (atr ?r ?p1)) (atr ?r ?p2))
```

```
)
```

```
(:action Deliver-letter
```

```
  :parameters (?r ?l ?p ?x)
```

```
  :precondition (and (atr ?r ?x) (at ?p ?x) (in ?l ?r))
```

```
  :effect (and (not (in ?l ?r)) (delivered ?l ?p))
```

```

)

(:action Pickup-letter
  :parameters (?r ?l ?p)
  :precondition (and (atr ?r ?p) (atl ?l ?p))
  :effect (and (not (atl ?l ?p)) (in ?l ?r))
)

)

```

### Archivo de Problema (RobotProblem.pddl)

En este problema el robot debe entregar la carta a la persona. La especificación en PDDL es la siguiente:

```

(define (problem Problem1)
  (:domain RobotDelivery)
  (:objects
    p
    r
    l
    A B C D E F
    doorA doorB doorC doorD)

  (:init
    (atr r E) (at p B) (atl l A)
    (open doorA)
    (open doorB)
    (connected E F)
    (connected F E)
    (conn-door A E doorA)
    (conn-door E A doorA)
    (conn-door E C doorC)
    (conn-door C E doorC)
    (conn-door B F doorB)
    (conn-door F B doorB)
    (conn-door D F doorD)
    (conn-door F D doorD)
    (next A doorA) (next E doorA) (next E doorC) (next C doorC)
    (next F doorB) (next B doorB) (next F doorD) (next D doorD)
  )

  (:goal (delivered l p))
)

```

### ANALISIS DEL PROBLEMA

Crearemos los dos archivos RobotDomain.pddl y RobotProblem.pddl haciendo uso del PDDL Editor asociado a la siguiente página:

<http://editor.planning.domains/>

Para ello renombre el archivo existente dándole doble click sobre su nombre para editarlo. Póngale el nombre de RobotDomain.pddl y copie el código en su respectiva ventana. Seguido

Cree un nuevo archivo haciendo uso del menú File-New y póngale el nombre de RobotProblem.pddl y pddl y copie el código en su respectiva ventana. Los dos archivos quedarán así respectivamente:

Apartes del dominio:

← → ↻ No es seguro | editor.planning.domains/#

### PDDL Editor

File Session Import Solve Plugins Help

RobotDomain.pddl

RobotProblem.pddl

```

1 (define (domain RobotDelivery)
2   (:requirements :strips)
3   (:predicates
4     (at ?x ?y)
5     (atl ?x ?y)
6     (atr ?x ?y)
7     (in ?l ?r)
8     (open ?d)
9     (delivered ?l ?p)
10    (connected ?p1 ?p2)
11    (conn-door ?p1 ?p2 ?d)
12    (next ?p ?d))
13
14   (:action Open-door
15     :parameters (?r ?d ?p)
16     :precondition (and (not(open ?d)) (atr ?r ?p) (next ?p ?d))
17     :effect (and (open ?d))
18   )
19

```

Apartes del problema:

### PDDL Editor

File Session Import Solve Plugins Help

RobotDomain.pddl

RobotProblem.pddl

```

1 (define (problem Problem1)
2   (:domain RobotDelivery)
3   (:objects
4     p
5     r
6     l
7     A B C D E F
8     doorA doorB doorC doorD)
9
10  (:init
11    (atr r E) (at p B) (atl l A)
12    (open doorA)
13    (open doorB)
14    (connected E F)
15    (connected F E)
16    (conn-door A E doorA)
17    (conn-door E A doorA)
18    (conn-door E C doorC)
19    (conn-door C E doorC)
20    (conn-door B F doorB)
21    (conn-door F B doorB)
22    (conn-door D F doorD)
23    (conn-door F D doorD)
24    (next A doorA) (next E doorA) (next E doorC) (next C doorC)
25    (next F doorB) (next B doorB) (next F doorD) (next D doorD)
26  )
27
28  (:goal (delivered l p))
29

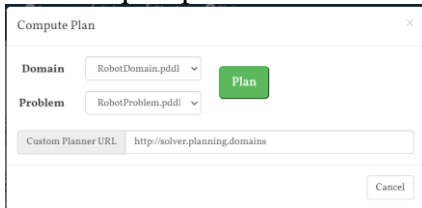
```

Primero analice el archivo del dominio e identifique:

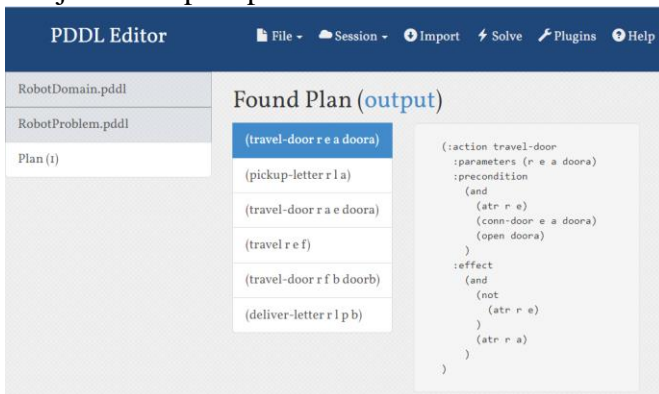
- Que representan los siguientes predicados?  
 (atr ?x ?y) \_\_\_\_\_  
 (conn-door ?p1 ?p2 ?d) \_\_\_\_\_  
 (next ?p ?d) \_\_\_\_\_  
 (delivered ?l ?p) \_\_\_\_\_
- Cuales son las acciones definidas en dicho problema?  
 \_\_\_\_\_,  
 \_\_\_\_\_
- Cuales son los significados de los parámetros manejados en la acción Travel-door?  
 (?r)\_\_\_\_\_, (?p1)\_\_\_\_\_, (?p2)\_\_\_\_\_, (?d)\_\_\_\_\_
- Cuales son las precondiciones en lenguaje natural de Travel-door?  
 \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_
- Cuales son los efectos en lenguaje natural de Travel-door?  
 \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

## EJECUTANDO UN PLANIFICADOR

Para ejecutar el planificador implementado en esta página (el cual es un planificador muy básico que maneja solo el lenguaje Strips), ir al menú solver y dar click al botón plan en la ventana que aparece.



Ahora dar doble click sobre el archivo Plan (I) que aparece en el menú de la izquierda luego de ejecutar el plan para abrir el archivo solución.



Pregunta 1. Analice el plan solución y escriba en lenguaje natural cada uno de las soluciones indicadas por dicho plan:

Acción 1: \_\_\_\_\_

Acción 2: \_\_\_\_\_

Acción 3: \_\_\_\_\_

Acción 4: \_\_\_\_\_

Acción 5: \_\_\_\_\_

Acción 6: \_\_\_\_\_

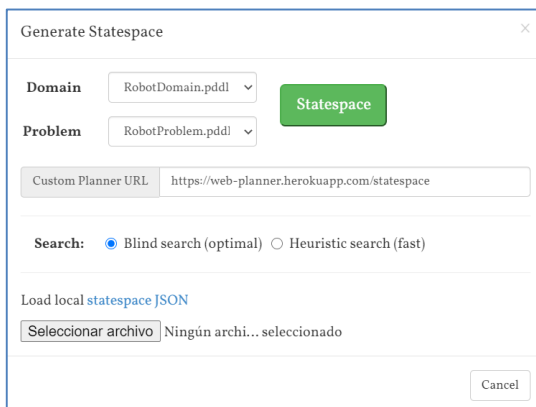
Pregunta 2. En el paso 2 del plan que es (pickup-letter r l a) en lenguaje natural cuales son las precondiciones que se indica se debieron cumplir:

Precondición 1: \_\_\_\_\_

Precondición 2: \_\_\_\_\_

### **ANALIZAR EL ESPACIO DE BUSQUEDA DEL PLANIFICADOR**

Desde el menú Plugins, instalar la opción Statespace (Visualize state-space explored during planning) haciendo click sobre su botón Install. Luego de esto aparecerá la opción en el menú superior de la aplicación. Darle click en esta y ejecutar la búsqueda según la figura



Generate Statespace

Domain: RobotDomain.pddl

Problem: RobotProblem.pddl

Statespace

Custom Planner URL: https://web-planner.herokuapp.com/statespace

Search: ☒ Blind search (optimal) ☐ Heuristic search (fast)

Load local statespace JSON

Seleccionar archivo | Ningún archi...seleccionado

Cancel

El resultado es el siguiente gráfico:

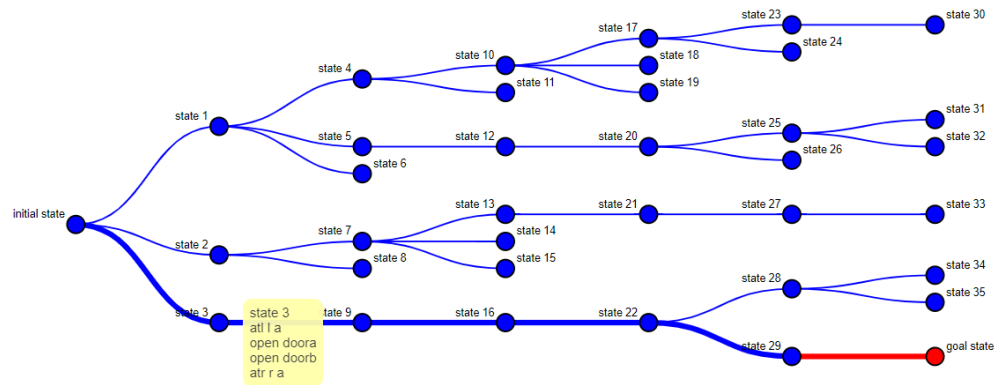
## Statespace

Visited States: 37 Tree Height: 6

0%

100%

Change Layout



La figura detalla el árbol de búsqueda recorrido hasta encontrar el estado meta (nodo rojo). Como se observa en la parte superior izquierda el planificador recorrido de 37 estados y alcanzó una profundidad de 6 niveles.

La línea azul gruesa es el plan solución seleccionado por el planificador.

Igualmente se detalla la descripción del estado 3 (cuadro en amarillo) en el cual se muestra los predicados verdaderos en dicho estado luego de aplicar la primera acción:  
(*travel-door r e a doora*).

## SALVAR ARCHIVOS

En el menú file y parándose en cada uno de los archivos se pueden descargar para guardarlos.

