UNIVERSIDAD NACIONAL DE COLOMBIA

| TALLER SESION No. 5 | 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 antecedidos 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 (∧): AND,
Ejemplo: (and (ROOM ?x) (ROOM ?y))
Negación (-): 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>)*
*)*

***+ Variables Tipadas***

*Con el fin de brindar un poco más de expresividad y detalle a la especificación STRIPS, PDDL, permite la creación de tipos de variables y así restringir los predicados para un determinado tipo de objeto, la definición de un tipo de variable se realiza como sigue a continuación:*

Para definir los objetos en PDDL se utiliza la siguiente sentencia:
*(:types Nombre - tipo ....)*

Para generar objetos de un tipo de variable determinado, en la definición de *objects* basta hacer una declaración como la siguiente:
*(:objects bathroom – room)*

**NOTA IMPORTANTE**: Por favor recuerde que para generar la extensión *.pddl* del archivo de dominio y del problema, es importante que los guarde con un editor de notas (se recomienda usar Note++) y que seleccione en la opción "Tipo", todos los archivos y en "Nombre" si guardarlo con el nombre que deseen preferiblemente que identifiquen que tipo de archivo es, como en el caso a continuación podría ser *RobotDomain.pddl*

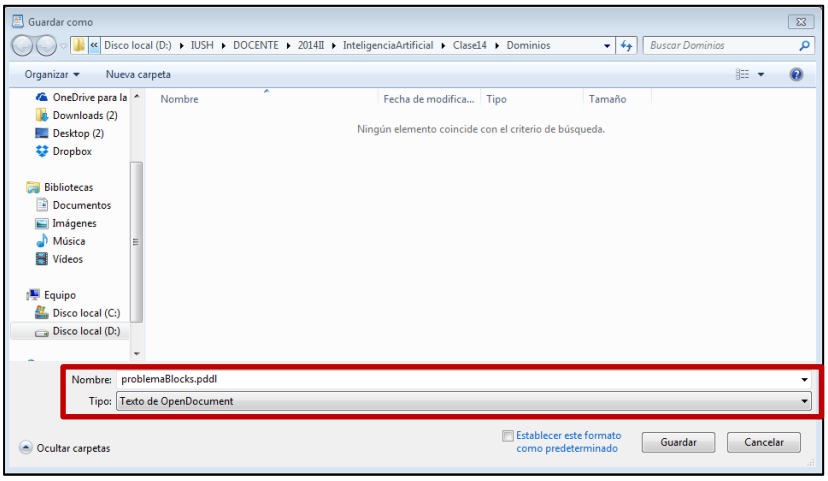



Figura 2. Guardando archivo problem.pddl en notepad++

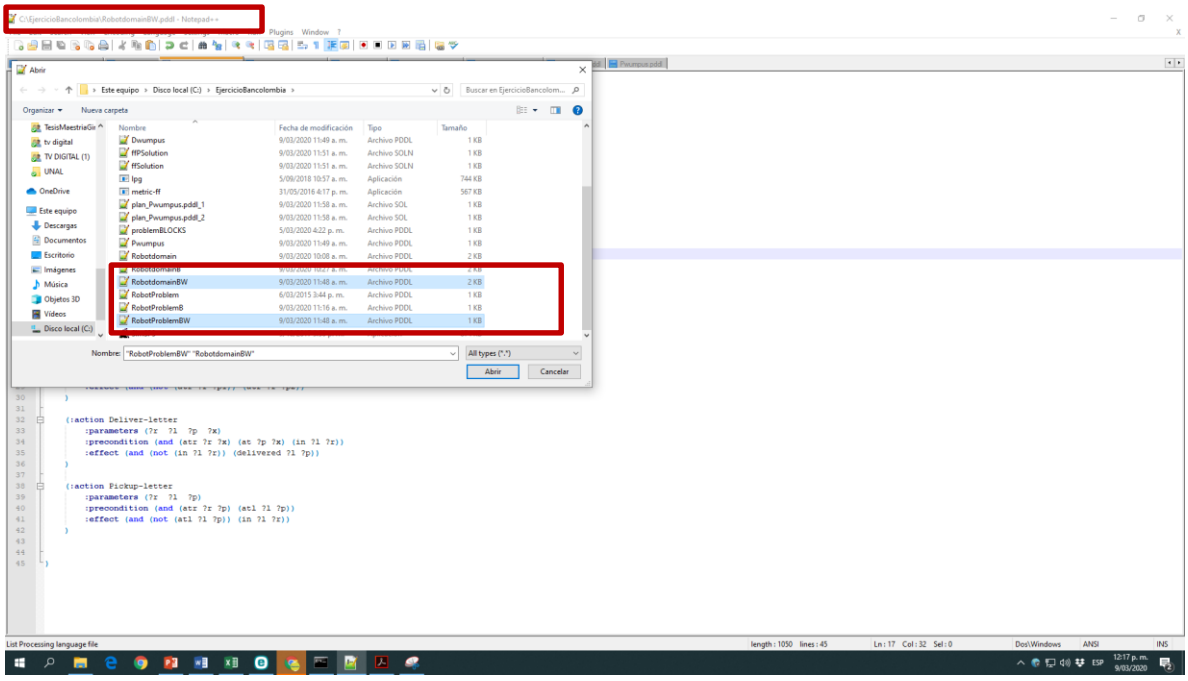



Figura 2. Abriendo archivos pddl en notepad++

## **Actividad 1**: ***STRIPS (SIN TIPADO)***

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.

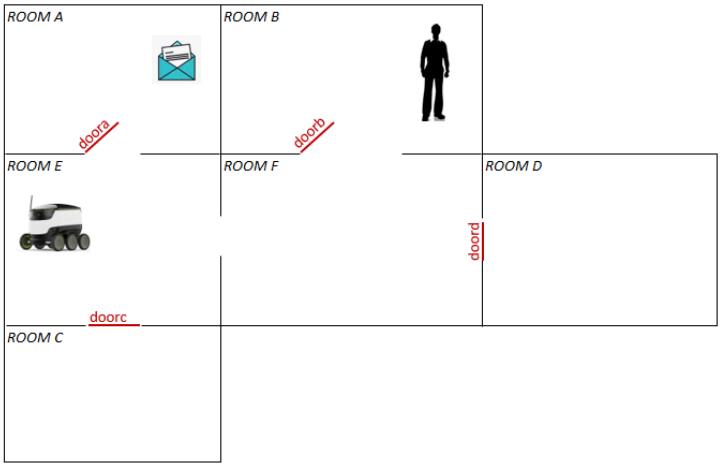



Figura 1. Robot Delivery estado inicial

### Archivo de Dominio

```
(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

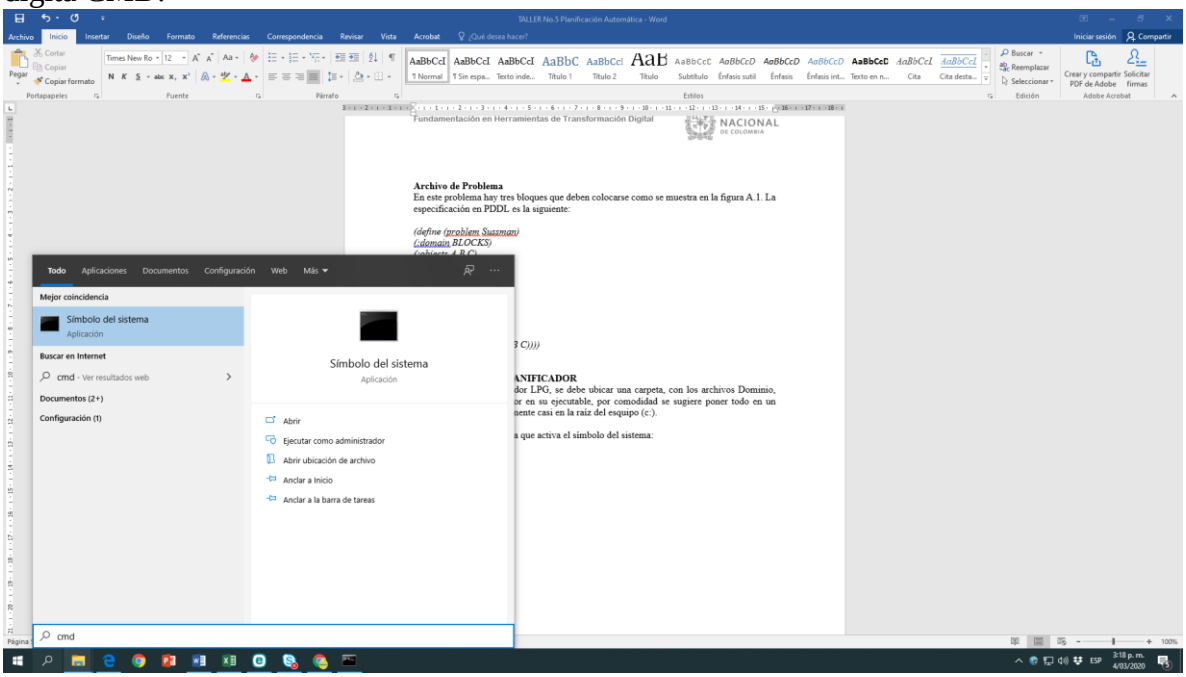
En este problema el robot debe entregar la carta a la persona. La especificación en PDDL es la siguiente:

```
(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))
 )
```

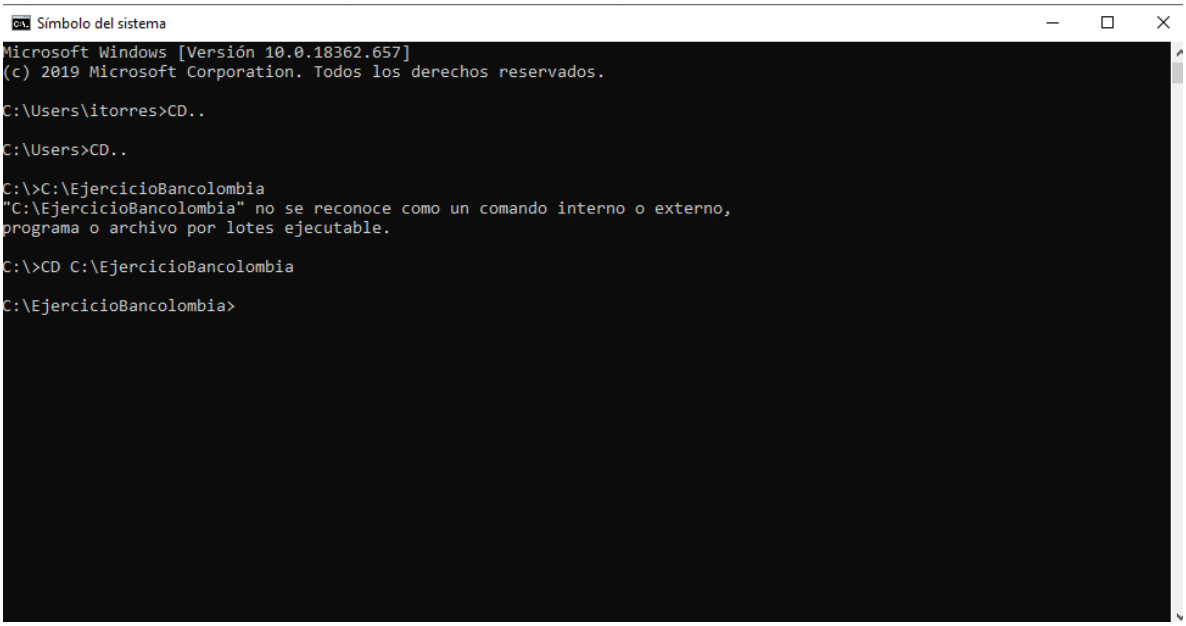
## EJECUTANDO UN PLANIFICADOR

Para ejecutar el planificador LPG, se debe ubicar una carpeta, con los archivos Dominio, Problema y el planificador en su ejecutable, por comodidad se sugiere poner todo en un mismo lugar y preferiblemente casi en la raíz del esquipo (c:).

Ahora se debe abrir la ruta que activa el símbolo del sistema, para ello en el explorador se digita CMD.




con ello, inmediatamente, se abre la ventana que accede el intérprete de comandos del S.O.



Allí se debe hacer uso del comando CD (CD.. ó CD), para actualizar la ruta de la carpeta, donde dejamos nuestros archivos, Domain, Problem y el mismo planificador.

Una vez allí debe invocar el planificador, para ello, en la ruta donde se encuentra almacenados nuestros archivos pddl, y desde el símbolo del sistema digitamos:

*lpg.exe –o nombredelardicvodominio.pddl –f nombredelarchivprobelma.pddl –n 1*

Los comandos lpg.exe –o, -f y –n son obligados, se digitan en minúscula y sin espacio entre el guion y la letra; respecto a los nombres de los archivos, también deben digitarse sin espacios, ni caracteres especiales y deben coincidir exactamente con los nombres de nuestros archivos respectivos. –n es para indicar el número de soluciones que deseamos, por ahora 1 esta perfecta.

Si todo fue exitoso el sistema retornará el correspondiente plan solución:

```
Computing mutex... done

Preprocessing total time: 0.00 seconds

Searching ('.' = every 50 search steps):
 solution found:

Plan computed:
   Time: (ACTION) [action Duration; action Cost]
 0.0000: (TRAVEL-DOOR R E A DOORA) [D:1.0000; C:1.0000]
 1.0000: (PICKUP-LETTER R L A) [D:1.0000; C:1.0000]
 2.0000: (TRAVEL-DOOR R A E DOORA) [D:1.0000; C:1.0000]
 3.0000: (TRAVEL R E F) [D:1.0000; C:1.0000]
 4.0000: (TRAVEL-DOOR R F B DOORB) [D:1.0000; C:1.0000]
 5.0000: (DELIVER-LETTER R L P B) [D:1.0000; C:1.0000]


Solution number: 1
Total time:      0.00
Search time:     0.00
Actions:         6
Execution cost:  6.00
Duration:        6.000
Plan quality:    6.000
     Plan file:       plan_RobotProblemBW.pddl_1.SOL


C:\EjercicioBancolombia>
```

Cualquier otro mensaje indica que hubo un error, en compilación, se deberá verificar.

## Actividad 1. Interpretación

Una vez verificado el éxito de la ejecución, analizar cuidadosamente los archivos dominio, problema y plan. Verifique coherencia de la solución.

## Actividad 2: *STRIPS (CON TIPADO)*

El siguiente, es el mismo problema del robot que reparte cartas, pero con tipado (*types:*)

```
define (domain RobotDelivery)
(:requirements :strips)
(:types locatable door point - object
    person letter robot - locatable)

(:predicates
        (at ?x - person ?y - point)
        (atl ?x - letter ?y - point)
        (atr ?x - robot ?y - point)
        (in ?l - letter ?r - robot)
        (open ?d - door)
        (delivered ?l - letter ?p - person)
        (connected ?p1 - point ?p2 - point)
        (conn-door ?p1 - point ?p2 - point ?d - door)
        (next ?p - point ?d - door))

        (:action Open-door
                :parameters (?r - robot ?d - door ?p - point)
                :precondition (and (not(open ?d)) (atr ?r ?p) (next ?p ?d))
                :effect (and (open ?d)))

        (:action Travel
                :parameters (?r - robot ?p1 - point ?p2 - point)
                :precondition (and (atr ?r ?p1) (connected ?p1 ?p2))
                :effect (and (not (atr ?r ?p1)) (atr ?r ?p2) ))

        (:action Travel-door
                :parameters (?r - robot ?p1 - point ?p2 - point ?d - door)
                :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 - robot ?l - letter ?p - person ?x - point)
                :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 - robot ?l - letter ?p - point)
                :precondition (and (atr ?r ?p) (atl ?l ?p))
                :effect (and (not (atl ?l ?p)) (in ?l ?r)))

        )
```

## Archivo de Problema

```
(define (problem Problem1)
(:domain RobotDelivery)
(:objects
        p - person
        r - robot
        l - letter
        A B C D E F - point
        doorA doorB doorC doorD - door)

(: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))
 )
```

**<u>Actividad 3</u>**: ***<u>STRIPS (CON TIPADO)</u>***

Usando el archivo dominio y problema anterior, adicione los siguientes objetos (no elimine, ni reemplace)

- Una carta l1, que se ubicará en D
- Una carta l2, que se ubicará en E
- Y en los objetivos incluir que todas las cartas (l, l1 y l2) deben ser entregadas a la persona.

Encuentre su plan solución, a continuación, usando el planificador LPG.