Práctica PDDL

El objetivo de esta práctica es familiarizarse con el lenguaje PDDL como un sistema para la representación del conocimiento mediante reglas.

1.- Introducción

¿Qué es un problema de planning?

Un problema de búsqueda cuyo objetivo es encontrar un plan que especifique un conjunto de acciones necesarias para ir desde un estado inicial a un estado objetivo. Para ello se describe: un **conjunto de operadores**, un **estado inicial** y una descripción del estado objetivo, es decir, el **estado final**. El resultado que tendremos será un **plan** que describe las acciones u operadores necesarios para ir desde el estado inicial al estado objetivo.

¿Qué es PDDL?

Planning Domain Definition Language (PDDL) es un lenguaje estándar para la representación de tareas de planificación: resolver **problemas de planning**. Normalmente se utilizan dos ficheros: **dominio** y **problema**. En el fichero de dominio se describen los **operadores** y **predicados** que modelan el problema, mientras que en el fichero de problema se recoge un caso concreto del problema a resolver: la instancia del problema. En este fichero se describen las constantes u **objetos**, así como el **estado inicial** y **estado objetivo**.

En PDDL, lo normal es que el planificador implemente la hipótesis de "**mundo cerrado**", es decir, todos los predicados que no se especifiquen se toman como falsos. Son los predicados los encargados de representar los estados. Por lo tanto, un estado se define como un conjunto de predicados instanciados por los objetos de la instancia. Además, **antes de aplicar una acción**, es necesario que se tomen por verdadero todos los predicados de la **precondición**. Lo normal, es que **después aplicar una acción** existan una serie de **efectos**, es decir, se instancien otro conjunto de predicados que darán como resultado un nuevo estado.

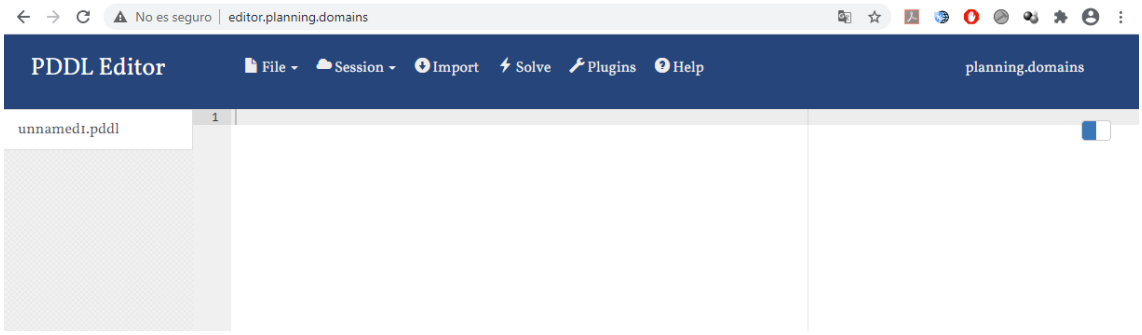
Un resumen de la sintaxis básica se puede encontrar en el documento "The STRIPS Subset of PDDL for the Learning Track of IPC-08" del Campus Virtual.

¿Qué es un planificador?

El planificador es el encargado de definir la estrategia de control: decide cuál será el siguiente nodo del espacio de búsqueda a explorar, es decir, decide el siguiente operador que se aplica.

Uno de los planificadores más famosos es el Metric-FF, que implementa una estrategia de control basada en primero ejecutar un algoritmo de búsqueda local (Hill Climbing) y si no encuentra una solución, ejecuta un weighted-A*, por defecto, f = g + 5*h. Más información en: https://fai.cs.uni-saarland.de/hoffmann/metric-ff.html.

Alternativamente, existen planificadores *online*. Un ejemplo es el siguiente: http://editor.planning.domains/



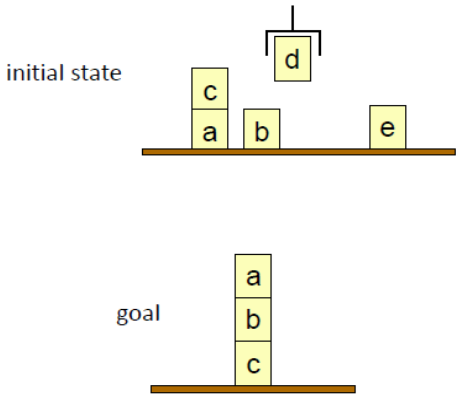
Por comodidad, utilizaremos el planificador online. Se puede encontrar más información sobre el planificador en http://solver.planning.domains/. Además, es posible instalar plugins o cargar los ficheros .pddl de una manera muy sencilla.

2.- Problemas de planificación

Mundo de Bloques

El mundo de los bloques es uno de los problemas de planificación más famosos de la inteligencia artificial. El objetivo es construir una o más pilas verticales de bloques. En la versión clásica del problema, solo se puede mover un bloque a la vez: se puede colocar sobre la mesa o encima de otro bloque. Debido a esto, cualquier bloque que se sitúe debajo de otro bloque no se puede mover. Además, solo hay una pinza y la mesa no tiene restricciones de espacio, es decir, tiene huecos infinitos donde soltar los bloques.

Un ejemplo de estado inicial y final se puede observar en la siguiente imagen:



El estado inicial y objetivo describen como están colocados los bloques, y si la pinza está vacía o no. En este caso, los objetos son únicamente los bloques, mientras que la pinza y la mesa no es necesario que se modelen como objetos. Únicamente es necesario incluir un predicado que especifique si la pinza está libre o no.

Los objetos y predicados para modelar este problema se resumen en la siguiente imagen:

Constant symbols:
- The blocks: a, b, c, d, e

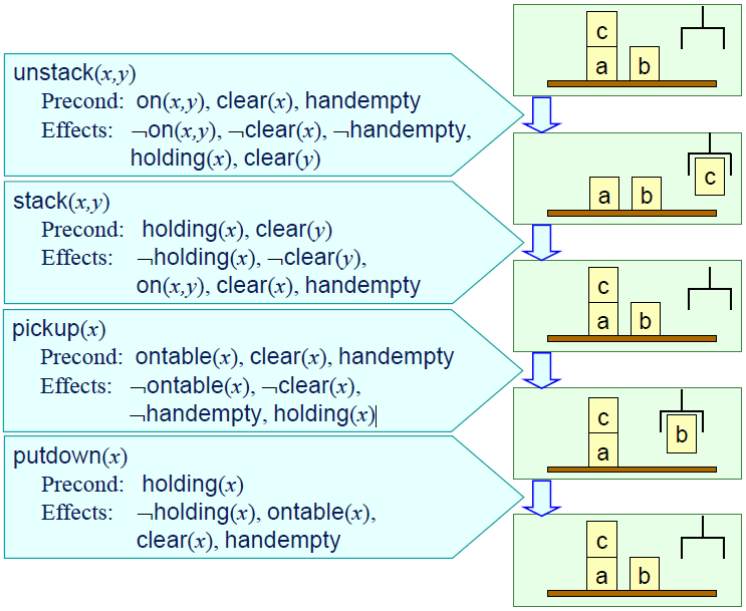
Predicates:
- ontable(*x*) - block *x* is on the table
- on(*x*,*y*) - block *x* is on block *y*
- clear(*x*) - block *x* has nothing on it
- holding(*x*) - the robot hand is holding block *x*
- handempty - the robot hand isn't holding anything

A través de este conjunto de predicados es sencillo representar el conjunto de predicados instanciados que representan el estado inicial:

(encima_mesa a) (not(sin_nada_encima a)) (encima_mesa b) (encima_bloque c a) (encima_mesa e) (sin_nada_encima b) (sin_nada_encima c) (sin_nada_encima e) (not(brazo_libre)) (sujeto d))

Se asume la hipótesis de mundo cerrado, esto implica que no es necesario indicar los predicados que son falsos.

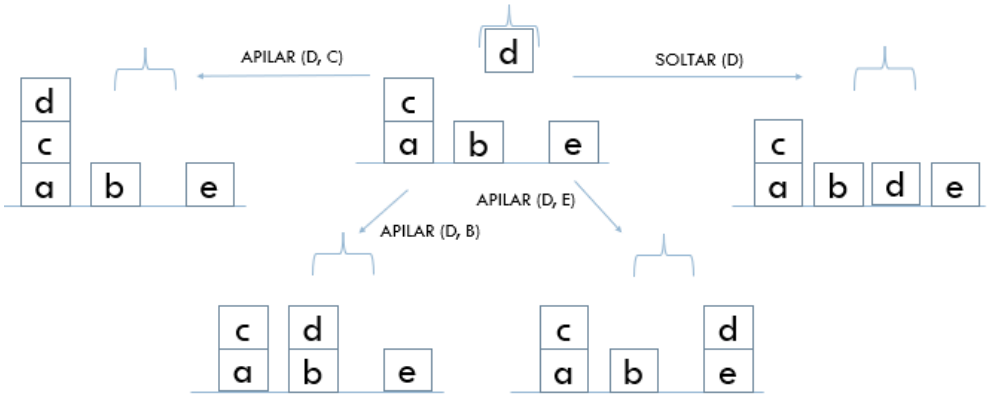
Por otro lado, los operadores, es decir, los movimientos que podemos hacer para mover los bloques son los de la siguiente imagen:



En esta versión del problema (dominio) los bloques se mueven utilizando una sola pinza, que debe estar vacía para poder coger un bloque. Además, para coger un bloque, este no puede tener otro bloque encima y la pinza debe estar libre. Todas estas restricciones se implementan en las precondiciones y efectos de las acciones.

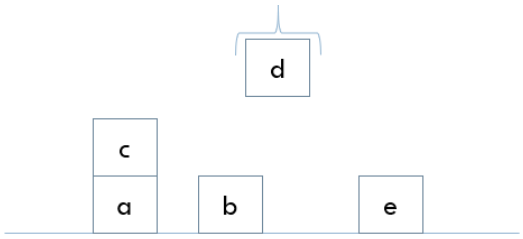
Un ejemplo del espacio de búsqueda que se genera a partir del estado inicial se observa en la siguiente imagen. Este de búsqueda viene definido por todas las posibles acciones que es

posible llevar a cabo, en este caso 4 acciones: APILAR(D,C), APILAR(D,B), APILAR(D,E) o SOLTAR(D).

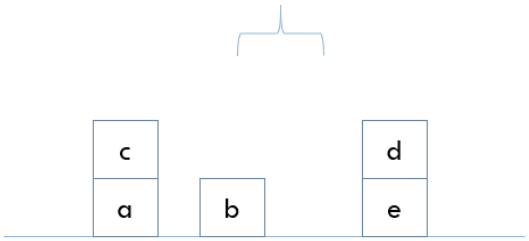


El plan, es decir, la solución, viene dada por el conjunto de operadores utilizados para llegar desde el estado inicial al final. Un ejemplo de plan sería el siguiente:

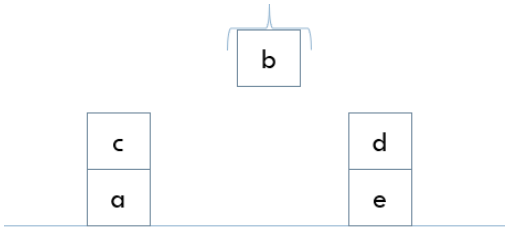
Estado inicial:



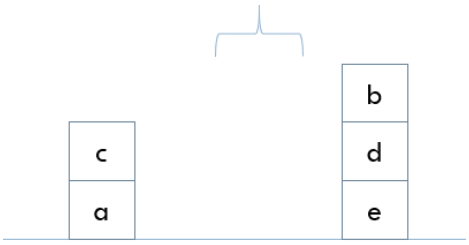
APILAR (D, E)



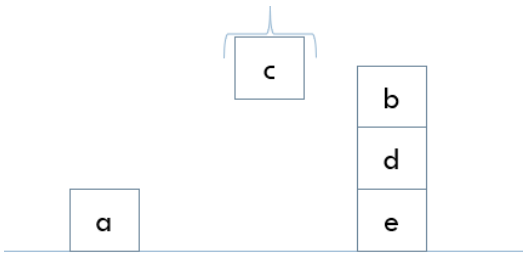
COGER (B)



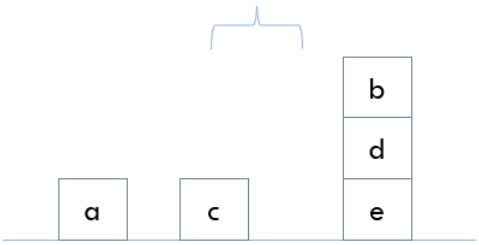
APILAR (B, D)



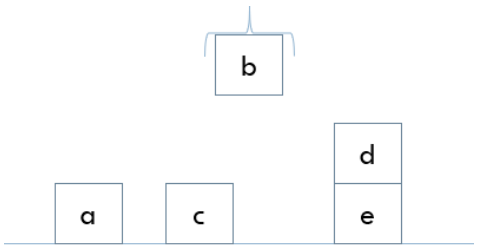
DESAPILAR (C, A)



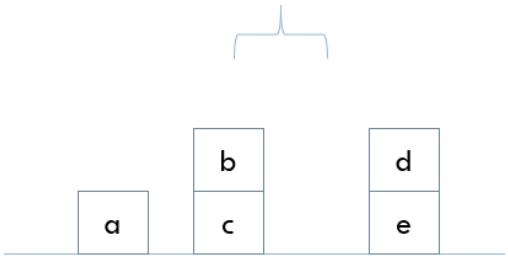
SOLTAR (C)



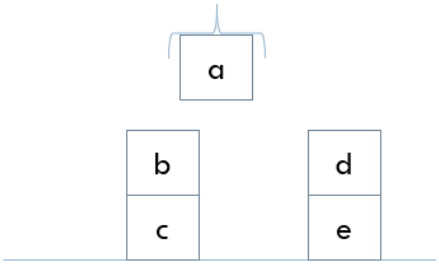
DESAPILAR (B, D)



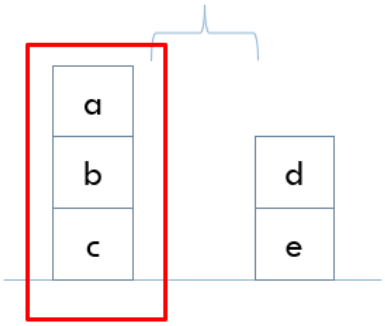
APILAR (B, C)



COGER (A)



APILAR (A, B)



Como se observa, el estado tras aplicar APILAR (A, B) cumple con predicados instanciados en el estado objetivo: (:goal (and (encima_bloque a b) (encima_bloque b c))). Por lo tanto, la solución sería el plan formado por las acciones: APILAR (D, E), COGER (B), APILAR (B, D), DESAPILAR (C, A), SOLTAR (C), DESAPILAR (B, D), APILAR (B, C), COGER (A), APILAR (A, B)

Para este caso, los ficheros de dominio y problema que codifican las imágenes anteriores serían los siguientes:

Dominio: domain01.pddl

```
(define (domain blocksword)
(:predicates
    (sin_nada_encima ?x)
    (encima_mesa ?x)
    (brazo_libre)
    (sujeto ?x)
    (encima_bloque ?x ?y)
)
(:action coger
    :parameters (?ob)
    :precondition (and (sin_nada_encima ?ob)(encima_mesa ?ob)(brazo_libre))
    :effect (and (sujeto ?ob) (not (sin_nada_encima ?ob)) (not (encima_mesa ?ob))(not (brazo_libre)))
)
(:action soltar
    :parameters (?ob)
    :precondition (and (sujeto ?ob))
    :effect (and (sin_nada_encima ?ob) (brazo_libre) (encima_mesa ?ob) (not (sujeto ?ob)))
)
(:action apilar
    :parameters (?ob ?underob)
    :precondition (and (sin_nada_encima ?underob)(sujeto ?ob))
    :effect (and (sin_nada_encima ?ob) (brazo_libre) (encima_bloque ?ob ?underob) (not (sujeto ?ob))(not (sin_nada_encima ?underob)))
)
(:action desapilar
    :parameters (?ob ?underob)
    :precondition (and (encima_bloque ?ob ?underob) (sin_nada_encima ?ob) (brazo_libre))
    :effect (and (sujeto ?ob) (sin_nada_encima ?underob) (not (encima_bloque ?ob ?underob)) (not (sin_nada_encima ?ob)) (not (brazo_libre))))
)
```

Como se observa, en este fichero se deberán especificar la lista de predicados (clear, on-table, holding, on, hand, block), así como la lista de operadores/acciones (pickup, putdown, stack, unstack). Cada una de las acciones está formada por una serie de parámetros, precondiciones y efectos.

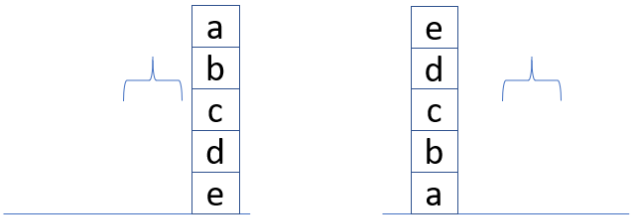
Problema: p01.pddl

```
(define (problem p01)
    (:domain blocksword)

    (:objects a b c d e)

    (:init (encima_mesa a) (encima_mesa b) (encima_bloque c a) (encima_mesa e) (sin_nada_encima b) (sin_nada_encima c) (sin_nada_encima e) (sujeto d))

    (:goal (and (encima_bloque a b) (encima_bloque b c) (sin_nada_encima a) (encima_mesa c)))
)
```

3.- Ejercicios Mundo de Bloques

1. Utiliza el planificador online (http://editor.planning.domains/) para resolver el problema p01.pddl y el dominio domain01.pddl. ¿Cuál es el plan calculado?

2. Codifica una instancia del problema tal como se muestra en la siguiente figura:



Estado inicial Estado final

3. A partir del dominio domain01.pddl, codifica un nuevo modelo de dominio donde no se defina el brazo. Guarda el modelo como domain02.pddl. ¿Qué acciones ya no son necesarias? ¿Qué acciones modificarías?

4. A partir de la instancia p01.pddl define una nueva instancia, p03.pddl, aplicable sobre el dominio domain02.pddl definido en 4). ¿Qué cambios se realizaron?

4.- Ejercicios extra

Como ejercicio evaluable se deberá realizar uno de los siguientes ejercicios:

1. Codifica un modelo y problema de ejemplo para el problema de las Torres de Hanói. El rompecabezas de Las Torres de Hanói tiene como objetivo mover la pila de fichas circulares del poste izquierdo al poste derecho, desplazando los discos individualmente entre los postes (es decir, solamente se puede mover un disco si los demás están en postes), respetando siempre el orden de menor a mayor, es decir, un disco de menor tamaño siempre debe estar encima de uno mayor y finalmente siempre desplazando los discos que no tengan otro encima.

2. Codifica un dominio y problema que permita resolver el 8-puzzle.

3. Codifica un dominio y problema que permita resolver las N-reinas.

4. A partir del documento “The STRIPS Subset of PDDL for the Learning Track of IPC-08” del Campus Virtual extiende el dominio domain01.pddl del Mundo de Bloques para permitir el uso de varios brazos. Además, define una nueva instancia donde, al menos, se tengan dos brazos.

Además del código, se deberá entregar un documento explicativo (pdf de máximo una página) donde se indiquen las partes más relevantes del código y se incluya una captura de pantalla del plan resuelto.