Prácticas con PDDL. Curso 2024-2025. Sesión 1. Introdución al Planning.

Ramiro Varela, Francisco J. Gil-Gala. Jesús Quesada. Sistemas Inteligentes. Grado en Ingeniería Informática. EII. Universidad de Oviedo. Campus de los Catalanes. Oviedo

Resumen. Esta sesión trata sobre el uso del estándar PDDL y su aplicación para resolver el problema de Mundo de Bloques así como variaciones al dominio.

Palabras claves: Planificación, heurísticos, Mundo de Bloques

1 Introducción a PDDL

1.1 ¿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 de aplicar una acción** existan una serie de **efectos**, es decir, se instancien otro conjunto de predicados que darán como resultado un nuevo estado.

1.2 ¿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.

Alternativamente, existen planificadores online. Un ejemplo es el siguiente: http://editor.planning.domains/. Por comodidad, utilizaremos el planificador online en esta sesión de laboratorio. 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 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 Ilustración 1 :

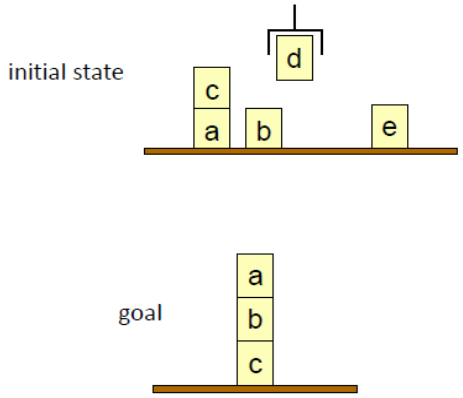


Ilustración 1 Ejemplo de estado inicial y objetivo en una instancia del Mundo de Bloques

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 Ilustración 2 :

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

Ilustración 2 Objetos y Predicados utilizados para modelar la instancia

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))

Por otro lado, los operadores, es decir, los movimientos que podemos hacer para mover los bloques son los de la Ilustración 3 :

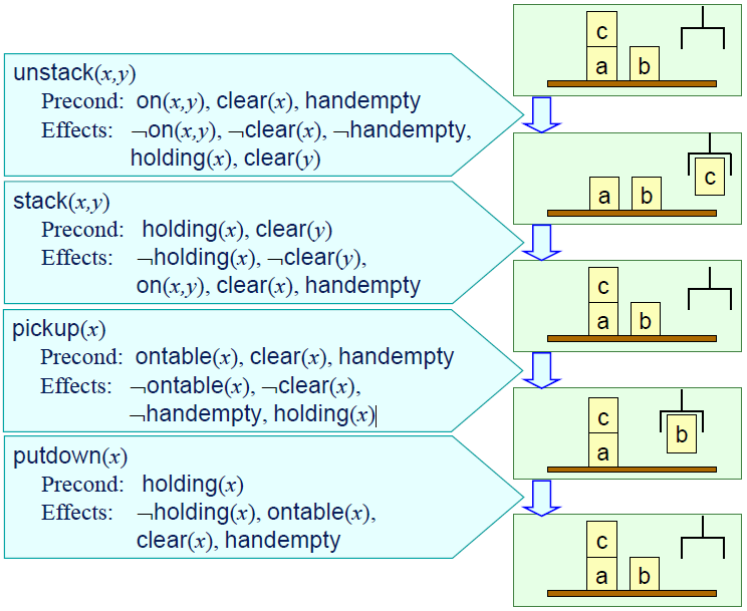


Ilustración 3 Operadores definidos en el dominio

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 Ilustración 4 . Este espacio 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).

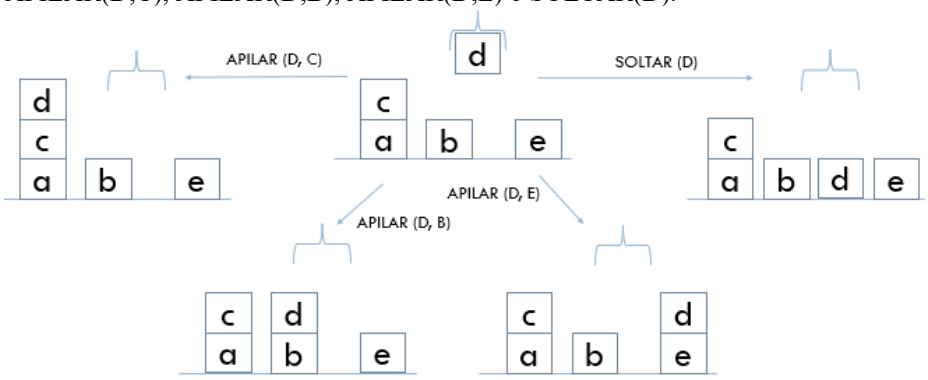
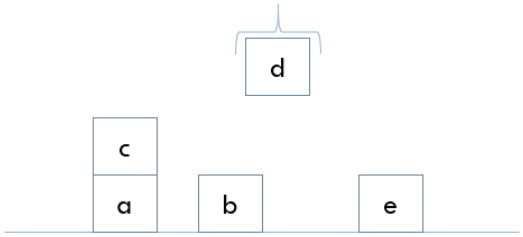


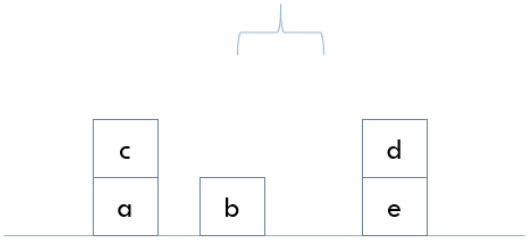
Ilustración 4 Ejemplo de acciones posibles a partir del estado inicial

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:

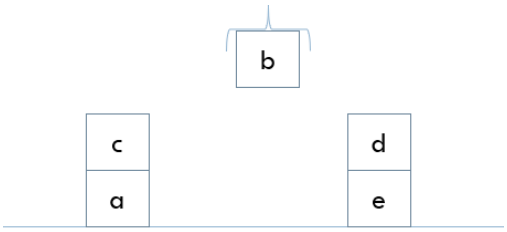
Estado inicial:



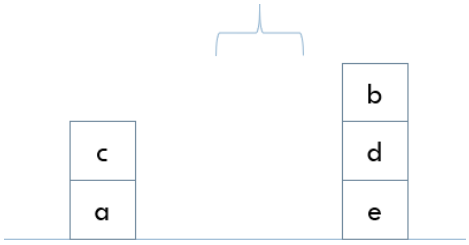
APILAR (D, E)



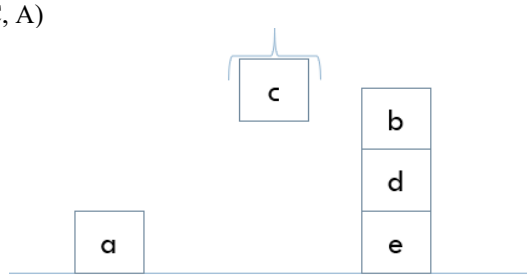
COGER (B)



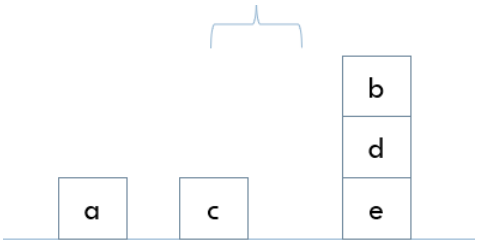
APILAR (B, D)



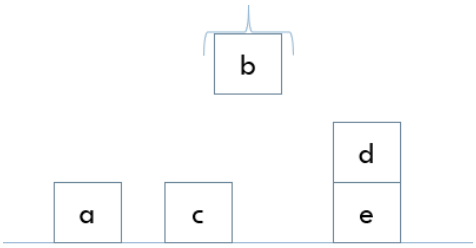
DESAPILAR (C, A)



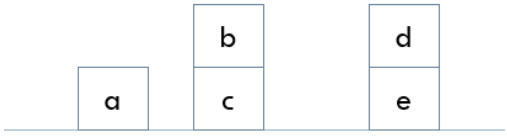
SOLTAR (C)



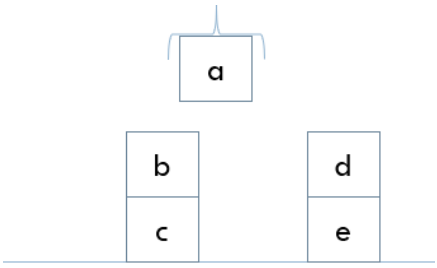
DESAPILAR (B, D)



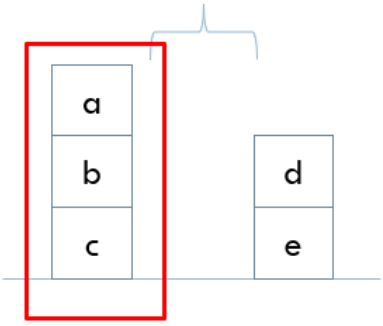
APILAR (B, C)



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)

3 Dominio y Problema utilizados

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_li-
     bre))
     :effect (and (sujeto ?ob) (sin_nada_encima ?underob) (not (encima_bloque ?ob ?unde-
     rob)) (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.

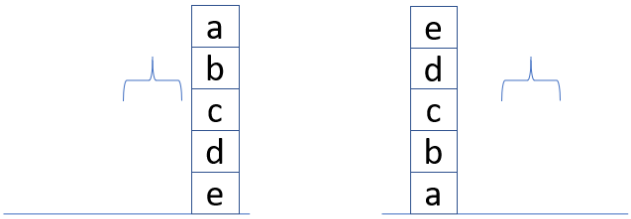
Problema: p01.pddl

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

    (:objects a b c d e)

    (:init (encima_mesa a) (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))

    (:goal (and (encima_bloque a b) (encima_bloque b c) (sin_nada_encima a) (en-
    cima_mesa c)))
)
```

4 Ejercicios para esta práctica

Ejercicio 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?

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



Ejercicio 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? Además, A partir de la instancia `p01.pddl` define una nueva instancia, `p03.pddl`, aplicable sobre el dominio `domain02.pddl` ¿Qué cambios se realizaron?

Ejercicio 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, lo llamaremos `domain03.pddl`. Además, define una nueva instancia donde, al menos, se tengan dos brazos.

Bibliografía

1. Russell, S. and Norvig P. Artificial Intelligence, A Modern Approach, Prentice All, New Jersey, 4th ed. 2021.
2. Nilsson, N., Principles of Artificial Intelligence, Tioga, Palo Alto, CA, 1980.
3. Fox, M. and Long, D., PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains, Journal of Artificial Intelligence Research, AI Access Foundation, 2003. doi: 10.1613/jair.1129
4. Alan Fern, The STRIPS Subset of PDDL for the Learning Track of IPC-08, International Conference on Automated Planning and Scheduling, 2008.
5. Ghallab, M., Knoblock, C., Wilkins, D., Barrett, A., Christianson, D., Friedman, M., Kwok, C., Golden, K., Penberthy, S., Smith, D., Sun, Y., and Weld, D. PDDL - The Planning Domain Definition Language. 1998.