Razonamiento y Planificación Automática		

Planificación de STRIPS

ÍNDICE:

ENUNCIADO:

Exprese el siguiente escenario en la representación tipo STRIPS:

En una habitación hay un mono, una caja y un plátano, tal como indica la ***Figura 1*** (situación inicial). El objetivo del mono es tener el plátano. El mono puede:

- **Ir** de una posición a otra.
- **Empujar** la caja de una posición a otra, si está en la misma posición que ella y no está sobre ella.
- **Subirse** a la caja si está en la misma posición que ella.
- **Coger** el plátano si está en encima de la caja.

Se deberá diseñar un programa en Python en el que se obtenga un plan de acción en modo texto, a través de la consola, que, para una configuración de mono, caja y plátano, nos devuelva las acciones. La salida de programa deberá ser una lista ordenada de operadores a aplicar.

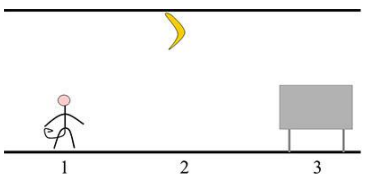


Figura 1

INTRODUCCIÓN:

Mediante **STRIPS** definiremos un **estado** no por su nombre, sino por el **conjunto** de sus **propiedades**. Estas propiedades, se expresan por medio de **valores *booleanos***, es decir, de existencia o no existencia en el estado actual. Esto nos permite por ejemplo usar los operadores AND, OR y máscaras de bits para determinar si las propiedades están presentes en el estado (PC), eliminar (E) algunas o añadirlas (A).

PROPIEDADES DE LOS ESTADOS:

Para el problema del **mono**, tendríamos las **siguientes propiedades** relevantes para caracterizar los estados del mundo: **M**=Mono, **P**=Plátano y **C**=Caja

Está(M,1)	Está(M,2)	Está(M,3)	Está(C,1)	Está(C,2)	Está(C,3)	Está(P,1)	Está(P,2)	Está(P,3)	Libre(C)	Sobre(M,C)	Libre(P)	Tiene(M,P)

Por otro lado, modelamos las acciones del mono usando operadores **STRIPS**. Estas acciones solo podrán aplicarse a un estado cuando se cumplan el conjunto de propiedades especificadas en las **precondiciones** del estado, **eliminado** las indicadas en el conjunto E y **añadiendo** las indicadas en el conjunto A.

ESTADO INICIAL:

Se proporciona en un vector (M,P,C) de valores [1-3], donde se indica en qué posición está el mono, el plátano y la caja (siempre asumiendo que el mono empieza en el suelo).

Por ejemplo:

Suponiendo que el vector de entrada es [1,2,3] que corresponde al conjunto de propiedades asociadas al estado de la Figura 1. El estado inicial estaría formado por las propiedades que tienen un 1:

1	0	0	0	0	1	0	1	0	1	0	1	0
Está(M,1)	Está(M,2)	Está(M,3)	Está(C,1)	Está(C,2)	Está(C,3)	Está(P,1)	Está(P,2)	Está(P,3)	Libre(C)	Sobre(M,C)	Libre(P)	Tiene(M,P)

OPERADORES STRIPS:

Modelamos con operadores STRIPS las **14** posibles acciones del mono. Para mayor claridad se describen textualmente algunas de ellas.

Ir(M,1,2) op=0 ***Con esta acción el mono se moverá por el suelo desde el punto 1 al 2.***

PC[0]=int('1000000000000' , 2) # PC =Está(M,1) *Precondición es que el mono esté en el punto 1.*

E[0]=int('0111111111111' , 2) # E = Está(M,1) *Se eliminará la propiedad Mono está en 1.*

A[0]=int('0100000000000' , 2) # A = Está(M,2) *Se añadirá la propiedad Mono está en 2.*

Ir(M,2,3) op=1 ***Con esta acción el mono se moverá por el suelo desde el punto 2 al 3.***

PC[1]=int('0100000000000' , 2) # PC =Está(M,2)

E[1]=int('1011111111111' , 2) # E = Está(M,2)

A[1]=int('0010000000000' , 2) # A = Está(M,3)

Ir(M,3,2) op=2 ***Con esta acción el mono se moverá por el suelo desde el punto 3 al 2.***

PC[2]=int('0010000000000' , 2) # PC =Está(M,3)

E[2]=int('1101111111111' , 2) # E = Está(M,3)

A[2]=int('0100000000000' , 2) # A = Está(M,2)

Ir(M,2,1) op=3 ***Con esta acción el mono se moverá por el suelo desde el punto 2 al 1.***

PC[3]=int('0100000000000' , 2) # PC =Está(M,2)

E[3]=int('1011111111111' , 2) # E = Está(M,2)

A[3]=int('1000000000000' , 2) # A = Está(M,1)

Empujar(M,C,1,2) op=4 ***Con esta acción el mono empujará la caja desde el punto 1 al 2.***

PC[4]=int('1001000001000' , 2) # PC =Está(M,1),Está(C,1),libre(C)

E[4]=int('0110111111111' , 2) # E = Está(M,1),Está(C,1)

A[4]=int('0100100000000' , 2) # A = Está(M,2),Está(C,2)

Empujar(M,C,2,3) op=5 ***Con esta acción el mono empujará la caja desde el punto 2 al 3.***

PC[5]=int('0100100001000' , 2) # PC =Está(M,2),Está(C,2),libre(C)

E[5]=int('1011011111111' , 2) # E = Está(M,2),Está(C,2)

A[5]=int('0010010000000' , 2) # A = Está(M,3),Está(C,3)

Empujar(M,C,2,1) op=6 ***Con esta acción el mono empujará la caja desde el punto 2 al 1.***

PC[6]=int('0100100001000' , 2) # PC =Está(M,2),Está(C,2),libre(C)

E[6]=int('1011011111111' , 2) # E = Está(M,2),Está(C,2)

A[6]=int('1001000000000' , 2) # A = Está(M,1),Está(C,1)

Empujar(M,C,3,2) op=7 ***Con esta acción el mono empujará la caja desde el punto 3 al 2.***

PC[7]=int('0010010001000' , 2) # PC =Está(M,3),Está(C,3),libre(C)

E[7]=int('1101101111111' , 2) # E = Está(M,3),Está(C,3)

A[7]=int('0100100000000' , 2) # A = Está(M,2),Está(C,2)

Subir(M,C,1) op=8 ***Con esta acción el mono se subirá a la caja en el punto 1.***

PC[8]=int('1001000001000' , 2) # PC =Está(M,1),Está(C,1),libre(C)

E[8]=int('0111111110111' , 2) # E = libre(C),Está(M,1)

A[8]=int('0000000000100' , 2) # A = Sobre(M,C)

Subir(M,C,2) op=9 ***Con esta acción el mono se subirá a la caja en el punto 2.***

PC[9]=int('0100100001000' , 2) # PC =Está(M,2),Está(C,2),libre(C)

E[9]=int('1011111110111' , 2) # E = libre(C),Está(M,2)

A[9]=int('0000000000100' , 2) # A = Sobre(M,C)

Subir(M,C,3) op=10 ***Con esta acción el mono se subirá a la caja en el punto 3.***

PC[10]=int('0010010001000' , 2) # PC =Está(M,3),Está(C,3),libre(C)

E[10]=int('1101111110111' , 2) # E = libre(C),Está(M,3)

A[10]=int('0000000000100' , 2) # A = Sobre(M,C)

Coger(M,P,1) op=11 ***Con esta acción el mono cogerá el plátano en el punto 1.***

PC[11]=int('0001001000110' , 2) # PC =Está(C,1),Esta(P,1),Sobre(M,C),libre(P)

E[11]=int('1111111111101' , 2) # E = libre(P)

A[11]=int('0000000000001' , 2) # A = Tiene(M,P)

Coger(M,P,2) op=12 ***Con esta acción el mono cogerá el plátano en el punto 2.***

PC[12]=int('0000100100110' , 2) # PC =Está(C,2),Esta(P,2),Sobre(M,C),libre(P)

E[12]=int('1111111111101' , 2) # E = libre(P)

A[12]=int('0000000000001' , 2) # A = Tiene(M,P)

Coger(M,P,3) op=13 ***Con esta acción el mono cogerá el plátano en el punto 3.***

PC[13]=int('0000010010110' , 2) # PC =Está(C,3),Esta(P,3),Sobre(M,C),libre(P)

E[13]=int('1111111111101' , 2) # E = libre(P)

A[13]=int('0000000000001' , 2) # A = Tiene(M,P)

EJEMPLO DE APLICACIÓN DE UN OPERADOR STRIPS:

Supongamos que nos encontramos en el estado determinado por las siguientes propiedades:

$$\{\text{Está}(M,3),\text{Está}(C,3),\text{Está}(P,2),\text{Libre}(C),\text{Libre}(P)\}$$

0	0	1	0	0	1	0	1	0	1	0	1	0
Está(M,1)	Está(M,2)	Está(M,3)	Está(C,1)	Está(C,2)	Está(C,3)	Está(P,1)	Está(P,2)	Está(P,3)	Libre(C)	Sobre(M,C)	Libre(P)	Tiene(M,P)

Y queremos aplicar el siguiente **operador**:

Empujar(M,C,3,2) op=7 ***Con esta acción el mono empujará la caja desde el punto 3 al 2.***

PC[7]=int('0010010001000' , 2) # PC =Está(M,3),Está(C,3),libre(C)

E[7]=int('1101101111111' , 2) # E = Está(M,3),Está(C,3)

A[7]=int('0100100000000' , 2) # A = Está(M,2),Está(C,2)

a) **PRECONDICIONES**: En primer lugar hay que comprobar que se cumplen las **Precondiciones**:

Esto puede hacerse con un **AND** entre el **estado** y la **máscara** que tiene todo a **0** excepto en las propiedades contenidas en las **precondiciones** del operador que valdrán **1**, debiéndose satisfacer todas, por lo que el resultado debe ser igual a la máscara.

$$PC=\{\text{Está}(M,3),\text{Está}(C,3),\text{libre}(C)\}$$

	Está(M,1)	Está(M,2)	Está(M,3)	Está(C,1)	Está(C,2)	Está(C,3)	Está(P,1)	Está(P,2)	Está(P,3)	Libre(C)	Sobre(M,C)	Libre(P)	Tiene(M,P)
ESTADO	0	0	1	0	0	1	0	1	0	1	0	1	0
MÁSCARA Precondiciones	0	0	1	0	0	1	0	0	0	1	0	0	0
AND	0	0	1	0	0	1	0	0	0	1	0	0	0

En el ejemplo vemos que el resultado es igual a la máscara con lo que se cumplen todas la precondiciones para aplicar el operador al estado (en caso contrario habría que probar con otro operador).

Asumiendo que se cumplen las precondiciones pasaríamos a:

b) **ELIMINACIÓN:** Esto puede hacerse con un **AND** entre el **estado** y la **máscara** que tiene todo a **1** excepto en las propiedades contenidas en el conjunto de Eliminación **E** que valen **0**.

$$E=\{\text{Está}(M,3),\text{Está}(C,3)\}$$

	Está(M,1)	Está(M,2)	Está(M,3)	Está(C,1)	Está(C,2)	Está(C,3)	Está(P,1)	Está(P,2)	Está(P,3)	Libre(C)	Sobre(M,C)	Libre(P)	Tiene(M,P)
ESTADO	0	0	1	0	0	1	0	1	0	1	0	1	0
MÁSCARA Eliminación	1	1	0	1	1	0	1	1	1	1	1	1	1
AND	0	0	0	0	0	0	0	1	0	1	0	1	0

c) AÑADIR: Esto puede hacerse con un **OR** entre el **estado** (resultante del paso anterior) y la **máscara** que tiene todo a **0** excepto en las propiedades contenidas en el conjunto de Añadir **A** que valen **1**.

$$A=\{\text{Está}(M,2),\text{Está}(C,2)\}$$

	Está(M,1)	Está(M,2)	Está(M,3)	Está(C,1)	Está(C,2)	Está(C,3)	Está(P,1)	Está(P,2)	Está(P,3)	Libre(C)	Sobre(M,C)	Libre(P)	Tiene(M,P)
ESTADO	0	0	0	0	0	0	0	1	0	1	0	1	0
MÁSCARA Añadir	0	1	0	0	1	0	0	0	0	0	0	0	0
OR	0	1	0	0	1	0	0	1	0	1	0	1	0

Hemos visto que:

Desde el estado: $\{\text{Está}(M,3),\text{Está}(C,3),\text{Está}(P,2),\text{Libre}(C),\text{Libre}(P)\}$

Aplicando el operador: $\text{Empujar}(M,C,3,2)$

Obtenemos el estado: $\{\text{Está}(M,2),\text{Está}(C,2),\text{Está}(P,2),\text{Libre}(C),\text{Libre}(P)\}$

ALGORITMO DE PLANIFICACIÓN:

Asumiendo que el **grado de ramificación** no es grande ya que pese a existir 14 acciones, en cada estado, solo se permiten un subconjunto pequeño de ellas (**5 en el peor de los casos**, cuando la caja y el mono están en el punto 2) y que la **profundidad** necesaria (inteligencia del mono) para alcanzar la solución en **el peor de los casos son 6 acciones** (mono y plátano en extremo opuesto al de la caja), se ha decidido usar un algoritmo de **planificación progresiva**. El árbol de exploración está acotado por 5^6 =15625 estados por lo que no nos enfrentamos a un problema de **explosión combinatoria**.

El método de **planificación progresiva** se ha implementado mediante un algoritmo iterativo (no recursivo) de **exploración en profundidad**, usándose una pila de estados y otra de acciones que en cada momento contienen la rama de exploración.

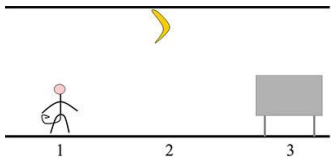
PSEUDOCÓDIGO:

```
top=0 // Cima de la pila
estado[top]=inicial
operaciones[top]=0 // Las operaciones van desde 0 hasta max_operaciones-1
op=0
actual=inicial
MIENTRAS (top>=0) HACER // Mientras queden estados por explorar
    MIENTRAS((op<max_operaciones) Y (top<max_profundidad-1) Y (actual!=meta)) HACER
        siguiente=intentar(actual,op) // Intenta realizar la acción op sobre el estado actual
        SI (siguiente!=-1)ENTONCES  // -1 es null acción no permitida
            SI (NO(encontrado(siguiente))) ENTONCES // Si no hemos pasado por ese estado
                top+=1 // Almacenamos en la pila el estado y la acción
                estado[top]=actual
                operaciones[top]=op
                //Saltamos al siguiente estado
                actual=siguiente
                op=-1 // Para que empiece con la op 0

            FINSI
        FINSI
        op+=1
        SI (int(bin(actual & meta),2)==int(bin(meta),2)) ENTONCES  // actual==meta
            ESCRIBIR("SOLUCIÓN:")
            mostrarSolucion()
        finsi
    FINMIENTRAS
FINMIENTRAS
```

RESULTADOS:

Se ilustra a continuación uno de los resultados obtenidos por el algoritmo a partir de la siguiente configuración inicial:




```
SOLUCIÓN:
Ir(M,1,2) -->Ir(M,2,3) -->Empujar(M,C,3,2) -->Subir(M,C,2) -->Coger(M,P,2)
```

CONCLUSIONES:

Se observa con esta tarea la dificultad de planificar acciones cuando existen muchos estados en el mundo. Usar mecanismos de **planificación regresiva** partiendo del estado meta y generando hacia atrás, estados consistentes que satisfacen **sub-metas,** es una alternativa que permite buscar de manera más eficiente la solución, pero que no nos libra del problema de explosión combinatoria (cuando es grande el grado de ramificación y/o la profundidad de la solución).

Se concluye que enfrentarse a problemas de planificación que requieren un tiempo corto de respuesta es especialmente complejo. Usar la heurística puede mejorar los tiempos de respuesta.