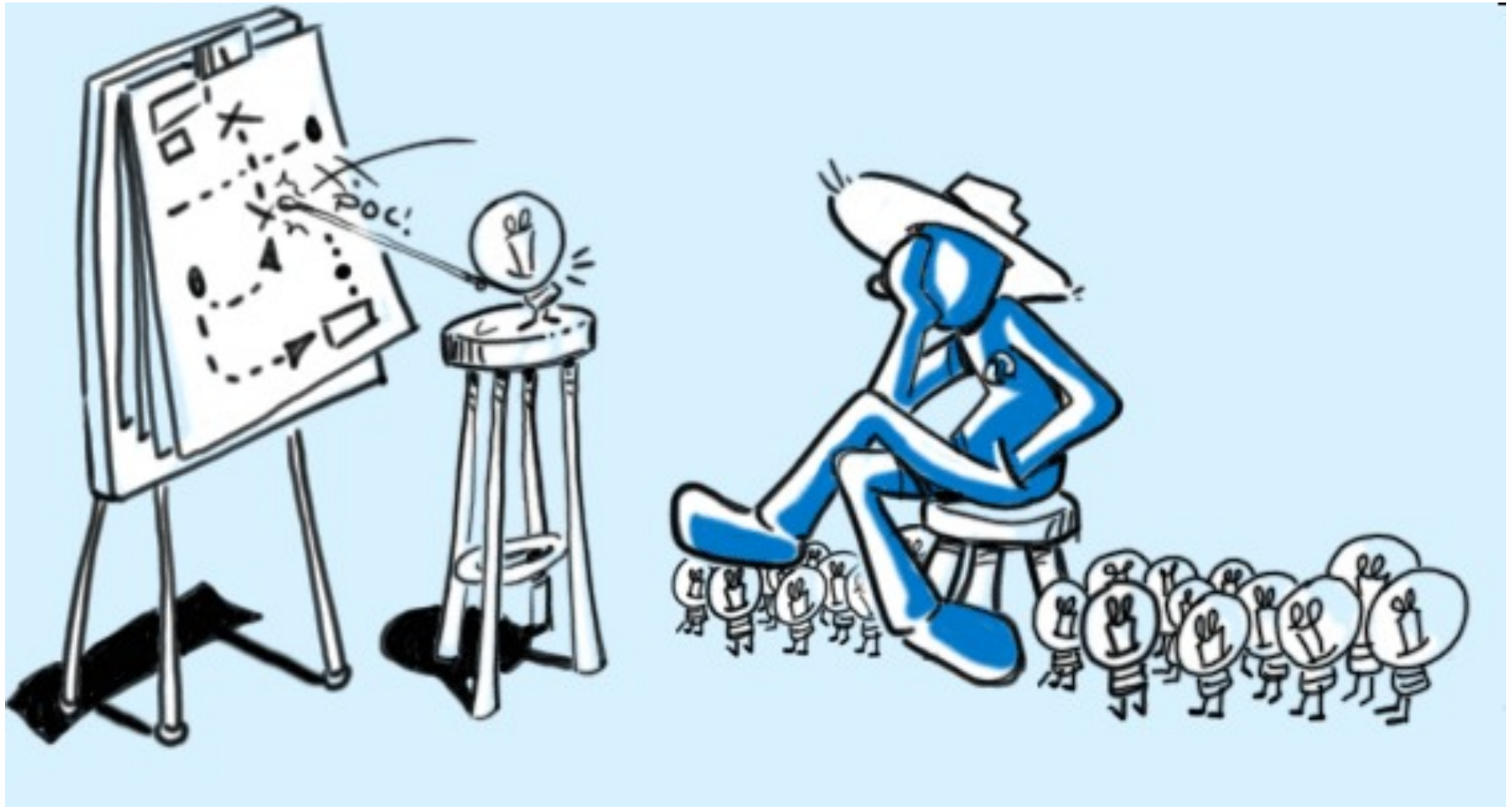


Tema 3: Planificación Inteligente

Sistemas Inteligentes
3 Grado de Ingeniería Informática
Esp. Computación

¿Qué es la planificación?



¿Qué es la planificación?

- Empresarios: planes de la empresa
- Abogados: planes de defensa del cliente
- Industriales: planes de movimiento de robots
- Arquitectos: planes de diseño de edificios
- Informáticos: planes de desarrollo del sistema
- Telecomunicaciones: planes de conexión
- Ejército: planes de ataque/defensa
- Logística de transportes: planes para llevar objetos /sujetos de un sitio a otro
- ...

Planificación

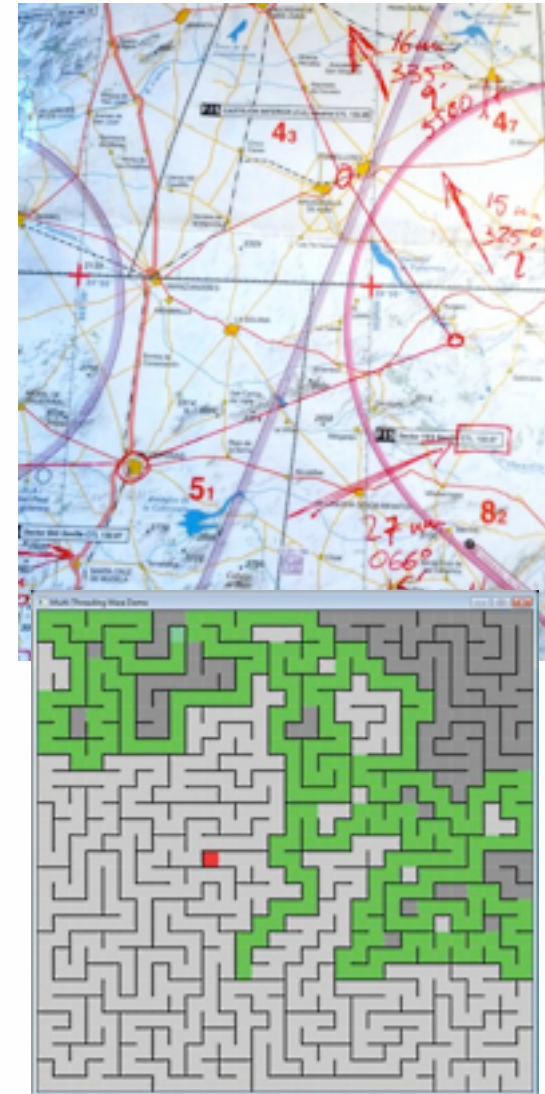
- Los problemas y soluciones que se abordan en planificación tienen aplicaciones directas en gestión de tareas (workflow), control de misiones complejas (espaciales, satélites, militares, etc.), turismo (visitas a ciudades, planificar rutas, ...), procesos de enseñanza/aprendizaje, robótica (planificar caminos), ...

Planificación

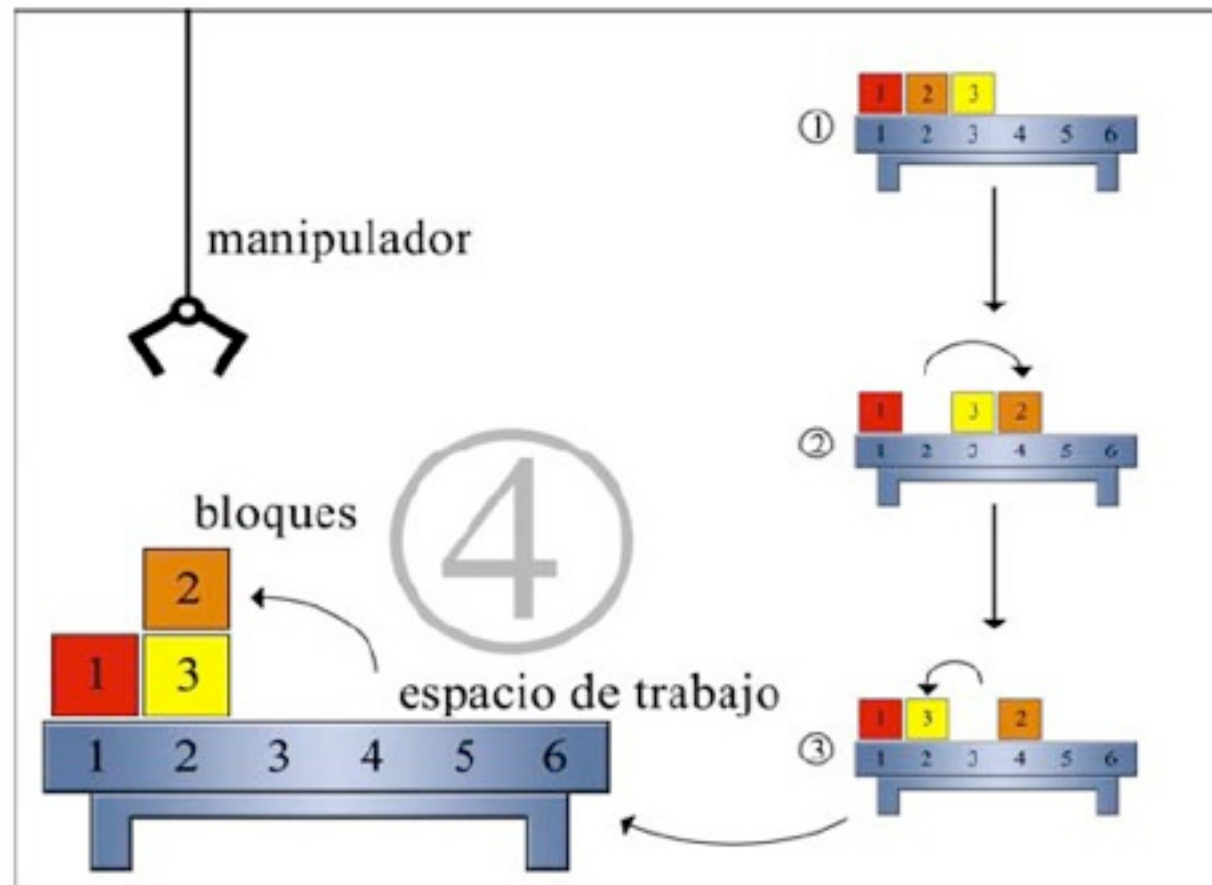
- **Planificación** es el proceso de **búsqueda** y articulación de una **secuencia de acciones** que permite alcanzar un **objetivo**.

Planificación

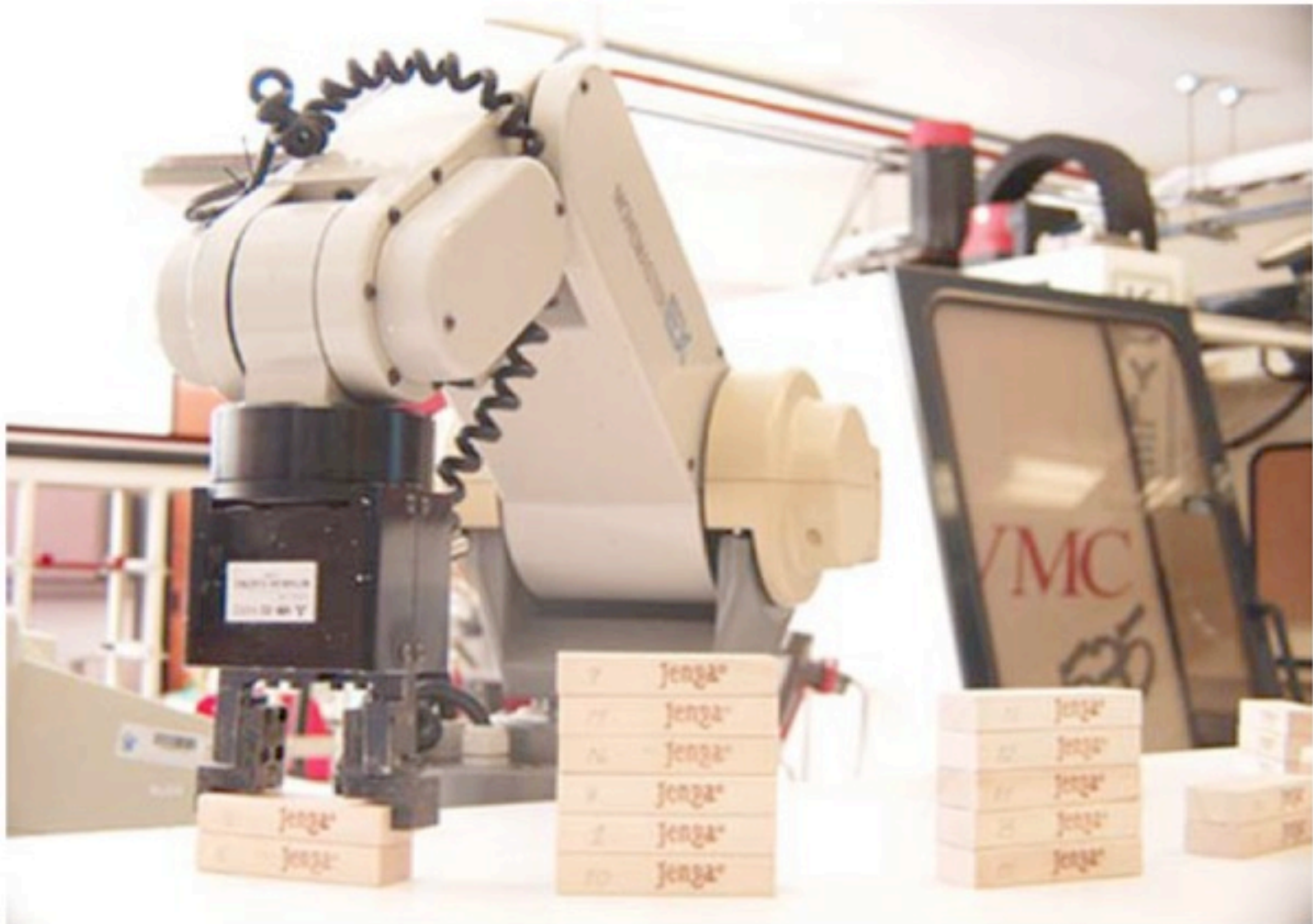
- Por ejemplo, si nuestro objetivo es viajar desde un pueblo perdido de Asturias y queremos llegar a Guatemala la secuencia de acciones serían los distintos transportes que se deben tomar para llegar.
- Otro ejemplo podría ser que tuviéramos un robot en un laberinto y nuestro objetivo fuera sacarle de él; en tal caso, nuestras acciones serían los tramos recorridos en línea recta y los giros dados por el robot.



Una instancia del problema



Una instancia del problema



Planificación

- Para la problemática de la planificación, las licencias libres han brillado por su ausencia. No obstante, todos los algoritmos se distribuyen con su código, lo cual también es cierto en arquitecturas integradas como PRODIGY, muy usadas en planificación.

Tipos de planificación

Previamente se ha introducido la problemática que conlleva la realización de planes. Ahora se especifican, en grados de dificultad creciente, las distintas técnicas de planificación.

- Planificación mediante pila de objetivos: STRIPS
- Planificación de orden parcial
- Descomposición Jerárquica (HTN)
- ...

Planificación STRIPS

- El uso de una pila de objetivos fue una de las primeras técnicas que surgieron para componer objetivos que pudieran interactuar.
- En este apartado se introducen problemas cada vez más complejos, utilizando para su descripción y tratamiento el lenguaje STRIPS y el dominio de los bloques, clásico en robótica y planificación
- se utilizará un dominio de planificación clásico.
- Los operadores utilizados en este dominio están limitados en cuanto a lo que pueden expresar y, por ello, son especialmente adecuados para introducir las mejoras requeridas.

Planificación de orden parcial

- La mayoría de los problemas difíciles provocan interacciones entre los objetivos.
- Los operadores que se utilizan para resolver un subobjetivo pueden interferir en la solución de un subobjetivo anterior. Esto es, se necesita un plan entrelazado en el que se trabaje simultáneamente con múltiples subobjetivos.
- Este tipo de planificación, denominada planificación no lineal es un tipo de planificación de orden parcial, ya que en el plan pueden aparecer pasos no ordenados y ordenados (antes o después).
- Se trabaja mediante fijación de restricciones.

Descomposición Jerárquica

- Se introduce este tipo de planificación para resolver el problema de tratar con planes demasiado extensos en detalles.
- Un primer paso fue la creación de los conocidos macrooperadores de STRIPS, en donde se construían operadores grandes que resumían de forma efectiva la combinación de otros más pequeños. Pero este enfoque sigue siendo insuficiente ya que mantiene las descripciones de los operadores.
- Una alternativa es partir de la idea de descomposición jerárquica: en la que un operador abstracto (ignoraba niveles más específicos) puede ser descompuesto en un grupo de pasos que forman un plan.
- Bajo este enfoque son especialmente conocidos los sistemas ABSTRIPS y NOHA [Sacerdoti, 1974; 1977].

[

]

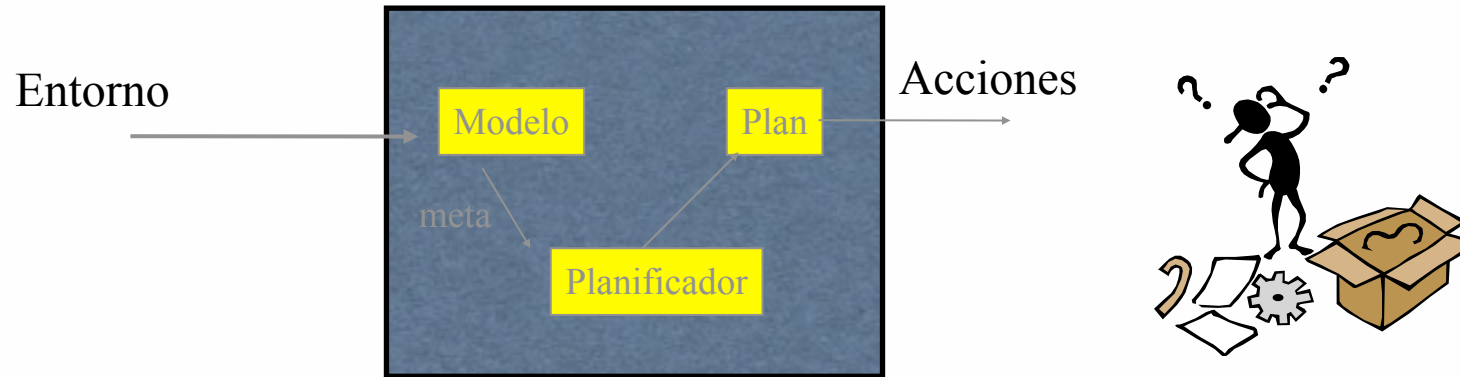
Planificación Clásica

- Los entornos de aplicación:
 - observables,
 - deterministas,
 - finitos,
 - estáticos, y
 - discretos (en tiempo, acciones, objetos y efectos)

Algunas cuestiones difíciles

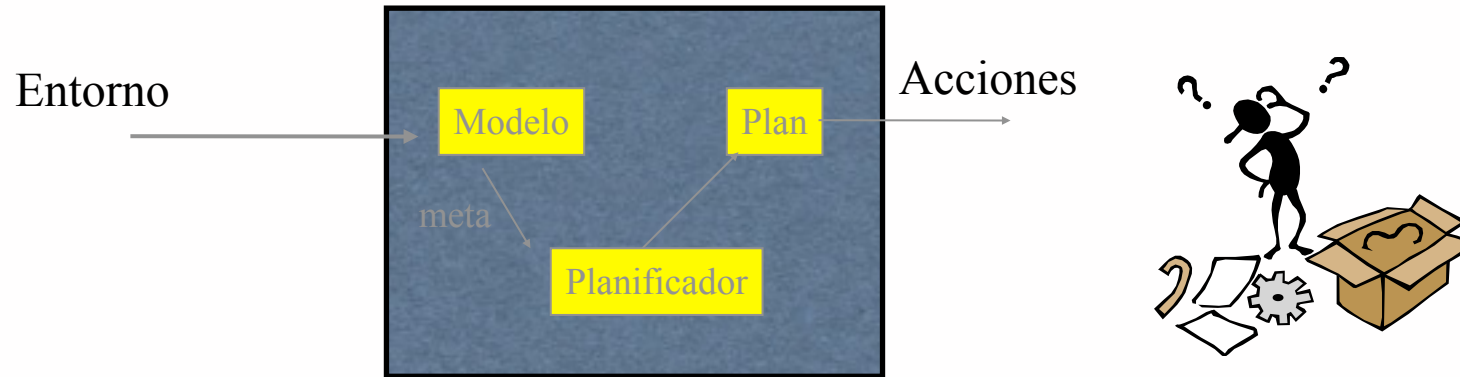
- Nuestra visión del mundo es **incompleta**: racionalidad limitada
- El mundo **cambia** constantemente: dinamismo
- Las acciones **tardan** en ejecutarse: razonamiento temporal
- Nuestras metas son contradictorias: **dependencia** entre metas
- Nuestro modelo del mundo **falla** muchas veces: incertidumbre
- Los planes no siempre son válidos: ejecución y **replanificación**
- No todos los planes son buenos: **calidad**
- Nos **adaptamos** al mundo: **aprendizaje**

Planificación

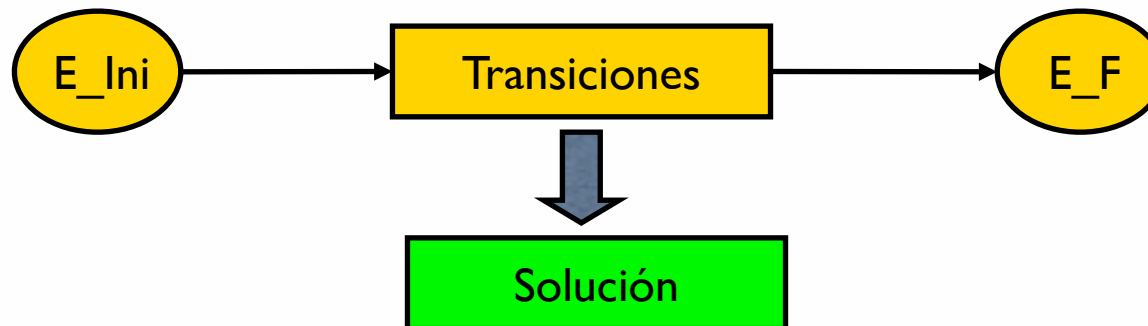


- Se puede ver como una búsqueda en el espacio de estados

Planificación



- Se puede ver como una búsqueda en el espacio de estados



Enfoques

- Como un problema deductivo.
[McCarthy, Berezin, Hoare, Spivey, Millner, etc.]
- Como un problema inductivo.
[Backhouse]
- Como un problema de optimización.
[Colorni, Dorigo, Bay, Harrison, etc.]
- Como un problema de búsqueda.
[Winston, Tate, Warren, Weld, Corkill, Sacerdoti, Lee, McAllester, etc.]

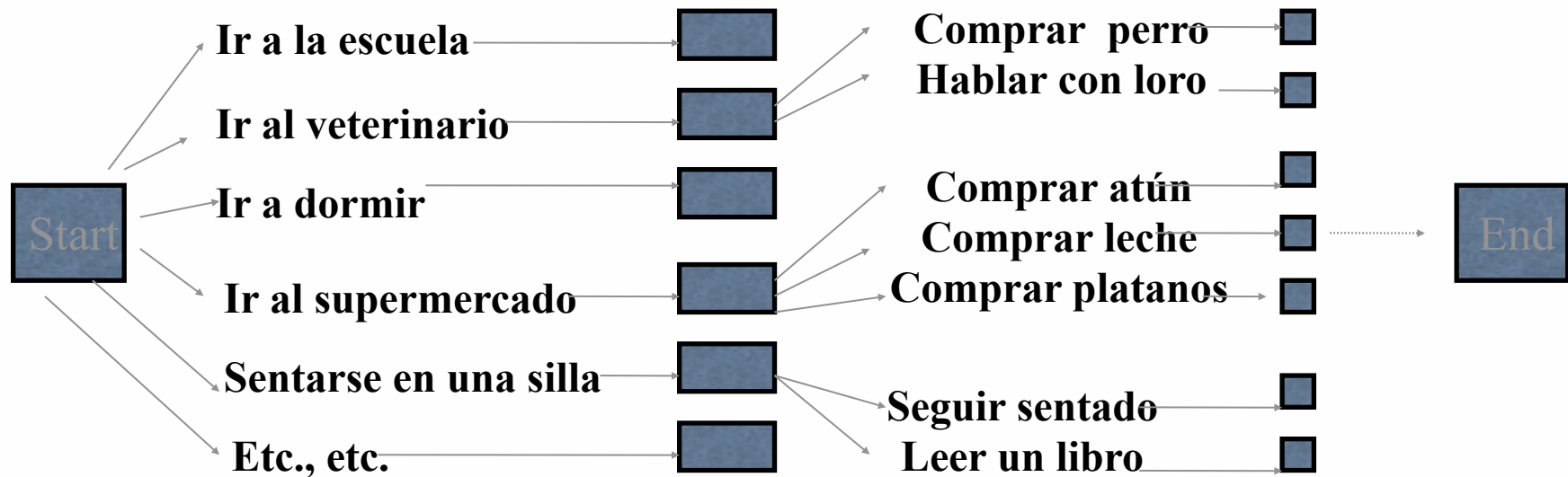
Ejemplo

“Conseguir un litro de leche, plátanos y un taladro de velocidades ajustables”

- Estado inicial: agente en casa, sin leche, sin plátanos y sin taladro
- Conjunto de operaciones: todo lo que pueda hacer
- Heurística: número de cosas sin adquirir

Ejemplo

Ejemplo



Desventajas

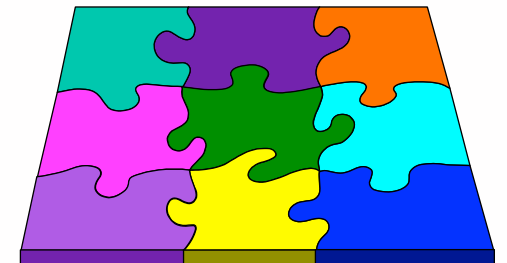
- Demasiadas acciones y estados que considerar
- Solo evaluar estados no permite descartar acciones y no se tiene idea de que intentar después (adivinan)
- Se fuerza al agente a decidir una acción inicial aunque las acciones relevantes no obligan a ningún orden
- No es posible trabajar en partes del problema más fáciles de resolver

Ideas clave

- Clave 1: la representación de estados, objetivos y acciones es abierta.
- Lenguaje formal (ej: lógica de primer orden)
- Estados y metas (objetivos) se representan por conjuntos de sentencias lógicas. ej: en(super), tener(leche)
- Las acciones se representan por descripciones lógicas de precondiciones y efectos
 - ej: Op(ACTION: Comprar(leche),
 - PRECOND: en(supermercado),
 - EFECTO: tener(leche))

Ideas clave

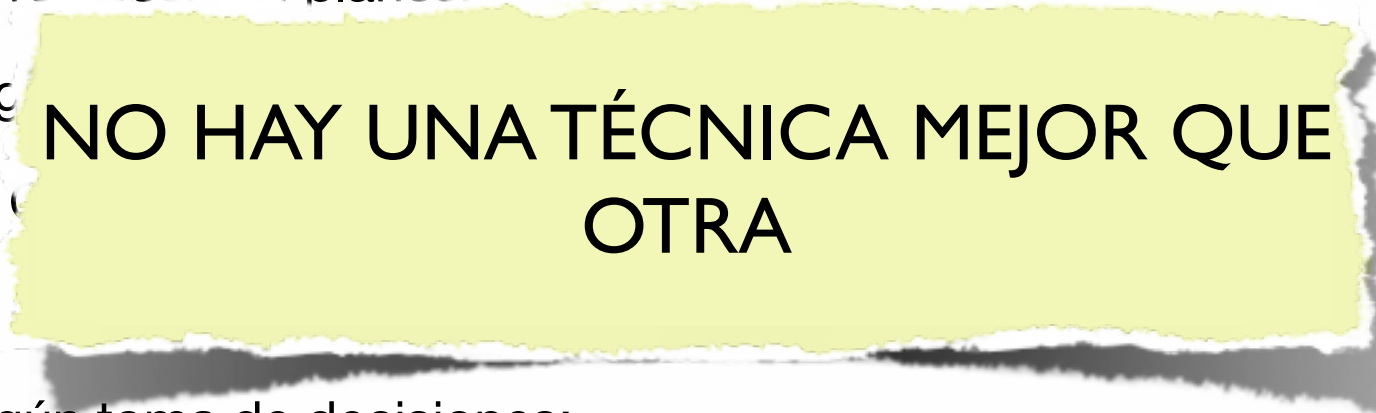
- Clave 2: el planificador es libre de añadir acciones.
 - Añaden acciones donde se necesitan y no necesariamente en una secuencia incremental desde el estado inicial, reduciendo el factor de ramificación
- Clave 3: partes del mundo son independientes entre sí.
 - Pueden usar estrategia “divide y vencerás” cuando partes del problema son independientes. Un Plan se divide en subplanes, un subplan puede dividirse a su vez, y después se combinan soluciones para resolver todo el problema



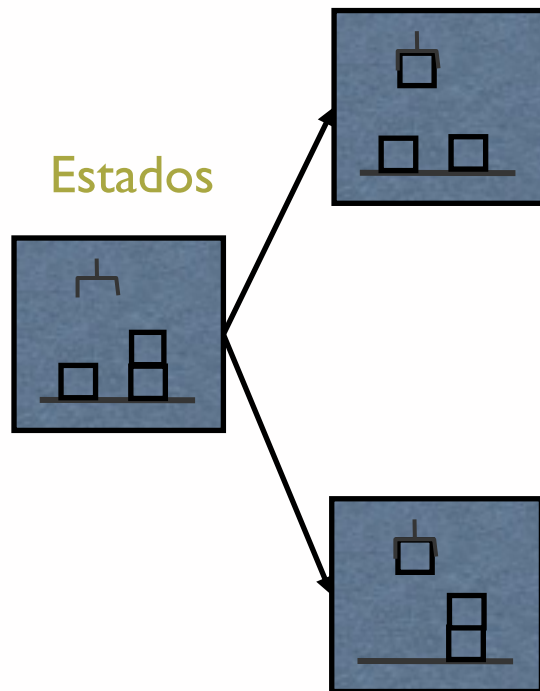
Clasificación

- Según espacio de problemas:
 - Estados (STRIPS, PRODIGY): nodos del árbol representan estados.
 - Planes (NOAH, TWEAK, UCPOP, SNLP, O-PLAN): nodos del árbol representan planes.
- Según plan generado:
 - Orden total, secuencia única de operadores.
 - Orden parcial: expresa múltiples secuencias posibles
- Según toma de decisiones:
 - Compromiso casual: se toman decisiones continuamente.
 - Mínimo compromiso: sólo se toman decisiones cuando se ven forzados.

Clasificación

- Según espacio de problemas:
 - Estados (STRIPS, PRODIGY): nodos del árbol representan estados.
 - Planes (NOAH, TWEAK, UCPOP, SNLP, O-PLAN): nodos del árbol representan planes.
- Según  **NO HAY UNA TÉCNICA MEJOR QUE OTRA**
 -
 -
- Según toma de decisiones:
 - Compromiso casual: se toman decisiones continuamente.
 - Mínimo compromiso: sólo se toman decisiones cuando se ven forzados.

Clasificación



Movimientos en el espacio:

- Modificar el estado del mundo mediante operador

Modelo del tiempo:

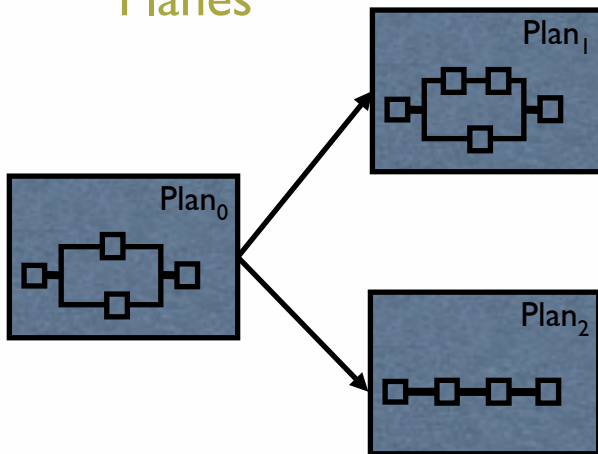
- La profundidad del nodo en el espacio de búsqueda.

Plan almacenado en:

- Series de transiciones de estados

Clasificación

Planes



Movimientos en el espacio:

- Añadir operadores
- Ordenar operadores
- Ligar variables
- O, en caso contrario, restringir el plan.

Modelo del tiempo:

- Conjunto parcialmente ordenado de operadores.

Plan almacenado en:

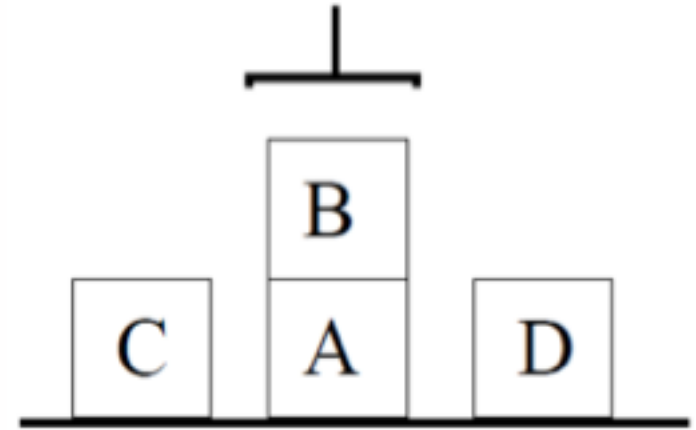
- Un único nodo

Elementos

- **Estados** adecuadamente representados.
- **Acciones** que generan descripciones de estados nuevos.
- **Metas** son los objetivos a lograr.
- **Plan** o solución como secuencia de acciones.

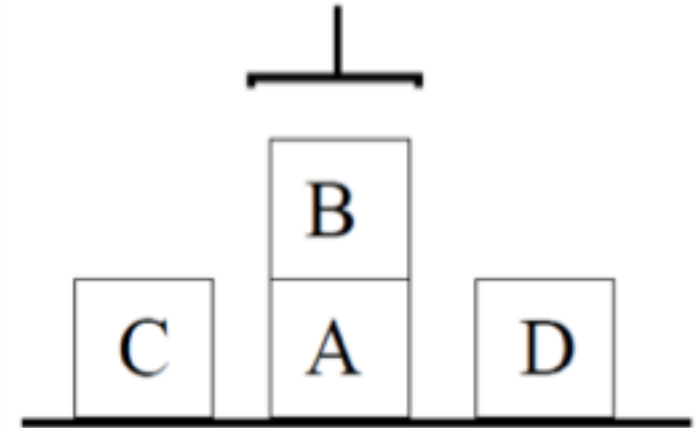
Ejemplo: El mundo de Bloques

- Elementos que intervienen:
 - Una superficie plana.
 - Una serie de bloques cúbicos.
 - Un brazo robotizado, que puede coger un bloque cada vez.
 - Un bloque puede estar sobre la mesa o apilado sobre otro bloque.



Ejemplo: El mundo de Bloques

- Elementos que intervienen:
 - Una superficie plana.
 - Una serie de bloques cúbicos.
 - Un brazo robotizado, que puede coger un bloque cada vez.
 - Un bloque puede estar sobre la mesa o apilado sobre otro bloque.



Ejemplo clásico en planificación.

Metodología

- Necesitamos 2 elementos para la resolución
 - **Lenguaje** de representación
 - Lógica (de 1er orden)
 - ...
 - **Algoritmo**: búsqueda de plan
 - Hay muchos...

Lenguaje

Algunas anotaciones:

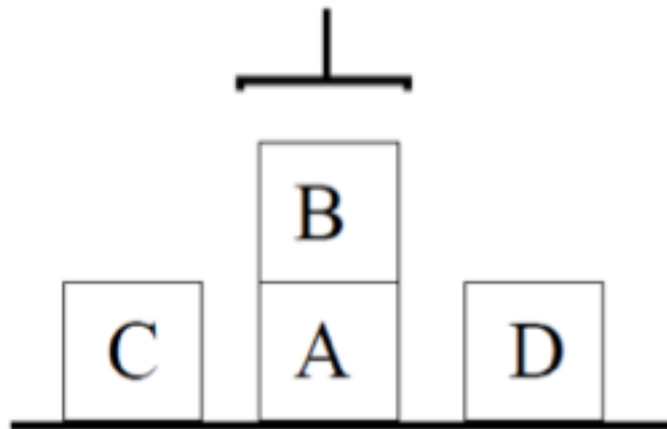
- **Constantes:** objetos del mundo (en mayúsculas)
- **Variables** para representar cualquier objeto (en minúsculas)
- Símbolos de **predicados** (para expresar propiedades de los objetos)
- Símbolos de **acciones** (para representar operadores)

Lenguaje

- Estados: conjunción de “literales cerrados”
- Acciones: fórmulas con literales (cerrados o no).
- Importante:
 - Hipótesis del mundo cerrado: las condiciones que no se mencionan se suponen falsas

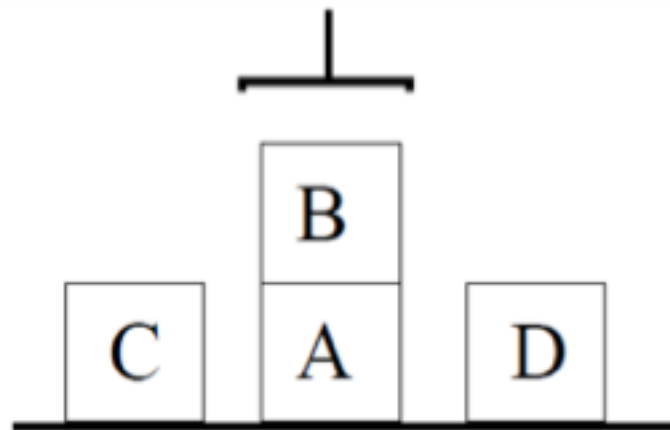
Ejemplo: Representación

- Descriptores de estados:
 - **DESPEJADO**(x), el bloque x está despejado.
 - **BRAZOLIBRE**, el brazo no agarra ningún bloque.
 - **SOBRELAMESA**(x), el bloque x está sobre la mesa.
 - **SOBRE**(x,y), el bloque “x” está sobre el “y”.
 - **AGARRADO**(x), el bloque “x” está sujeto por el brazo.



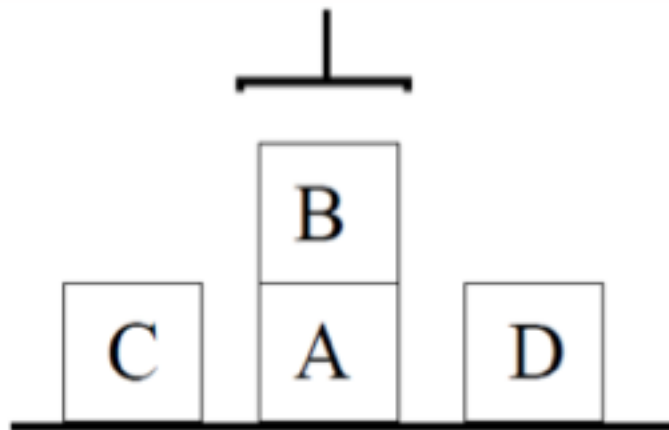
Ejemplo: Representación

- Descriptores de acciones
 - Colocar un bloque sobre otro: **APILAR**(x,y)
 - Quitar un bloque que estaba sobre otro: **DESAPILAR**(x,y)
 - Agarrar un bloque con el robot: **AGARRAR**(x)
 - Bajar un bloque hasta la superficie: **BAJAR**(x)



Ejemplo: Representación

- Estado Inicial
 - DESPEJADO(B), DESPEJADO(C), DESPEJADO(D), BRAZOLIBRE, SOBRE(B,A), SOBRELAMESA(C), SOBRELAMESA(D), SOBRELAMESA(A)



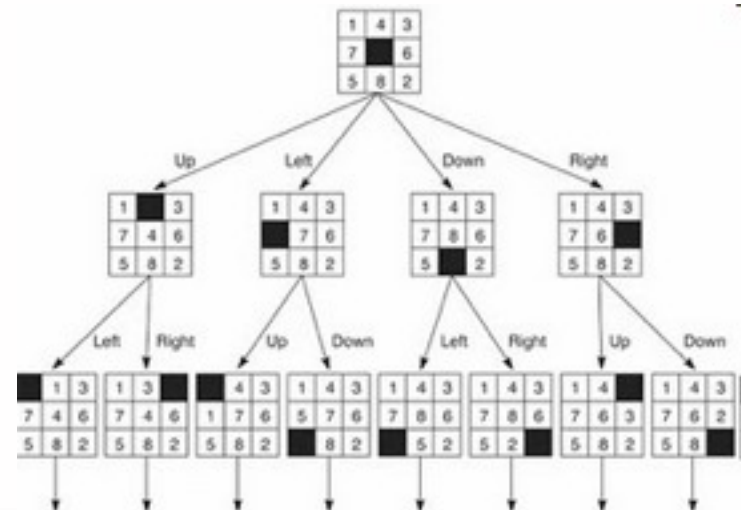
Búsqueda de la solución

Primera aproximación

- Hay muchos algoritmos o formas de abordar este problema.
- En primer lugar lo consideraremos una búsqueda en el espacio de estados
- Cada nodo, un estado

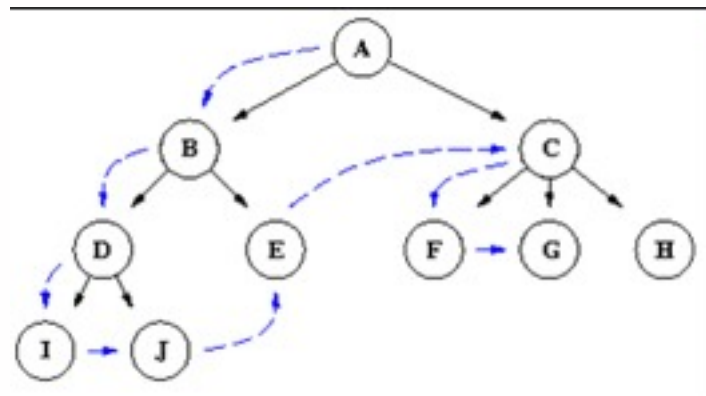
Primera aproximación

- Hay muchos algoritmos o formas de abordar este problema.
- En primer lugar lo consideraremos una búsqueda en el espacio de estados
- Cada nodo, un estado



Primera aproximación

- Podemos generar todos los estados posibles a partir del inicial
- Búsqueda en profundidad o en anchura (clásicas)



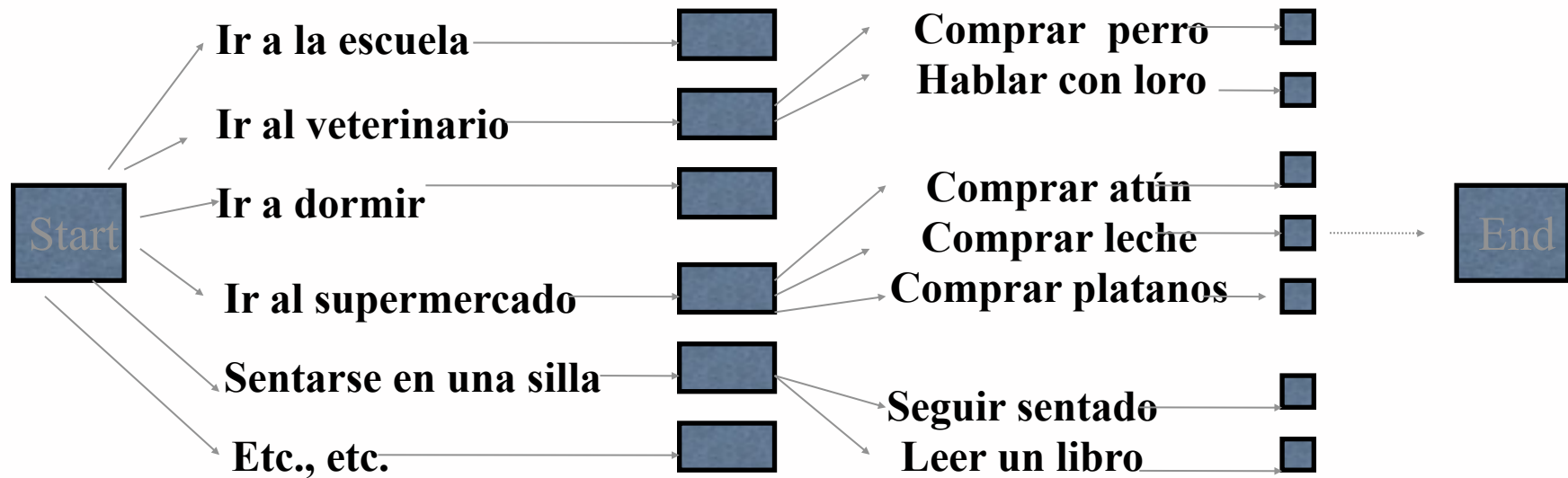
Ejemplo

“Conseguir un litro de leche, plátanos y un taladro de velocidades ajustables”

- Estado inicial: agente en casa, sin leche, sin plátanos y sin taladro
- Conjunto de operaciones: todo lo que pueda hacer
- Heurística: número de cosas sin adquirir

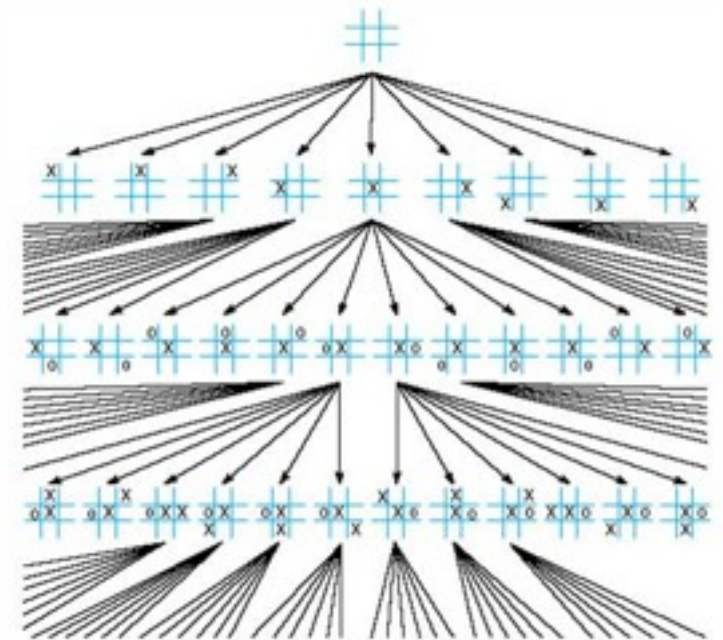
Ejemplo

Ejemplo



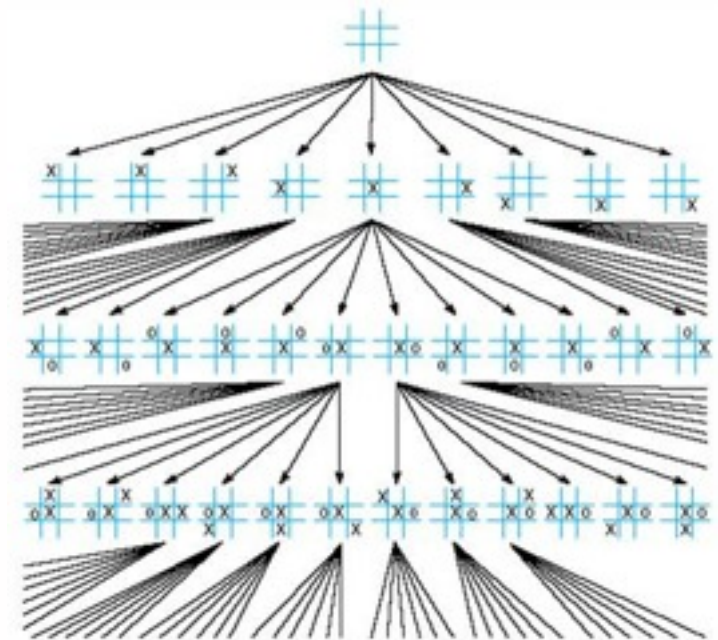
Primera aproximación

- La explosión de estados es ENORME!
- Se pueden usar heurísticas(difícil) e implementar A^*
- Poco resultado



Primera aproximación

- La explosión de estados es ENORME!
- Se pueden usar heurísticas(difícil) e implementar A^*
- Poco resultado



Necesitamos técnicas
diferentes

METODOLOGÍA STRIPS

STRIPS

- STRIPS = **ST**anford **R**esearch **I**nstitute **P**roblem **S**olver
- Utiliza gran parte de los elementos del sistema **GPS**(*)
- Es un **lenguaje** y un **algoritmo**
- Realiza una exploración en profundidad (hacia adelante y hacia atrás, simultáneamente) sobre el espacio de estados del problema

Características STRIPS

- Los objetivos son conjunciones:
 - Pobre \wedge desconocido
- Los efectos son conjunciones
- No tiene infraestructura para soportar igualdades
- No tiene infraestructura para soportar tipos

Características STRIPS

- Sólo literales **positivos** en estados: pobre, desconocido
- Hipótesis del **mundo cerrado**: los literales no mencionados son falsos
- El efecto $P \wedge \neg Q$, significa insertar P y eliminar Q
- Sólo **literales simples** en objetivos

Representación de los estados

- Los estados se expresan definiendo un conjunto de predicados:
 - `tiene_dinero(x, y)`, `vuelo_reservado(x, vuelo, dia)`,
 - `Vuelo(vuelo, origen, destino, hora_salida, hora_llegada)`, ...
- Cada estado se representa por un conjunto de predicados instanciados.
 - `tiene_dinero(Pepe, 1.000)`, `vuelo_reservado(Pepe, IB304, 3Abr06)`, `vuelo(IB304, Madrid, New_York, 12:05, 12:30)`,
 - `vuelo(TWA2001, New_York, San_Francisco, 13:55, 17:20)`, ...

Operadores STRIPS

Operadores STRIPS

Consta de los siguientes elementos:

Operadores STRIPS

Consta de los siguientes elementos:

- *Nombre* del operador

Operadores STRIPS

Consta de los siguientes elementos:

- *Nombre* del operador
- *Parámetros* del operador:

Operadores STRIPS

Consta de los siguientes elementos:

- *Nombre* del operador
- *Parámetros* del operador:
 - *Precondiciones* (P): lista de selectores que se tienen que cumplir para poder aplicar el operador

Operadores STRIPS

Consta de los siguientes elementos:

- *Nombre* del operador
- *Parámetros* del operador:
 - *Precondiciones* (P): lista de selectores que se tienen que cumplir para poder aplicar el operador
 - *Lista de Adición* (A): lista de selectores que se añaden al estado actual al aplicar el operador

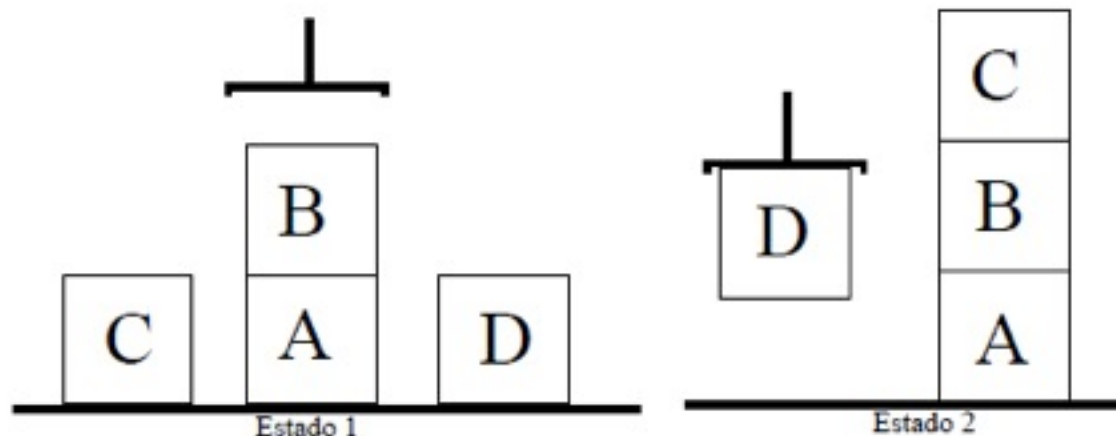
Operadores STRIPS

Consta de los siguientes elementos:

- *Nombre* del operador
- *Parámetros* del operador:
 - *Precondiciones* (P): lista de selectores que se tienen que cumplir para poder aplicar el operador
 - *Lista de Adición* (A): lista de selectores que se añaden al estado actual al aplicar el operador
 - *Lista de Supresión* (S): lista de selectores que se suprimen del estado actual al aplicar el operador

Ejemplo: Mundo de Bloques

- El objetivo $\text{SOBRE}(B,A)$, $\text{SOBRELAMESA}(A)$ es satisfecho por el estado 1 y por el estado 2
- El objetivo $\text{DESPEJADO}(B)$, BRAZOLIBRE es satisfecho por el estado 1 pero no por el estado 2
- El objetivo $\text{SOBRE}(C,B)$, $\text{SOBRE}(D,C)$ no es satisfecho por el estado 1 pero sí por el estado 2



Ejemplo: Operadores

Ejemplo: Operadores

- Colocar un bloque sobre otro:

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)
 - B: DESPEJADO(y), AGARRADO(x)

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)
 - B: DESPEJADO(y), AGARRADO(x)
 - A: BRAZOLIBRE, SOBRE(x,y), DESPEJADO(x)

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)
 - B: DESPEJADO(y), AGARRADO(x)
 - A: BRAZOLIBRE, SOBRE(x,y), DESPEJADO(x)

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)
 - B: DESPEJADO(y), AGARRADO(x)
 - A: BRAZOLIBRE, SOBRE(x,y), DESPEJADO(x)
- Quitar un bloque que estaba sobre otro:

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)
 - B: DESPEJADO(y), AGARRADO(x)
 - A: BRAZOLIBRE, SOBRE(x,y), DESPEJADO(x)
- Quitar un bloque que estaba sobre otro:
 - DESAPILAR(x,y)

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)
 - B: DESPEJADO(y), AGARRADO(x)
 - A: BRAZOLIBRE, SOBRE(x,y), DESPEJADO(x)

- Quitar un bloque que estaba sobre otro:
 - DESAPILAR(x,y)
 - P: SOBRE(x,y), DESPEJADO(x), BRAZOLIBRE

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)
 - B: DESPEJADO(y), AGARRADO(x)
 - A: BRAZOLIBRE, SOBRE(x,y), DESPEJADO(x)

- Quitar un bloque que estaba sobre otro:
 - DESAPILAR(x,y)
 - P: SOBRE(x,y), DESPEJADO(x), BRAZOLIBRE
 - B: SOBRE(x,y), DESPEJADO(x), BRAZOLIBRE

Ejemplo: Operadores

- Colocar un bloque sobre otro:
 - APILAR(x,y)
 - P: DESPEJADO(y), AGARRADO(x)
 - B: DESPEJADO(y), AGARRADO(x)
 - A: BRAZOLIBRE, SOBRE(x,y), DESPEJADO(x)

- Quitar un bloque que estaba sobre otro:
 - DESAPILAR(x,y)
 - P: SOBRE(x,y), DESPEJADO(x), BRAZOLIBRE
 - B: SOBRE(x,y), DESPEJADO(x), BRAZOLIBRE
 - A: AGARRADO(x), DESPEJADO(y)

Ejemplo: Acciones

Ejemplo: Acciones

- Agarrar un bloque con el robot:
 - AGARRAR(x)
 - P: DESPEJADO(x), SOBRELAMESA(x), BRAZOLIBRE
 - B: DESPEJADO(x), SOBRELAMESA(x), BRAZOLIBRE
 - A: AGARRADO(x)

Ejemplo: Acciones

- Agarrar un bloque con el robot:
 - AGARRAR(x)
 - P: DESPEJADO(x), SOBRELAMESA(x), BRAZOLIBRE
 - B: DESPEJADO(x), SOBRELAMESA(x), BRAZOLIBRE
 - A: AGARRADO(x)

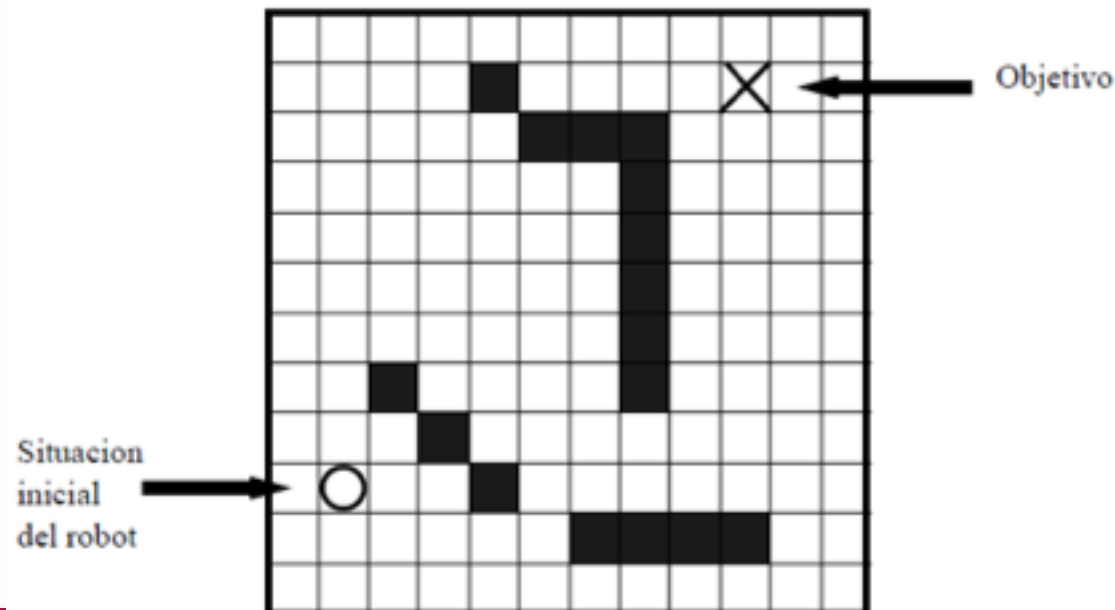
Ejemplo: Acciones

- Agarrar un bloque con el robot:
 - AGARRAR(x)
 - P: DESPEJADO(x), SOBRELAMESA(x), BRAZOLIBRE
 - B: DESPEJADO(x), SOBRELAMESA(x), BRAZOLIBRE
 - A: AGARRADO(x)

- Bajar un bloque hasta la superficie:
 - BAJAR(x)
 - P: AGARRADO(x)
 - B: AGARRADO(x)
 - A: SOBRELAMESA(x), BRAZOLIBRE, DESPEJADO(x)

Movimiento de un robot

- Un robot ha de desplazarse por una rejilla, desde una posición inicial a una final
 - 8 movimientos posibles: N, S, E, O, NO, NE, SO, SE
 - En algunas de las rejillas existen obstáculos no franqueables



Movimiento de un robot

Movimiento de un robot

- Lenguaje:
 - Constantes: números que indican coordenadas horizontales y verticales
 - Predicados: ROBOT-EN(-,-) y LIBRE(-,-)

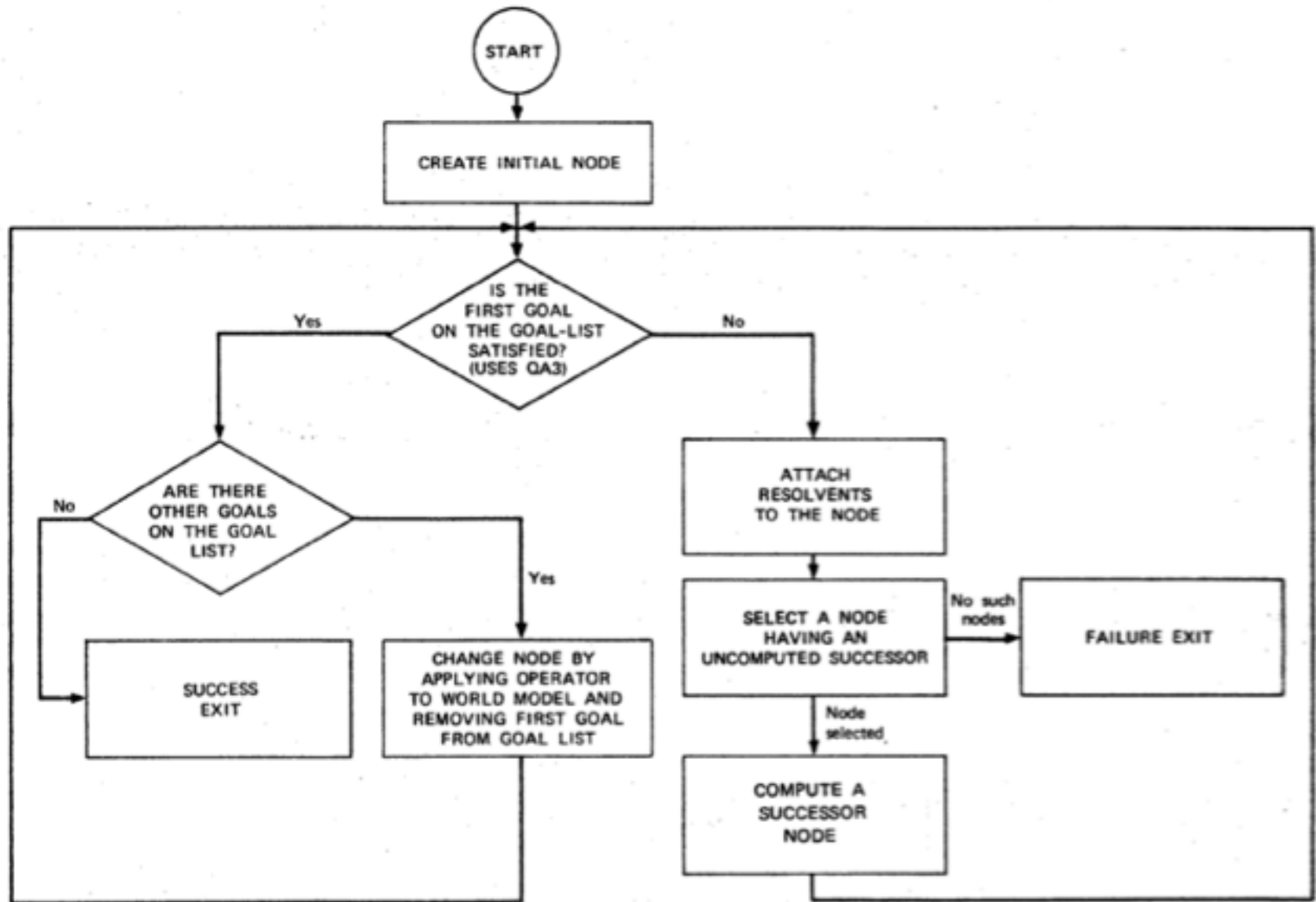
Movimiento de un robot

- Lenguaje:
 - Constantes: números que indican coordenadas horizontales y verticales
 - Predicados: ROBOT-EN(-,-) y LIBRE(-,-)
- Estado inicial (casillas sin obstáculos y posición del robot):
 - LIBRE(1,1),...,LIBRE(6,2),LIBRE(11,2),...,LIBRE(12,12),ROBOT-EN(2,3).
- Objetivo: ROBOT-EN(10,11)

Movimiento de un robot

- Lenguaje:
 - Constantes: números que indican coordenadas horizontales y verticales
 - Predicados: ROBOT-EN(-,-) y LIBRE(-,-)
- Estado inicial (casillas sin obstáculos y posición del robot):
 - LIBRE(1,1),...,LIBRE(6,2),LIBRE(11,2),...,LIBRE(12,12),ROBOT-EN(2,3).
- Objetivo: ROBOT-EN(10,11)
- Acciones (sólo una, las siete restantes son análogas):
 - MOVER-SE(x,y)
 - P: ROBOT-EN(x,y),LIBRE(x+1,y-1)
 - B: ROBOT-EN(x,y),LIBRE(x+1,y-1)
 - A: ROBOT-EN(x+1,y-1),LIBRE(x,y)

Algoritmo Original STRIPS



TA-8258-30

Algoritmo STRIPS

- Se necesita una pila
 - Objetivos NO conseguidos
 - Acciones a aplicar
- En cada momento se conoce la descripción del estado actual.
- Un conjunto de acciones con sus listas precondition, añadido y borrado.
- Inicializar la pila con el estado final

Algoritmo de STRIPS

Repetir hasta que pila = \emptyset OR no se puedan expandir más nodos

- Si la cima de la pila del nodo es un operador,

 - Si el operador se puede ejecutar,

 - ejecutar operador

 - quitarlo de la pila

 - añadirlo al plan

 - Si no, se introducen sus precondiciones en la pila

- Si la cima de la pila del nodo es una meta

 - Si la meta es cierta en el estado, se elimina de la pila

 - Si no,

 - Si hay bucle ¿ciclo? de meta, retroceder

 - Si no generar un sucesor por cada instanciación de operador que añade la meta

 - Si hay sucesores, elegir uno

 - Si no retroceder

- Si la cima de la pila del nodo es una conjunción de metas

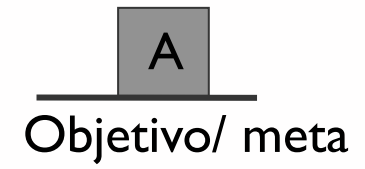
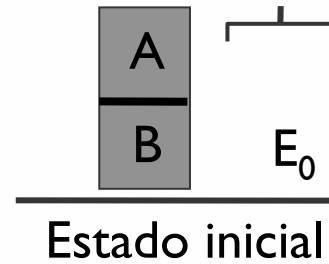
 - Si la conjunción es cierta en el estado, se elimina de la pila

 - Si no generar como sucesores todas las posibles combinaciones de las metas
seleccionar una de ellas

Algoritmo STRIPS

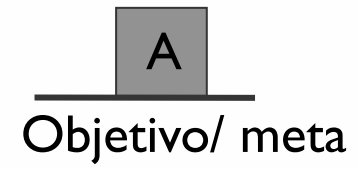
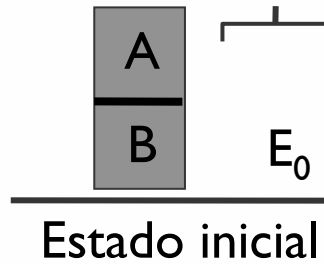
- STRIPS consiste en seleccionar un operador que permita alcanzar el *objetivo* al aplicarlo. Las precondiciones pasan a ser subobjetivos que deben alcanzarse
- Se seguirá descomponiendo hasta que todas las precondiciones se cumplan y se pueda aplicar el operador
- Un *nodo de exploración* contiene dos variables:
 - ESTADO, que almacena el estado en ese nodo
 - PILA, que almacena selectores, operadores y conjunciones de selectores
- Un *nodo de FALLO* es aquél en el cual, para alcanzar un determinado objetivo A, es necesario que A se cumpla

Ejemplo STRIPS



Ejemplo STRIPS

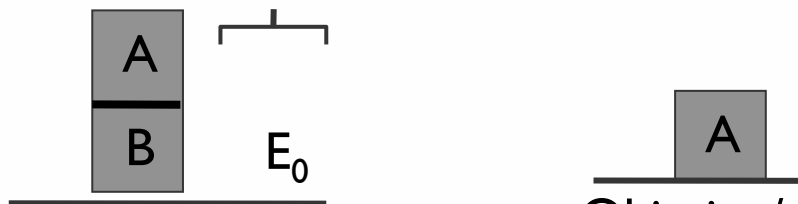
en_mesa(A) E_0



E0

Ejemplo STRIPS

en_mesa(A) E_0



- **DESAPILAR(x, y)**

precondición: sobre(x,y), libre(x), mano_libre

añadido: en_mano(x), libre(y)

borrado: sobre(x, y), mano_libre, libre(x)

- **COGER(x)**

precondición: en_mesa(x), libre(x), mano_libre

añadido: en_mano(x)

borrado: en_mesa(x), mano_libre, libre(x)

- **APILAR(x, y)**

precondición: en_mano(x), libre(y)

añadido: sobre(x, y), libre(x), mano_libre

borrado: en_mano(x), libre(y)

- **DEJAR(x)**

precondición: en_mano(x)

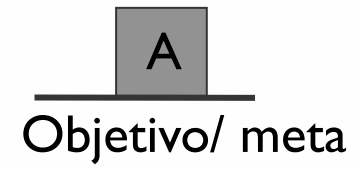
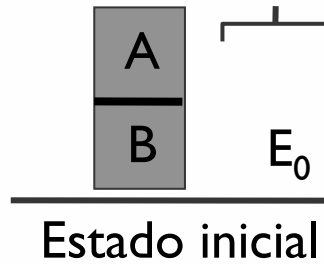
añadido: en_mesa(x), libre(x), mano_libre

borrado: en_mano(x)

E0

Ejemplo STRIPS

en_mesa(A) E_0

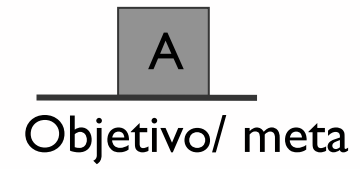
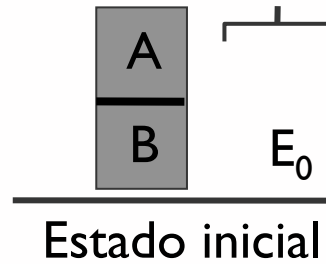


E_0

Ejemplo STRIPS

en_mesa(A) E_0

DEJAR(A)
en_mesa(A) E_0

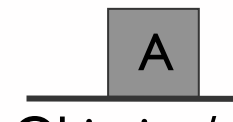
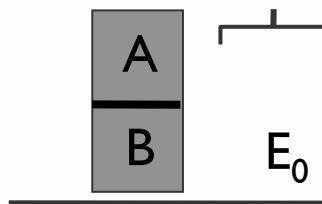


E0

Ejemplo STRIPS

en_mesa(A) E_0

DEJAR(A)
en_mesa(A) E_0



Objetivo / meta

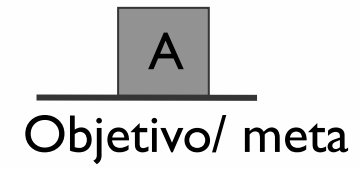
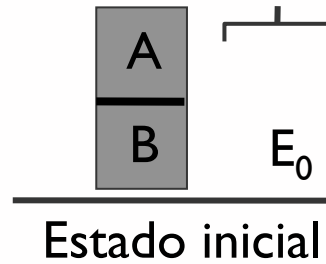
- **DESAPILAR(x, y)**
precondición: sobre(x,y), libre(x), mano_libre
añadido: en_mano(x), libre(y)
borrado: sobre(x, y), mano_libre, libre(x)
- **COGER(x)**
precondición: en_mesa(x), libre(x), mano_libre
añadido: en_mano(x)
borrado: en_mesa(x), mano_libre, libre(x)
- **APILAR(x, y)**
precondición: en_mano(x), libre(y)
añadido: sobre(x, y), libre(x), mano_libre
borrado: en_mano(x), libre(y)
- **DEJAR(x)**
precondición: en_mano(x)
añadido: en_mesa(x), libre(x), mano_libre
borrado: en_mano(x)

E0

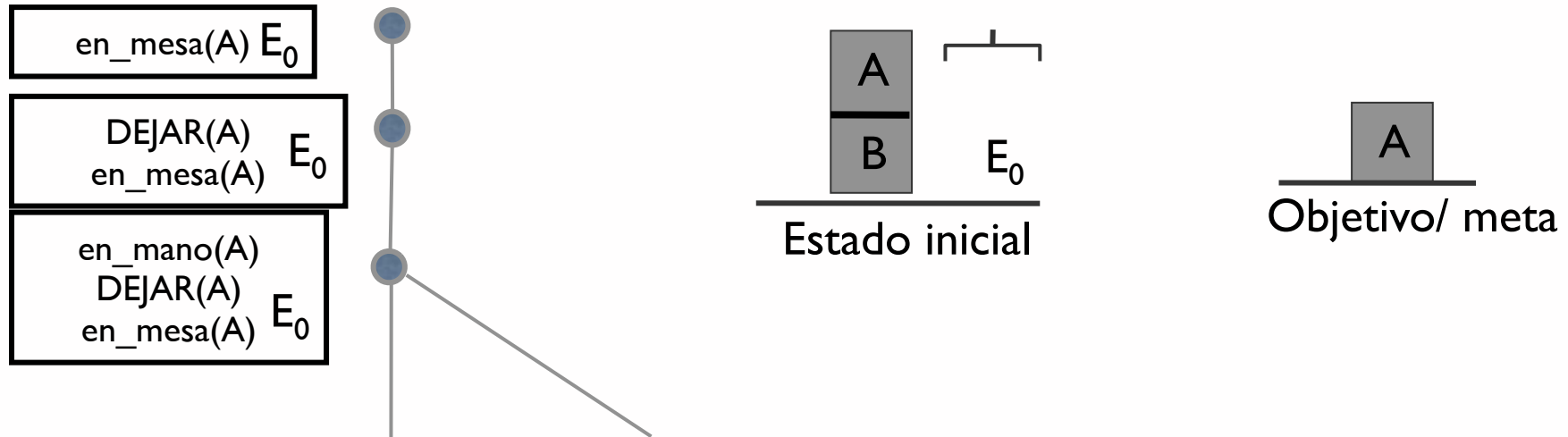
Ejemplo STRIPS

en_mesa(A) E_0

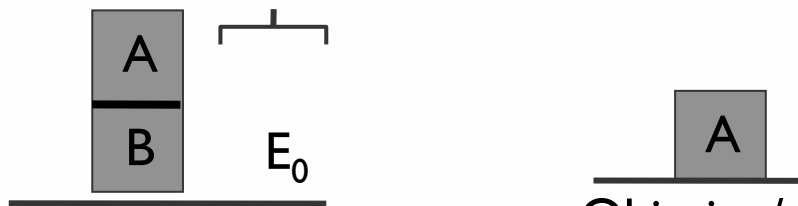
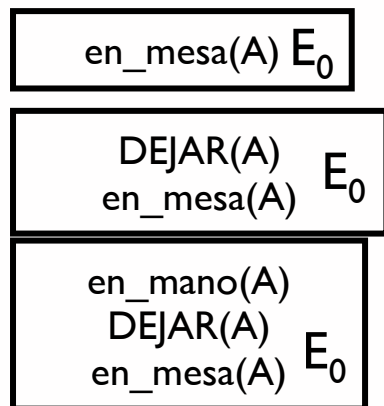
DEJAR(A)
en_mesa(A) E_0



Ejemplo STRIPS



Ejemplo STRIPS



DESAPILAR(x, y)

precondición: sobre(x,y), libre(x), mano_libre

añadido: en_mano(x), libre(y)

borrado: sobre(x, y), mano_libre, libre(x)

COGER(x)

precondición: en_mesa(x), libre(x), mano_libre

añadido: en_mano(x)

borrado: en_mesa(x), mano_libre, libre(x)

APILAR(x, y)

precondición: en_mano(x), libre(y)

añadido: sobre(x, y), libre(x), mano_libre

borrado: en_mano(x), libre(y)

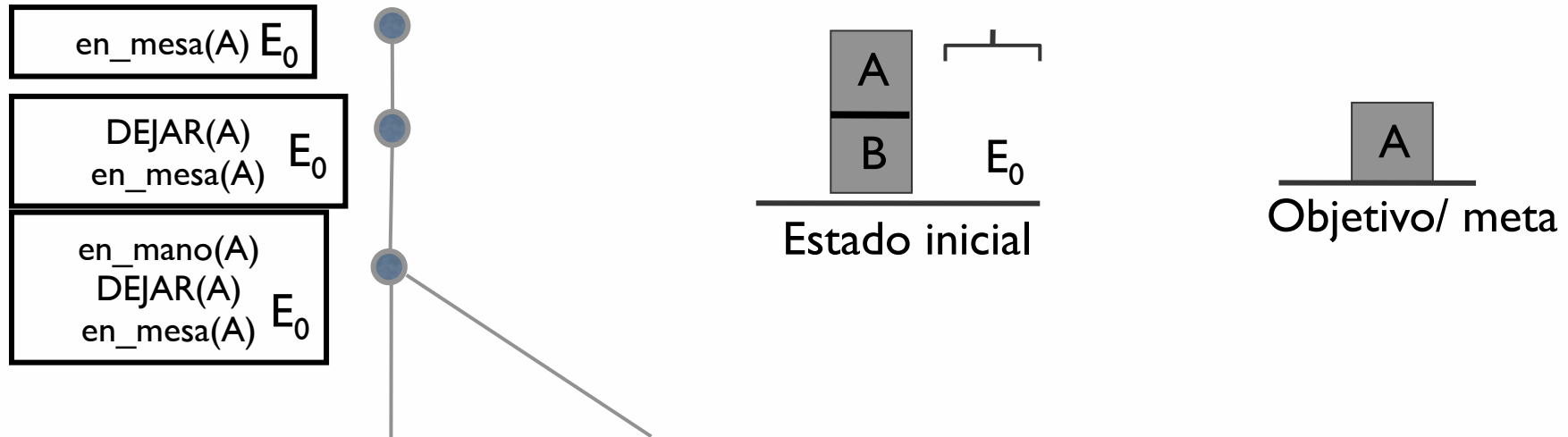
DEJAR(x)

precondición: en_mano(x)

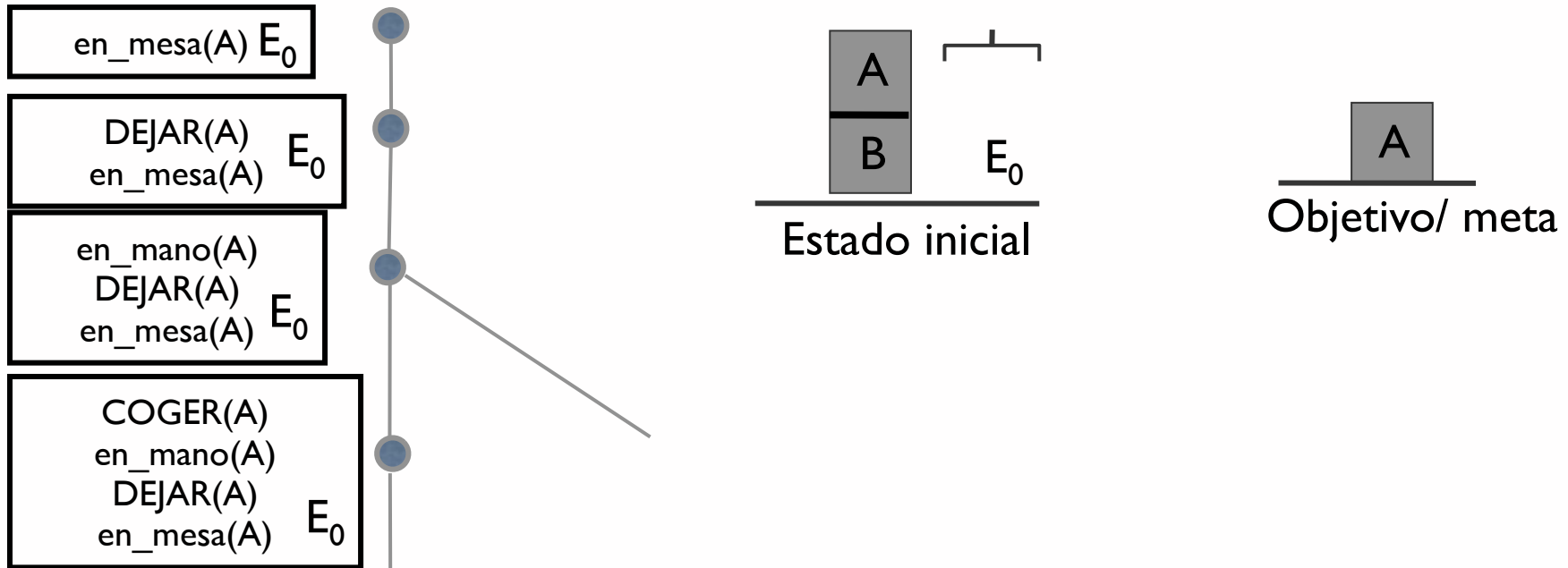
añadido: en_mesa(x), libre(x), mano_libre

borrado: en_mano(x)

Ejemplo STRIPS

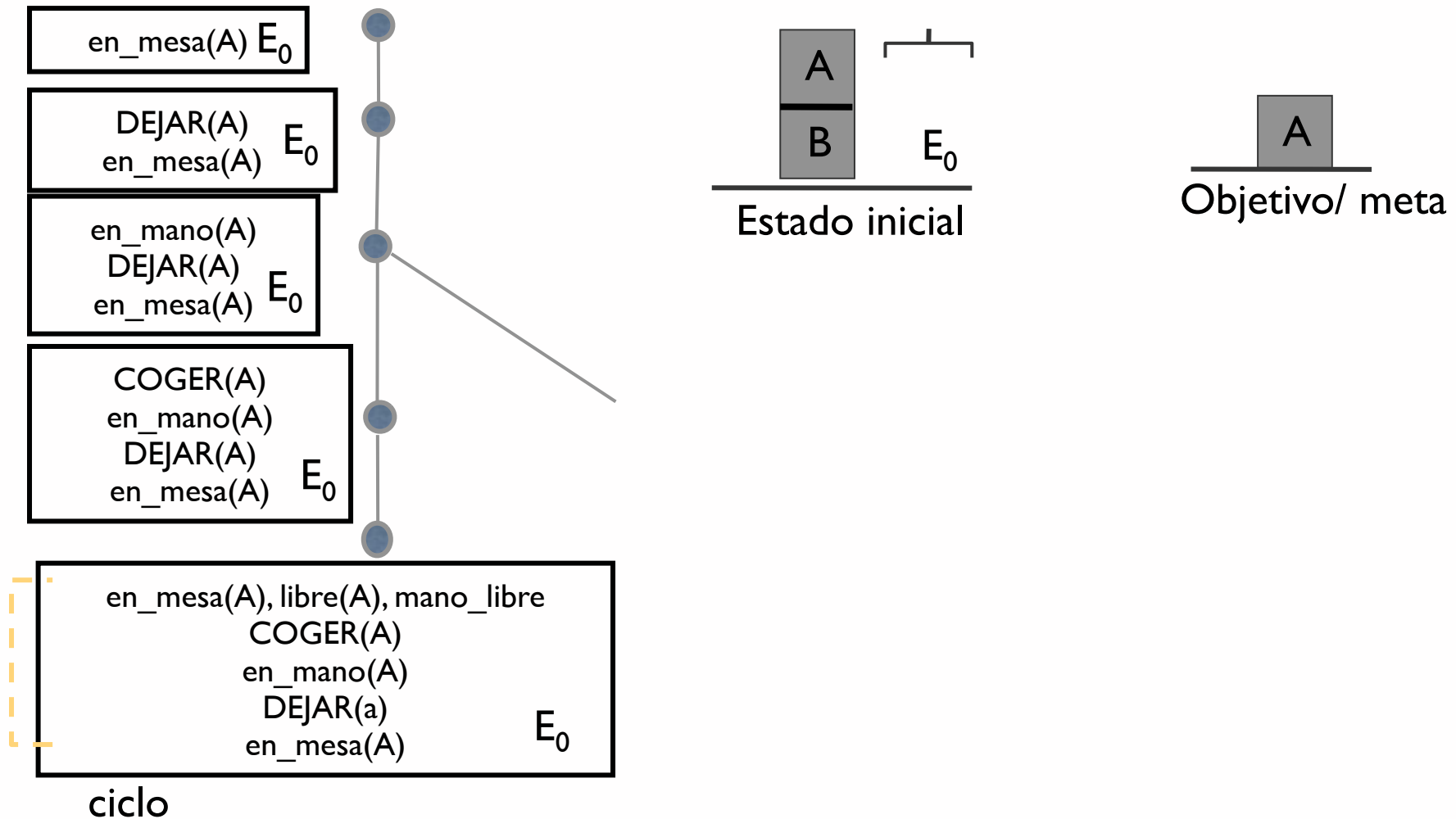


Ejemplo STRIPS

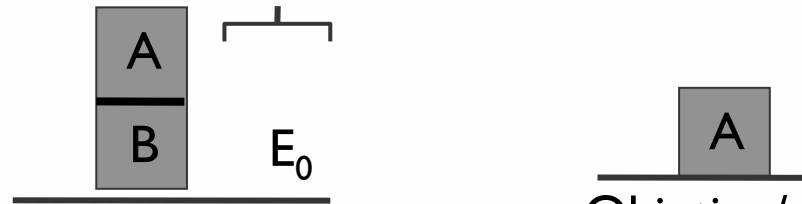
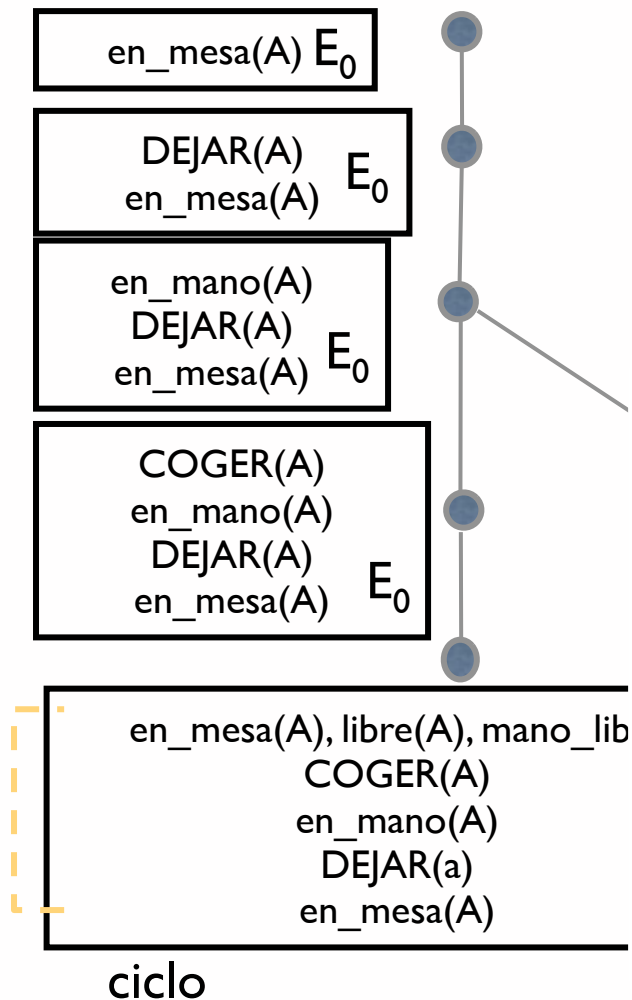


E0

Ejemplo STRIPS

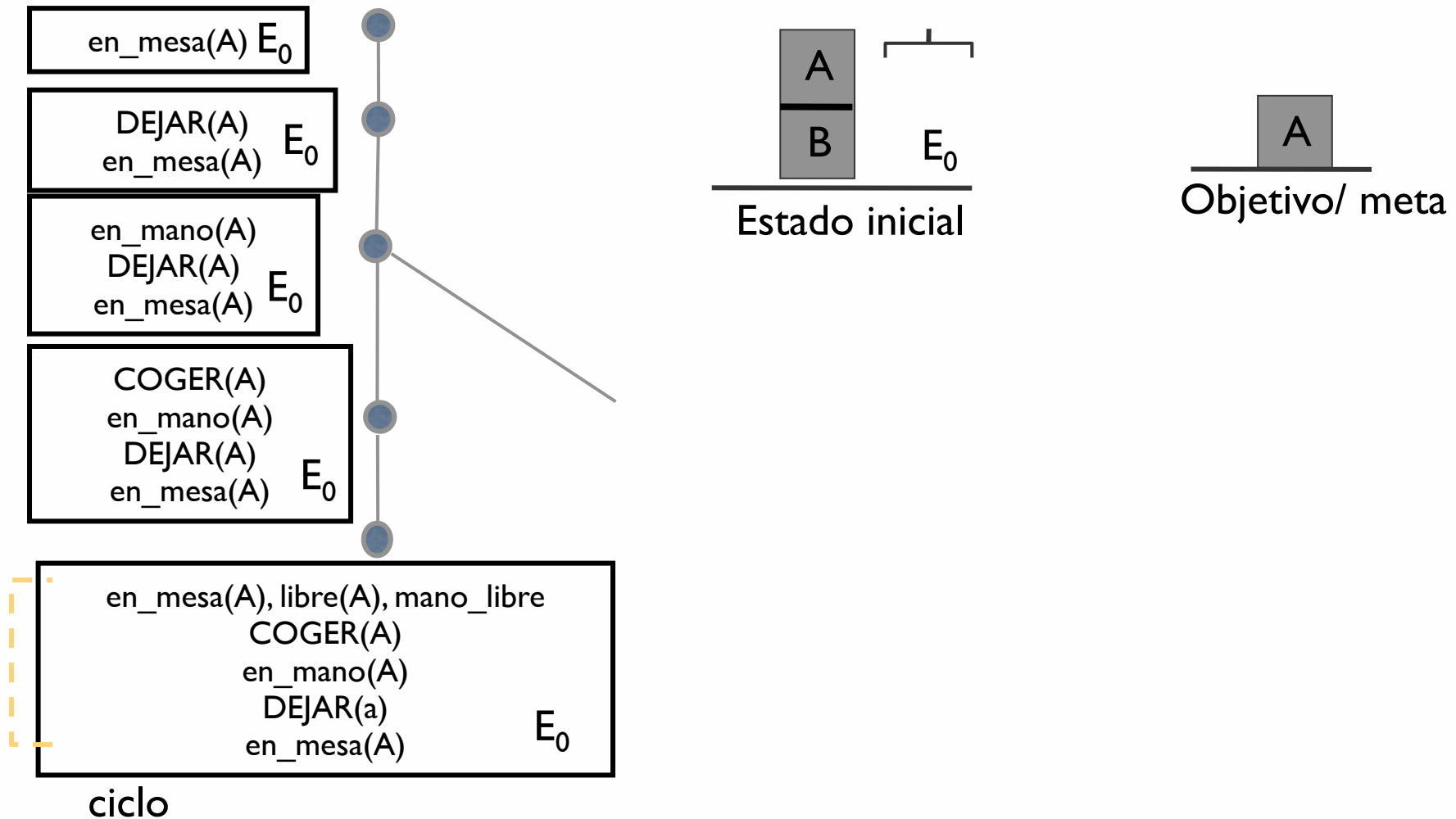


Ejemplo STRIPS

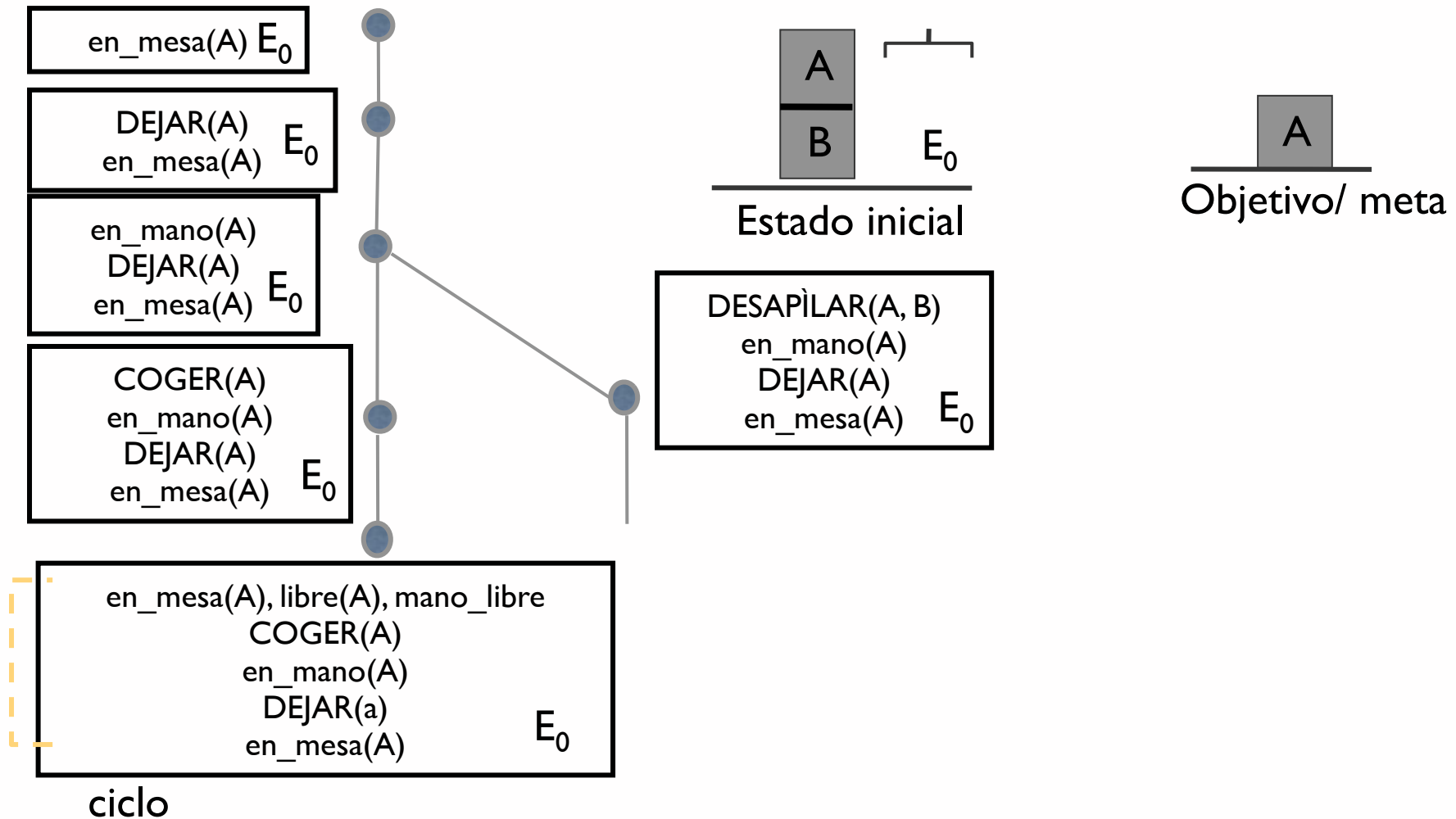


- DESAPILAR(x, y)**
 precondition: $sobre(x,y), libre(x), mano_libre$
 added: $en_mano(x), libre(y)$
 deleted: $sobre(x, y), mano_libre, libre(x)$
- COGER(x)**
 precondition: $en_mesa(x), libre(x), mano_libre$
 added: $en_mano(x)$
 deleted: $en_mesa(x), mano_libre, libre(x)$
- APILAR(x, y)**
 precondition: $en_mano(x), libre(y)$
 added: $sobre(x, y), libre(x), mano_libre$
 deleted: $en_mano(x), libre(y)$
- DEJAR(x)**
 precondition: $en_mano(x)$
 added: $en_mesa(x), libre(x), mano_libre$
 deleted: $en_mano(x)$

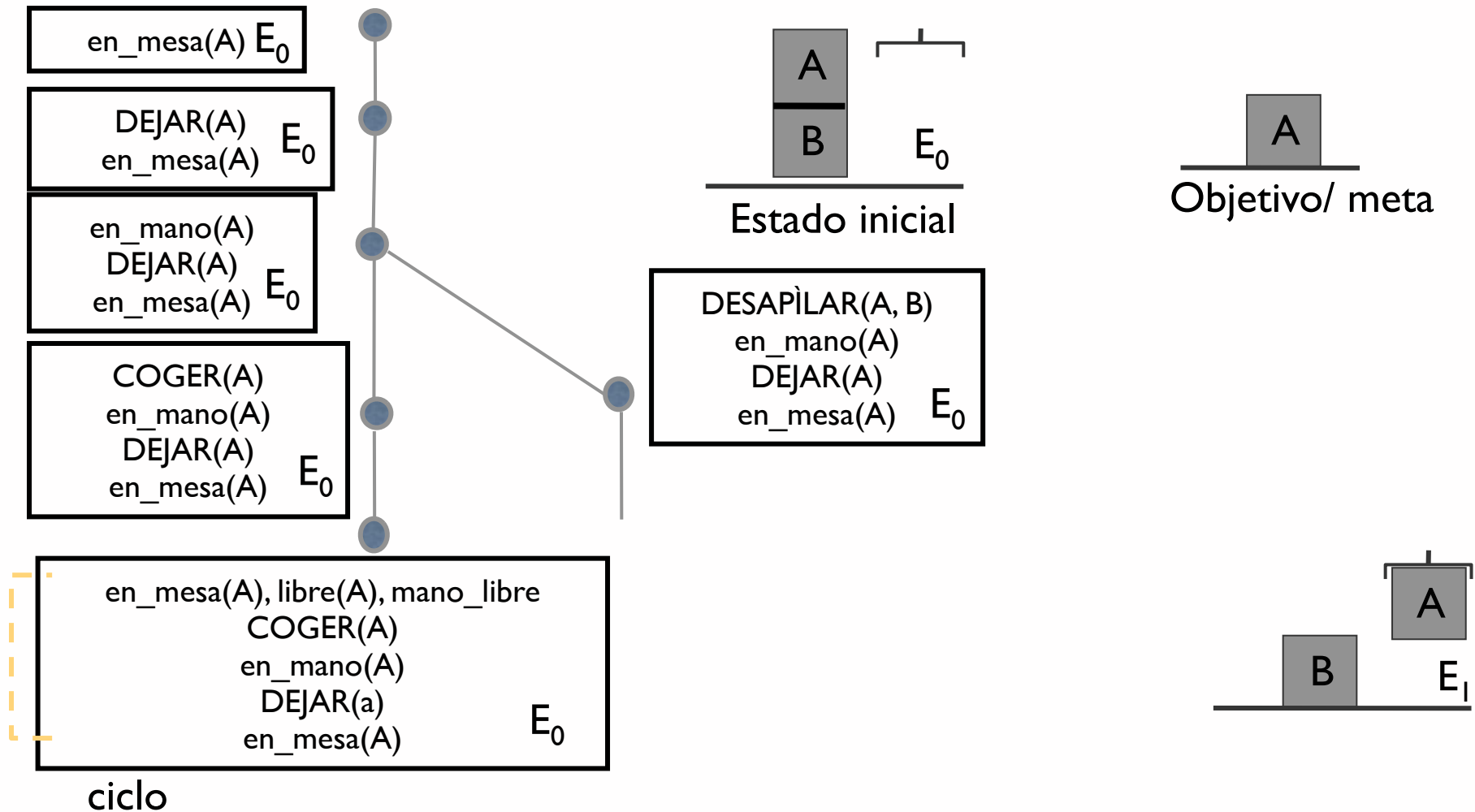
Ejemplo STRIPS



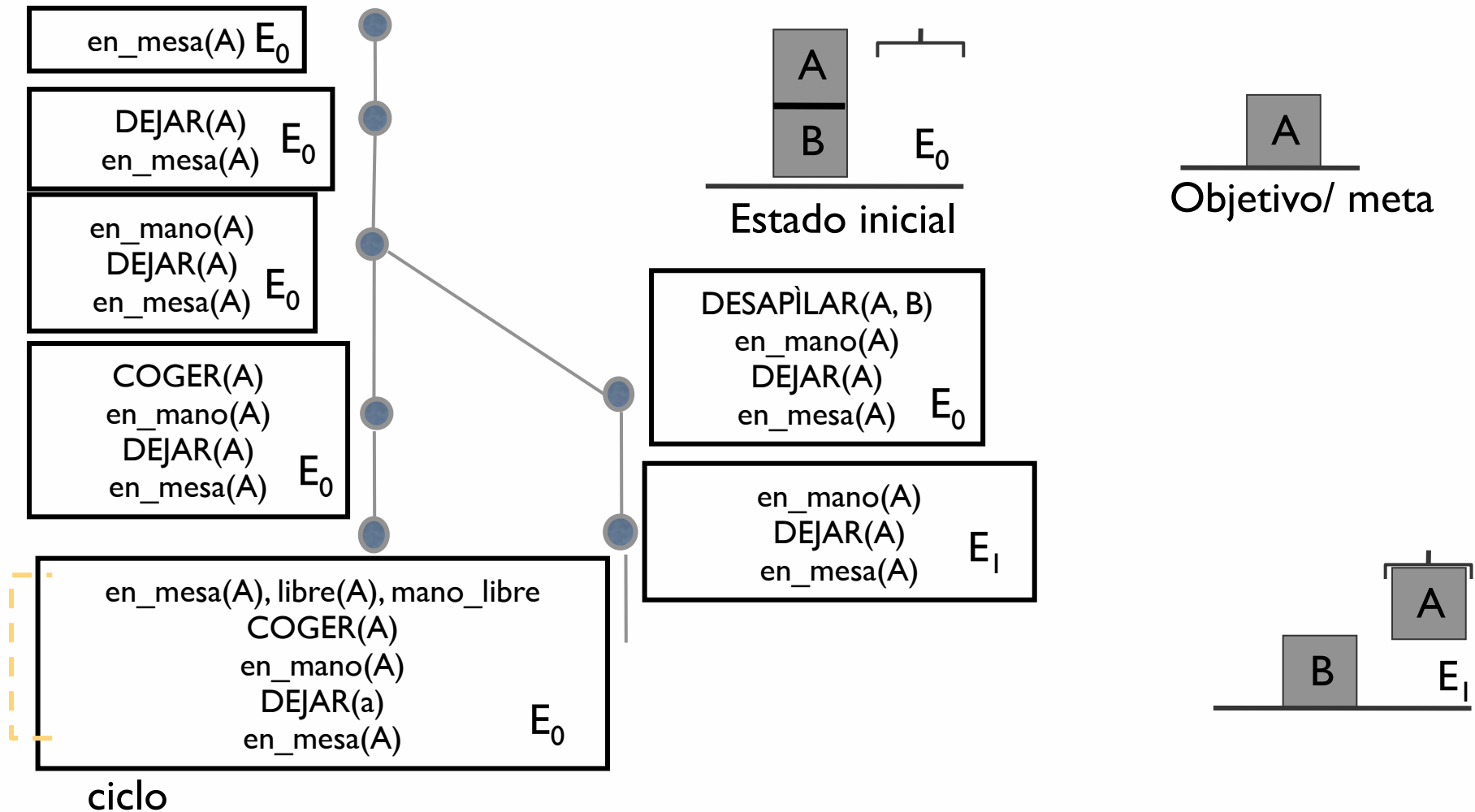
Ejemplo STRIPS



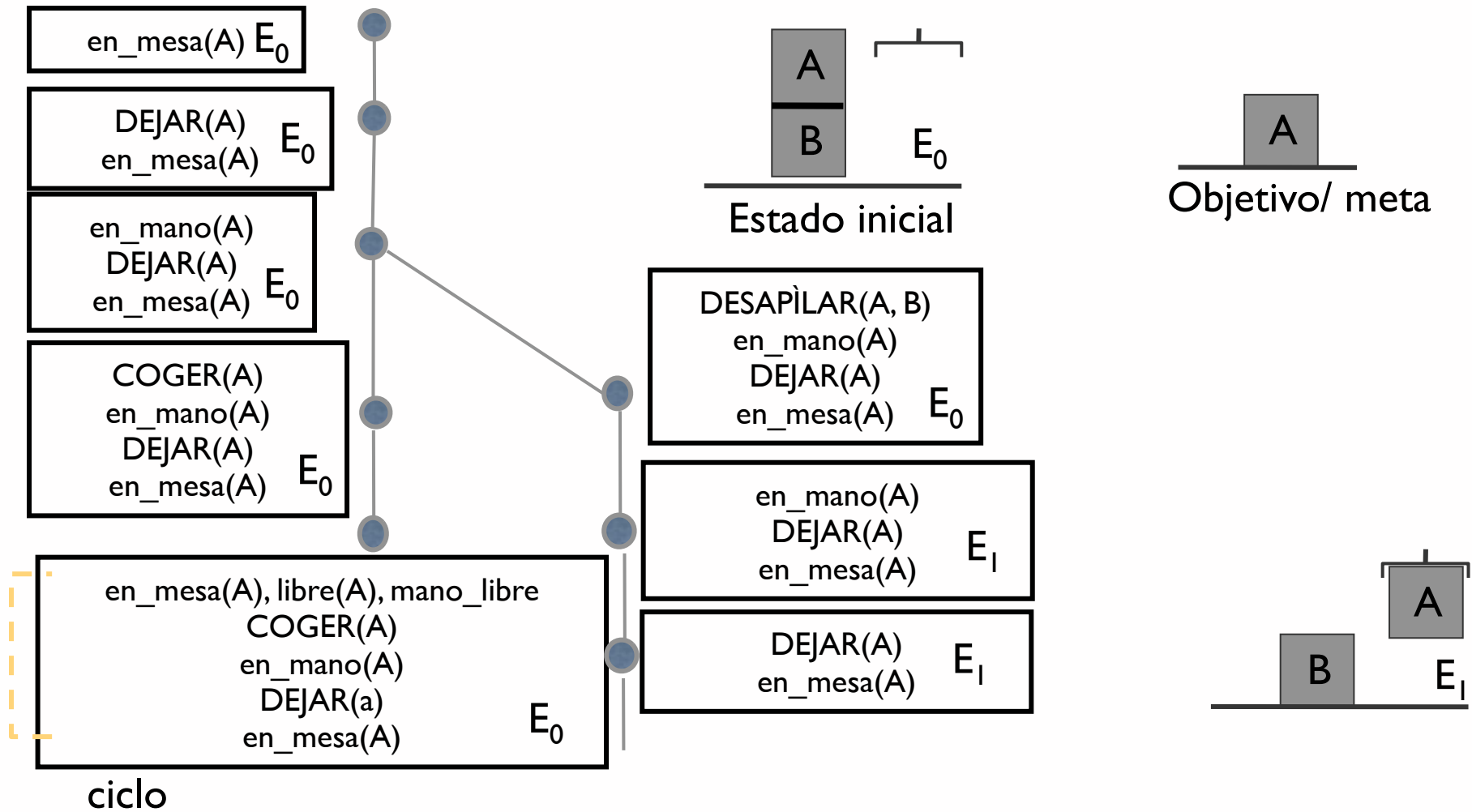
Ejemplo STRIPS



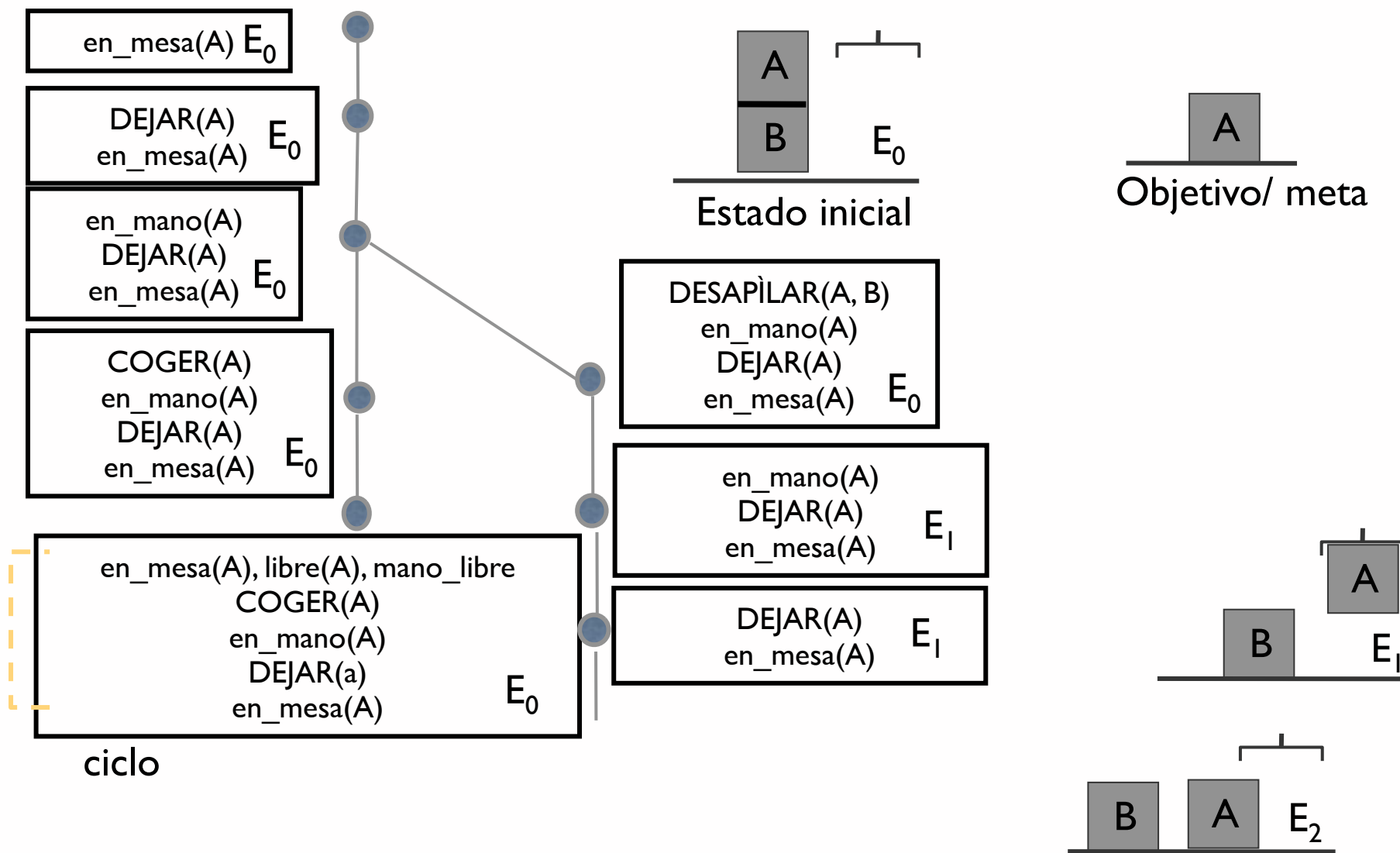
Ejemplo STRIPS



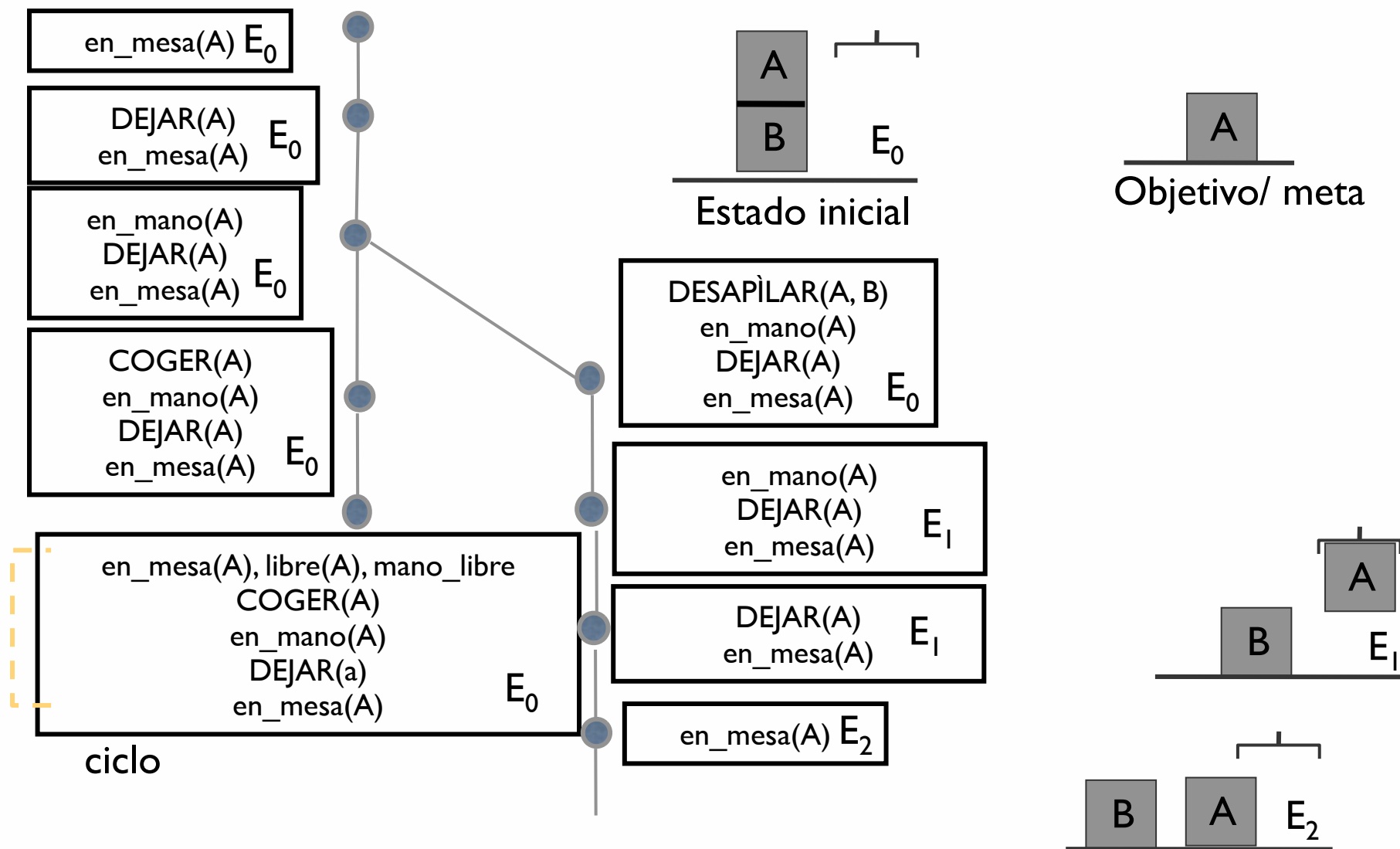
Ejemplo STRIPS



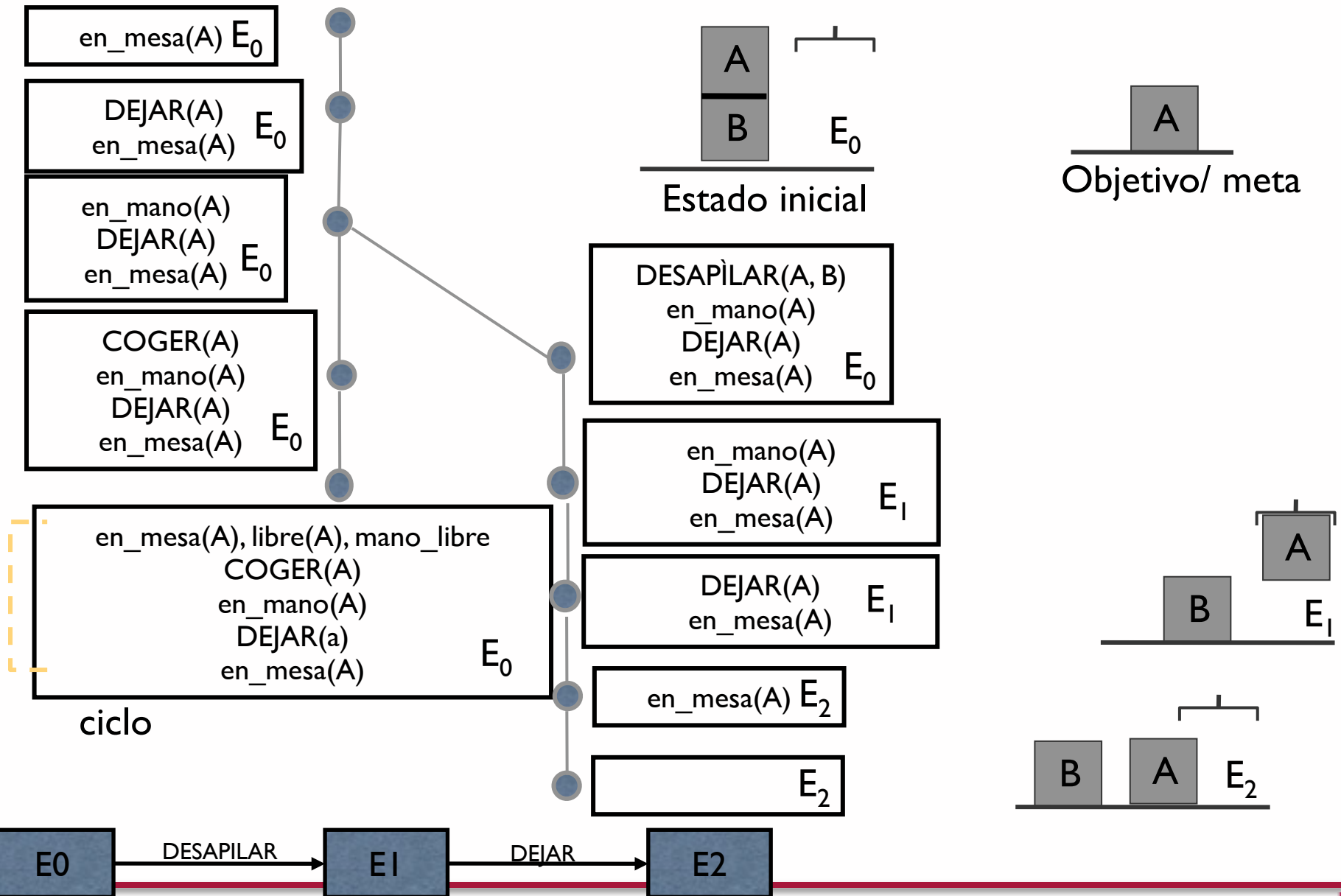
Ejemplo STRIPS



Ejemplo STRIPS



Ejemplo STRIPS

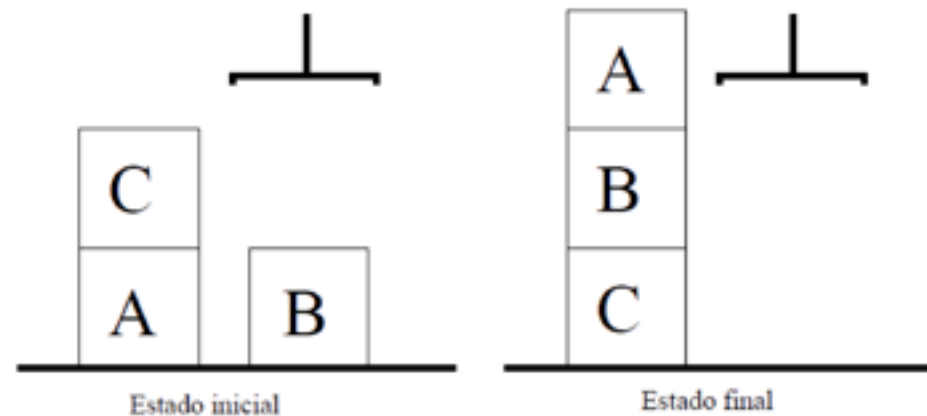


Anomalía de Sussmann

- Es un problema **intrínseco** a la Planificación Lineal.
- Está basado en la “**independencia**” de los objetivos

Anomalia de Sussman

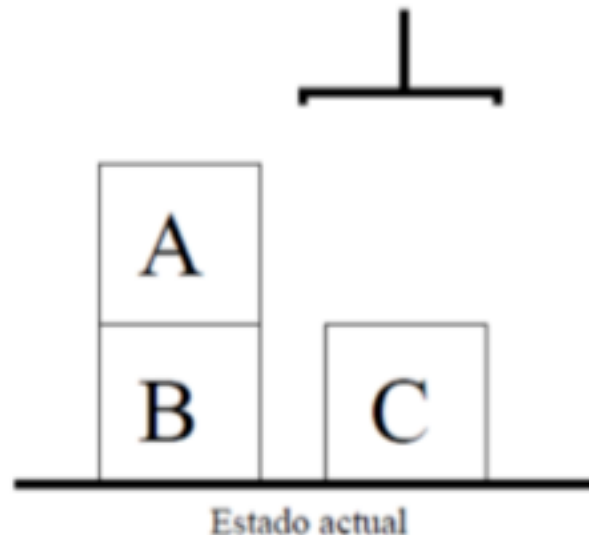
- Carencias de la planificación lineal



- Estados
 - Estado inicial = {SOBRELAMESA(A), SOBRELAMESA(B), DESPEJADO(B), DESPEJADO(C), BRAZOLIBRE, SOBRE(C,A)}
 - Objetivo = { SOBRE(A,B),SOBRE(B,C) }

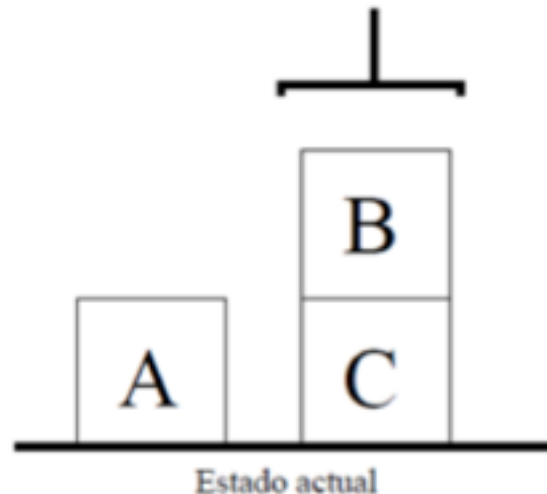
Anomalia de Sussman

- Un plan para satisfacer **SOBRE(A,B)**
- Estados intermedio



Anomalía de Sussman

- Un plan para satisfacer: SOBRE(B,C):
 - DESAPILAR(A,B),BAJAR(A), AGARRAR(B),APILAR(B,C)
- ESTADO INTERMEDIO:



- Problema SOBRE(A,B) ha dejado de ser cierto.

Mejoras

- El algoritmo STRIPS es una búsqueda más eficiente que los tradicionales
- Aunque necesita mejoras
- Aprendizaje de operadores:
 - Tablas triangulares

Alternativas

- Búsqueda de planes no lineales, entrelazando objetivos.

- Ejemplo:

La anomalía de Sussman se puede solucionar:

- 1. Comenzado a satisfacer SOBRE(A,B), en parte sólo, despejando A, (DESAPILAR(C,A), BAJAR(C)).
- 2. Satisfacer SOBRE(B,C), haciendo APILAR(B,C).
- 3. Completar SOBRE(A,B), haciendo APILAR(A,B)

- Principio de *mínimo compromiso*: no determinar completamente el orden entre acciones, a no ser que sea estrictamente necesario