Inteligencia Artificial e Ingeniería del Conocimiento

Elena Verdú Pérez

Planificación Automática

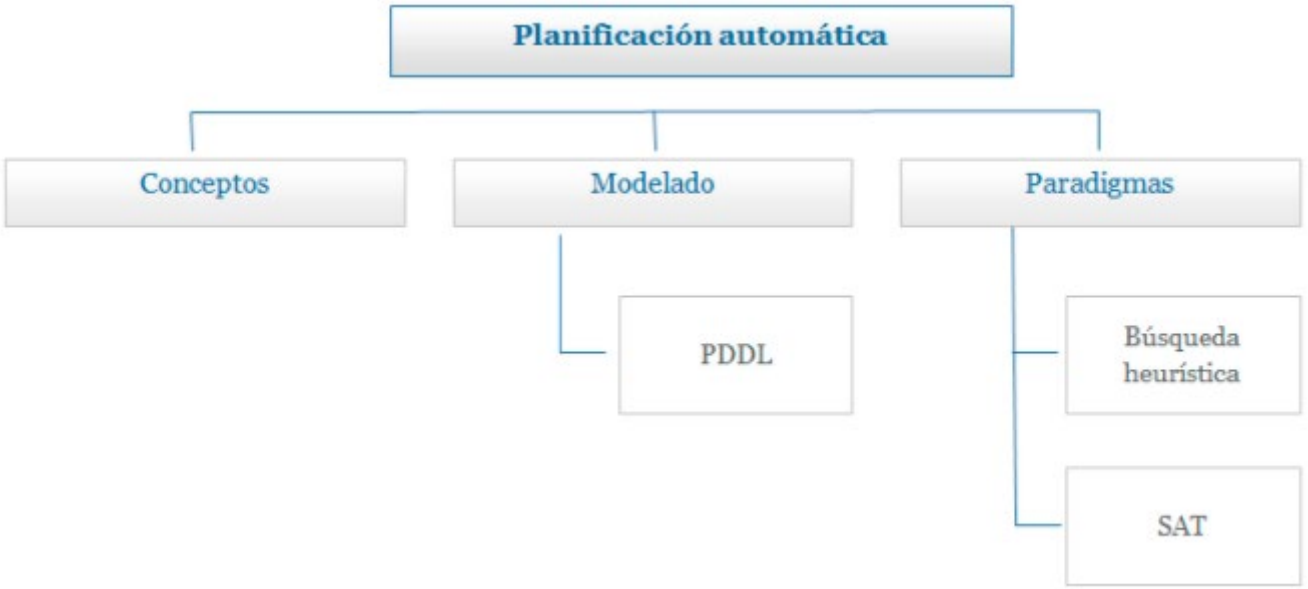
unir LA UNIVERSIDAD EN INTERNET

Universidad Internacional de La Rioja

Objetivos de esta clase

- Describir qué problemas se resuelven mediante planificación automática.
- Conocer cómo modelar un problema de planificación automática.
- Comprender el modelado de un problema mediante el lenguaje de definición de dominios de planificación PDDL.
- Comprender cómo podemos utilizar la búsqueda heurística para resolver un problema de planificación.
- Desarrollar un Grafo de planificación que permite solucionar el problema con SAT o como un problema de búsqueda hacia atrás.

¿Cómo estudiar este tema?



» TEMA 4. PLANIFICACIÓN AUTOMÁTICA [Esquema Tema]

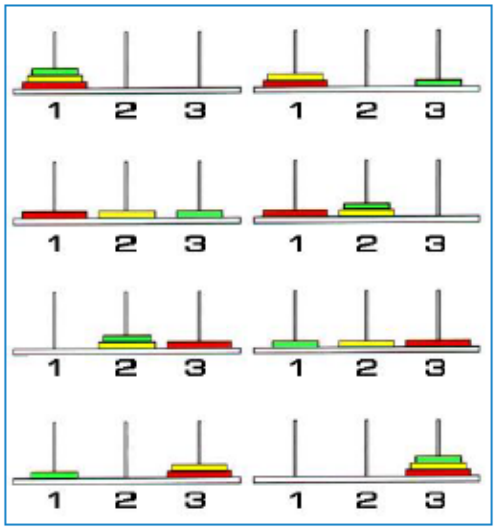
IDEAS CLAVE	LO + RECOMENDADO	+ INFORMACIÓN	TEST
¿Cómo estudiar este tema?	**Lecciones magistrales**	**A fondo**	
Planificación Automática y PDDL	Planificación Automática	Planificación Automática	
Planificación como Búsqueda Heurística	**No dejes de visitar...**	Material de un curso de Automated Planning	
Planificación con SAT	Competición Internacional de Planificación Automática	**Bibliografía**	
		Recursos externos	
		Fast Downward	

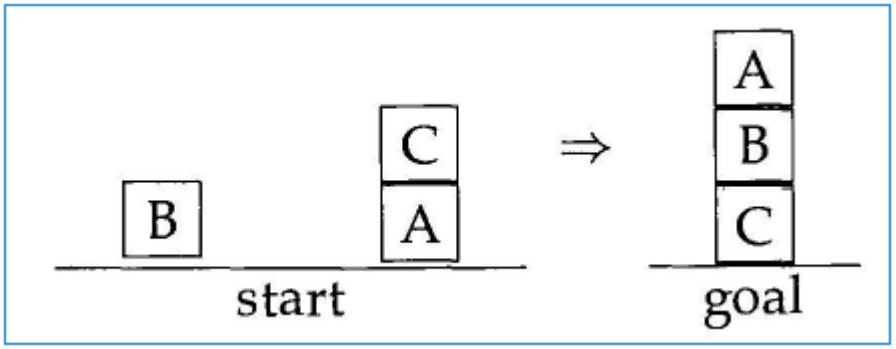
Planificación automática

Área dentro de la IA cuyo objetivo es resolver **problemas cuya solución sea una secuencia de acciones**

Ejemplos de problemas:

- ❑ Red de vehículos para transportar paquetes.
- ❑ Cargar y descargar contenedores en un puerto.
- ❑ Resolver un puzzle.





Planificación automática

La solución es trazar un plan: determinar una serie de acciones que nos lleven de un estado inicial a un estado final

Múltiples características:

- ❑ Determinismo.
- ❑ Conocimiento completo o incompleto.
- ❑ Estados continuos o discretos.
- ❑ Temporalidad.
- ❑ Paralelismo.
- ❑ Función objetivo.
- ❑ Optimalidad.

Planificación automática

PLANIFICACIÓN CLÁSICA

P=<S,A,I,G>

S - proposiciones del problema
A – acciones
I – proposiciones del estado inicial
G – proposiciones meta

Un estado s es meta si contiene la totalidad de proposiciones meta: $G \subseteq s \subseteq S$

Planificación automática

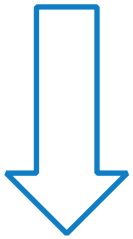
$a \in A,$
$pre(a) \subseteq S$
$add(a) \subseteq S \ - del(a) \subseteq S$
$a\ es\ ejecutable\ en\ un\ estado\ s \subseteq S\ si\ satisface\ pre(a) \subseteq s$
$s' \subseteq S\ es\ el\ estado\ resultante\ de\ aplicar\ a\ en\ s$
$mantiene\ las\ proposiciones\ de\ s, eliminando\ del(a) y\ añadiendo\ add(a)$

$s = \{a, b, c\}$
$G = \{g\}$
$a \in A \mid pre(a) = \{a, b\}, del(a) = \{b, c\}, add(a) = \{f, g\}$
$s' \subseteq S\ resultante\ de\ aplicar\ a\ en\ s\ es:$

?

Planificación automática

Es independiente del dominio



El usuario se limita a modelar el problema, no tiene que construir un resolutor propio.

Planificación automática

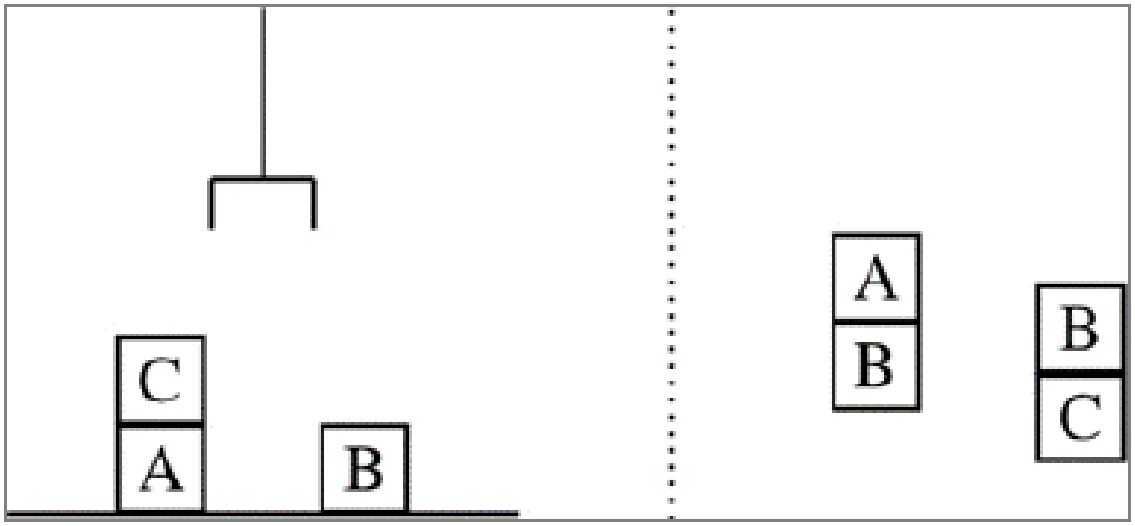
PDDL (*Planning Domain Definition Language*)

Dos ficheros

- Dominio
  - Tipos de objetos
  - Atributos de los objetos
  - Acciones que se pueden realizar
- Problema
  - Objetos
  - Estado inicial
  - Metas
  - Métricas

Planificación automática

El dominio del Mundo de los Bloques



Estado inicial Metas

Planificación automática

Tipos de objetos

Atributos y relaciones entre objetos

Acciones

```
(define (domain BLOCKS)
 (:requirements :strips :typing)
 (:types block)
 (:predicates (on ?x - block ?y - block)
              (ontable ?x - block)
              (clear ?x - block)
              (handempty)
              (holding ?x – block) )
(:action pick-up
         :parameters (?x - block)
         :precondition (and (clear ?x) (ontable ?x) (handempty))
         :effect
         (and (not (ontable ?x))
              (not (clear ?x))
              (not (handempty))
              (holding ?x)))
(:action put-down
         :parameters (?x - block)
         :precondition (holding ?x)
         :effect
         (and (not (holding ?x))
              (clear ?x)
              (handempty)
              (ontable ?x)))
```

Fichero de dominio del Mundo de los Bloques

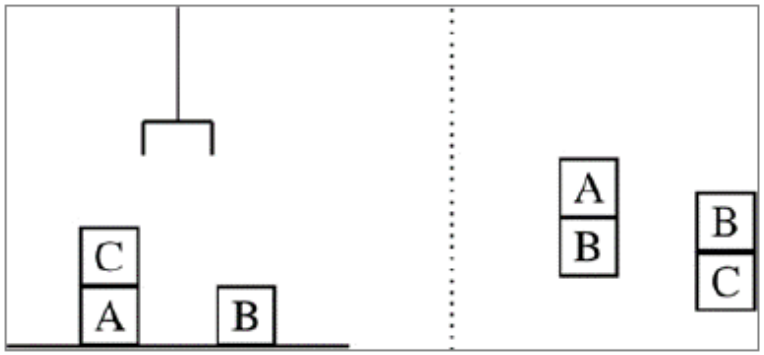
Planificación automática

Fichero de dominio del Mundo de los Bloques

```
(:action stack
        :parameters (?x - block ?y - block)
        :precondition (and (holding ?x) (clear ?y))
        :effect
        (and (not (holding ?x))
             (not (clear ?y))
             (clear ?x)
             (handempty)
             (on ?x ?y)))
 (:action unstack
        :parameters (?x - block ?y - block)
        :precondition (and (on ?x ?y) (clear ?x) (handempty))
        :effect
        (and (holding ?x)
             (clear ?y)
             (not (clear ?x))
             (not (handempty))
             (not (on ?x ?y)))))
```

Planificación automática

Fichero de problema del Mundo de los Bloques



Estado inicial Metas

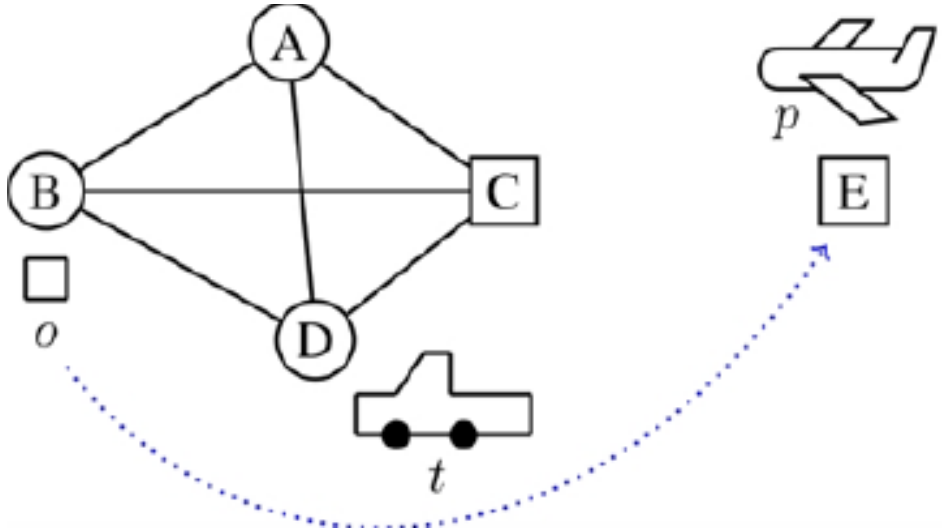
```
(define (problem BLOCKS-EX)
(:domain BLOCKS)
(:objects A B C - block)
(:init      (CLEAR C)
     (CLEAR B)
     (ONTABLE A)
     (ONTABLE B)
     (ON C A)
     (HANDEMPTY))
(:goal (AND (ON A B) (ON B C)))
)
```

Solución óptima:

unstack C A → put-down C → pick-up B → stack B C → pick-up A → stack A B

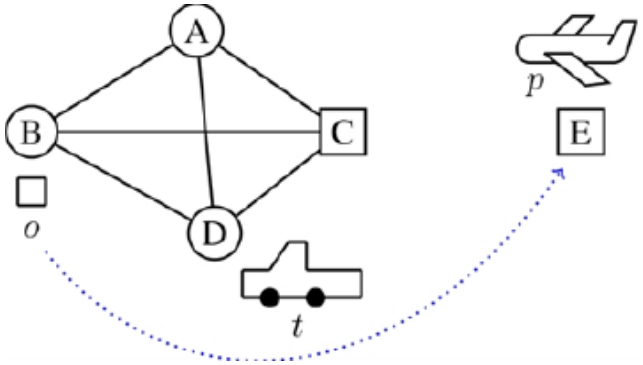
Planificación automática

Problema de logística en el que:
- los camiones sólo pueden moverse entre localizaciones de la misma ciudad
- los aviones pueden volar a cualquier localización con aeropuerto (Cuadrados)



La tarea es llevar el paquete de B a E

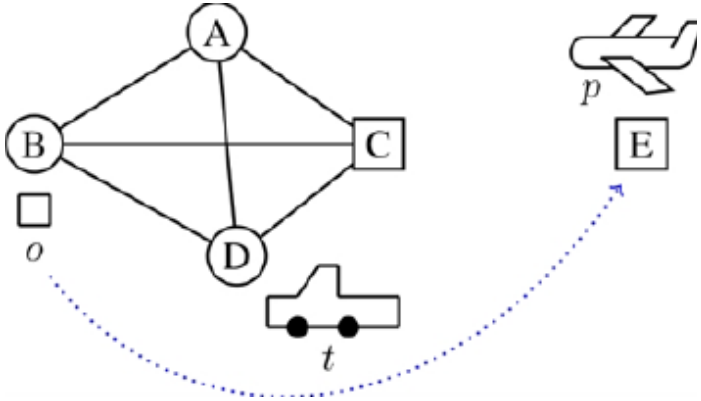
Planificación automática



fichero de dominio

```
(define (domain logistics)
(:requirements :typing)
(:types  truck airplane - vehicle
         package vehicle - thing
         airport - location
         city location thing - object)

(:predicates  (in-city ?l - location ?c - city)
              (at ?obj - thing ?l - location)
              (in ?p - package ?veh - vehicle))

(:action drive
         :parameters (?t - truck ?from ?to - location ?c - city)
         :precondition  (and (at ?t ?from)
                             (in-city ?from ?c)
                             (in-city ?to ?c))
         :effect        (and (not (at ?t ?from))
                             (at ?t ?to)))
```

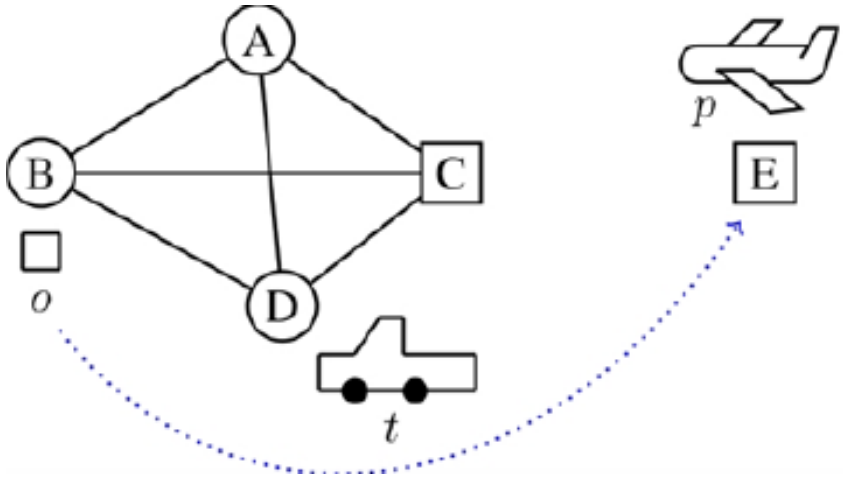
Planificación automática



fichero de dominio

```
(:action fly
      :parameters   (?a - airplane ?from ?to - airport)
      :precondition  (at ?a ?from)
      :effect       (and (not (at ?a ?from))
                    (at ?a ?to)))
(:action load
      :parameters   (?v - vehicle ?p - package ?l - location)
      :precondition  (and (at ?v ?l)
                    (at ?p ?l))
      :effect       (and (not (at ?p ?l))
                    (in ?p ?v)))

(:action unload
      :parameters   (?v - vehicle ?p - package ?l - location)
      :precondition  (and (at ?v ?l)
                    (in ?p ?v))
      :effect       (and (not (in ?p ?v))
                    (at ?p ?l)))
)
```

Planificación automática

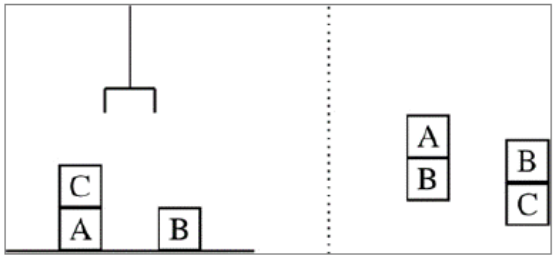


fichero de problema

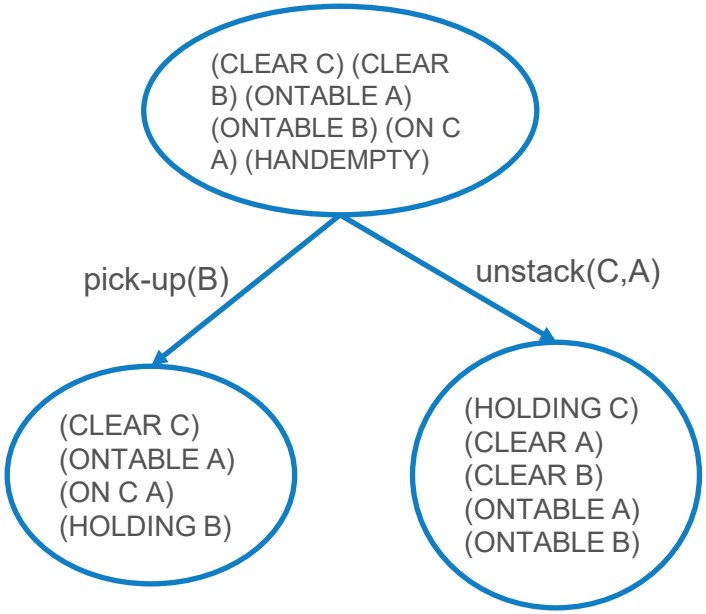
```
(define (problem logistics-ex)
(:domain logistics)
(:objects  apn1 - airplane
           tru1 - truck
           obj1 - package
           aptC aptE - airport
           posA posB posD - location
           cit1 cit2 - city)

(:init  (at apn1 aptE)
        (at tru1 posD)
        (at obj11 posB)
        (in-city aptC cit1)
        (in-city aptE cit2)
        (in-city posA cit1)
        (in-city posB cit1)
        (in-city posD cit1))
(:goal  (at obj11 aptE))
)
```

Planificación como búsqueda heurística

- Estado inicial → Nodo raíz del árbol
- Se aplican acciones para generar estados sucesores
- Búsqueda avariciosa, A*…

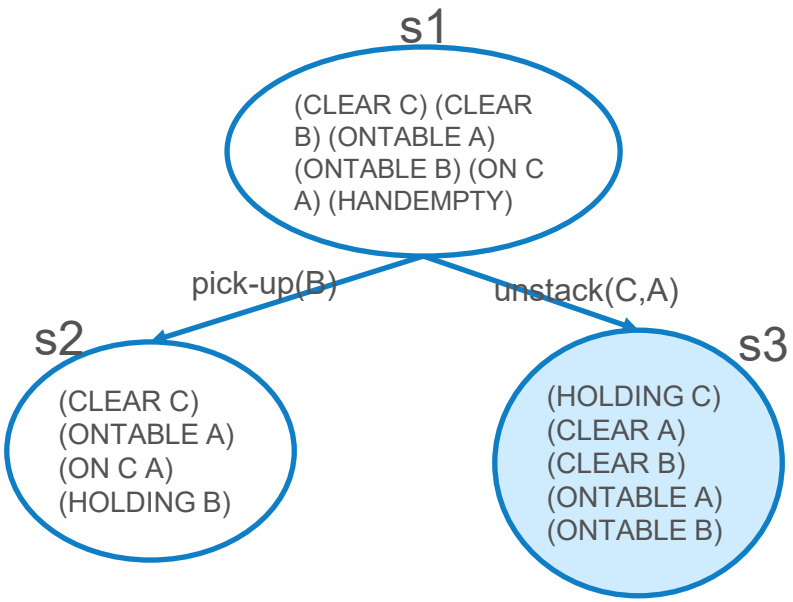


Estado inicial Metas

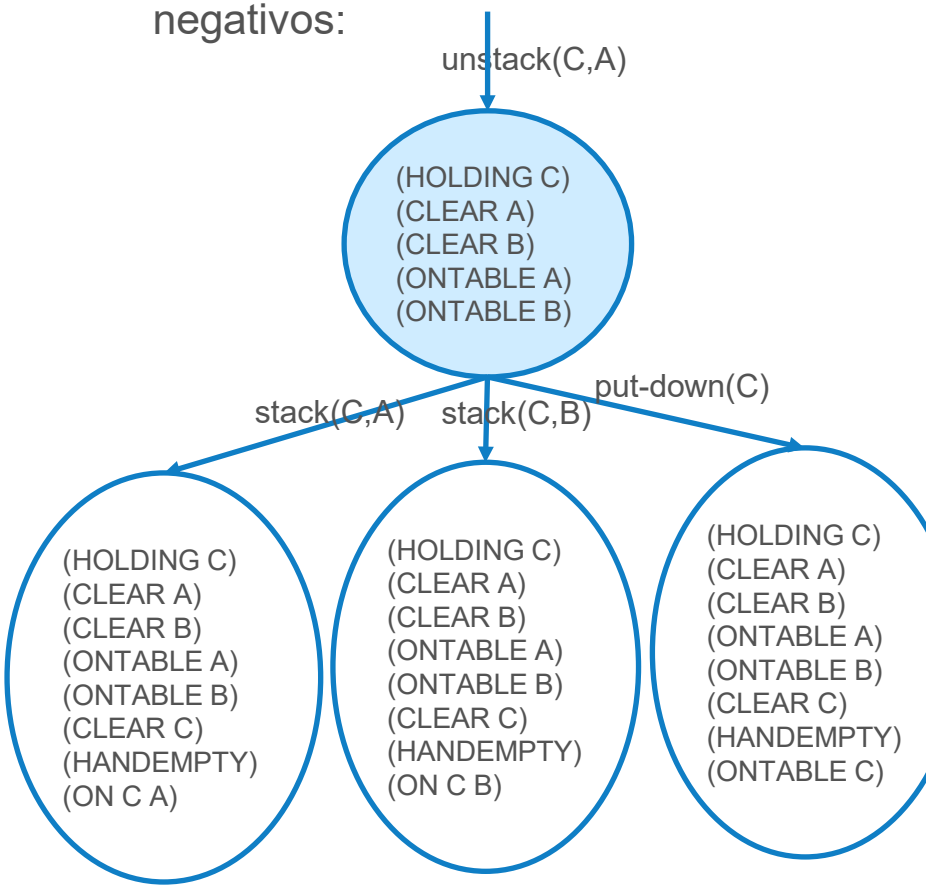


Planificación como búsqueda heurística

- Diseño de heurística: independiente del dominio, relajación del problema
  - Ignorar efectos negativos: computar las acciones aplicables y se añaden todos los efectos positivos hasta que las metas aparezcan.



Resultado de acciones para el cálculo de la heurística de s3 ignorando efectos negativos:



Se sigue explorando el árbol hasta que aparece un estado s' con todas las proposiciones meta, h se calcula como el número de transiciones desde s3 hasta s'.

Ejemplo

El problema: “Tener una tarta y comerla también” (Russell y Norvig, 2014)

Estado inicial:
Tener(tarta) Λ ¬ Comida(tarta)

Meta:
Tener(tarta)ΛComida(tarta)

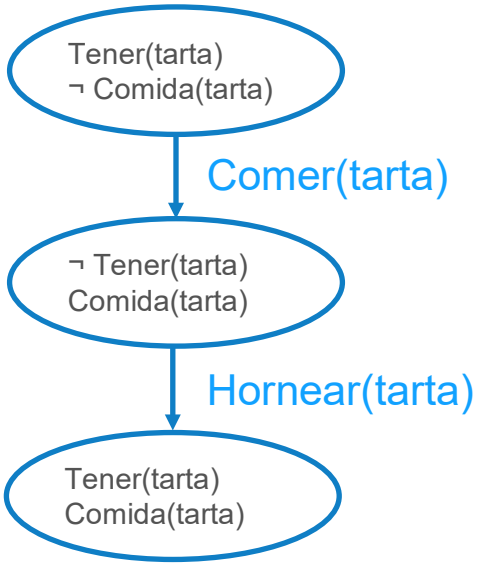
Acciones:
Comer(tarta)
Precondiciones: Tener(tarta)
Efectos: ¬ Tener(tarta) ΛComida(tarta)
Hornear(tarta):
Precondiciones: ¬ Tener(tarta)
Efectos: Tener(tarta)

El problema: "Tener una tarta y comerla también" (Russell y Norvig, 2014)

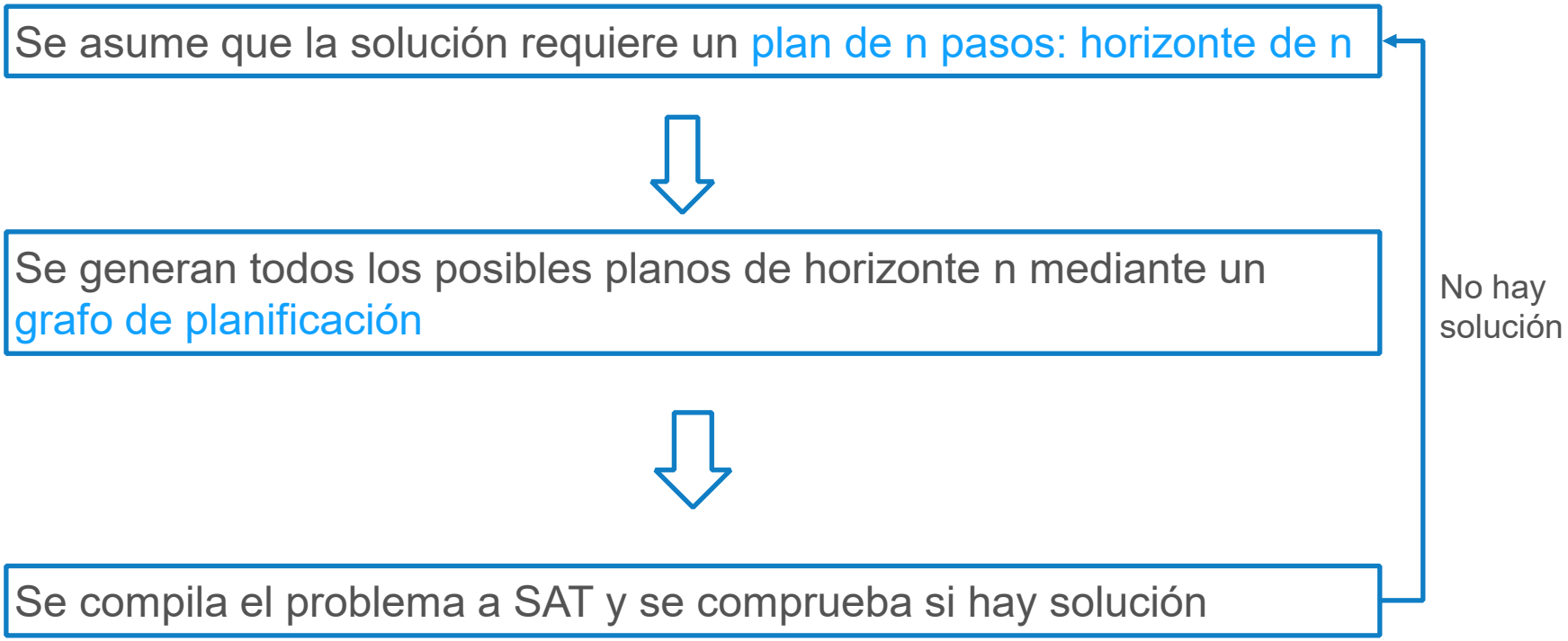
Estado inicial:
Tener(tarta) Λ ¬ Comida(tarta)

Meta: Tener(tarta)ΛComida(tarta)

Acciones:
Comer(tarta)
Precondiciones: Tener(tarta)
Efectos: ¬ Tener(tarta) ΛComida(tarta)
Hornear(tarta):
Precondiciones: ¬ Tener(tarta)
Efectos: Tener(tarta)



Planificación con SAT



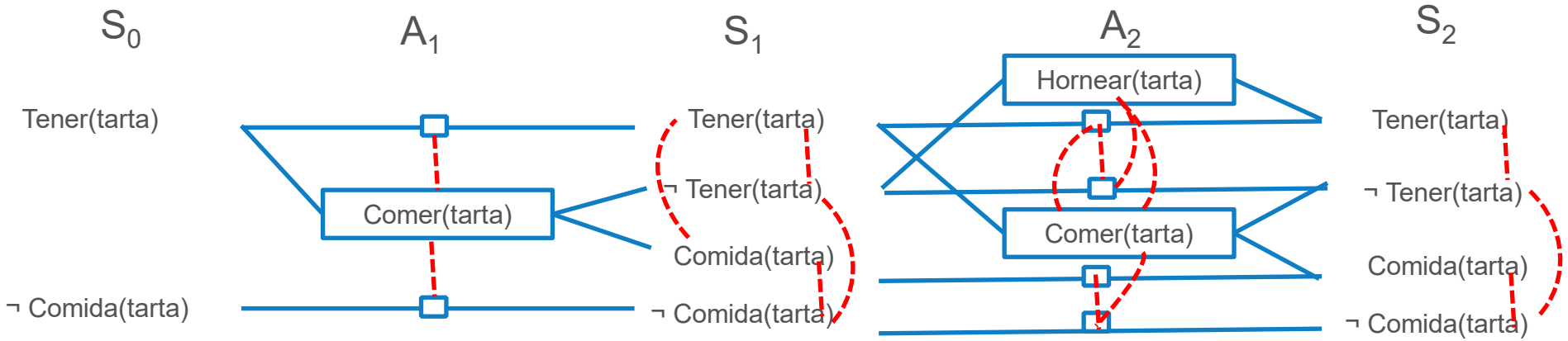
Planificación con SAT

El problema: "Tener una tarta y comerla también" (Russell y Norvig, 2014)

Estado inicial:
Tener(tarta) Λ ¬ Comida(tarta)

Meta:
Tener(tarta)ΛComida(tarta)

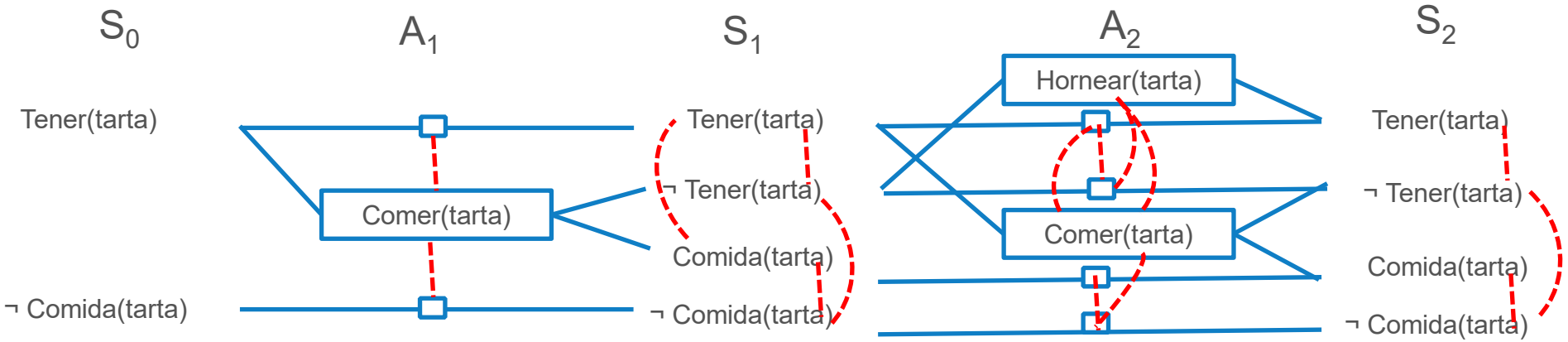
Acciones:
Comer(tarta)
Precondiciones: Tener(tarta)
Efectos: ¬ Tener(tarta) ΛComida(tarta)
Hornear(tarta):
Precondiciones: ¬ Tener(tarta)
Efectos: Tener(tarta)



Planificación con SAT

Grafo de planificación:

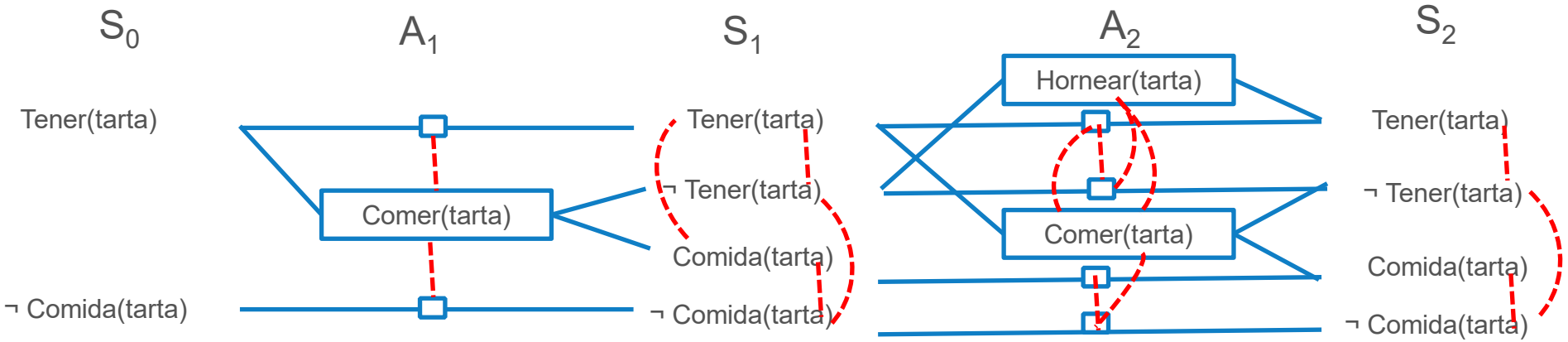
- Nivel 0: formado por las proposiciones del estado inicial
- Acciones del nivel 1: acciones aplicables dadas las proposiciones del nivel 0
- Proposiciones del nivel 1: proposiciones del nivel 0 + las añadidas por los efectos de las acciones del nivel 1.



Planificación con SAT

Grafo de planificación:

- Cada nivel $\mathbf{A_i}$ contiene las acciones aplicables según $\mathbf{S_{i-1}}$ y las restricciones que indican que dos acciones no pueden tener lugar al mismo momento.
- Cada nivel $\mathbf{S_i}$ contiene todas las proposiciones que podrían resultar posibles en dicho nivel y las restricciones que indican que dos proposiciones a la vez no son posibles.



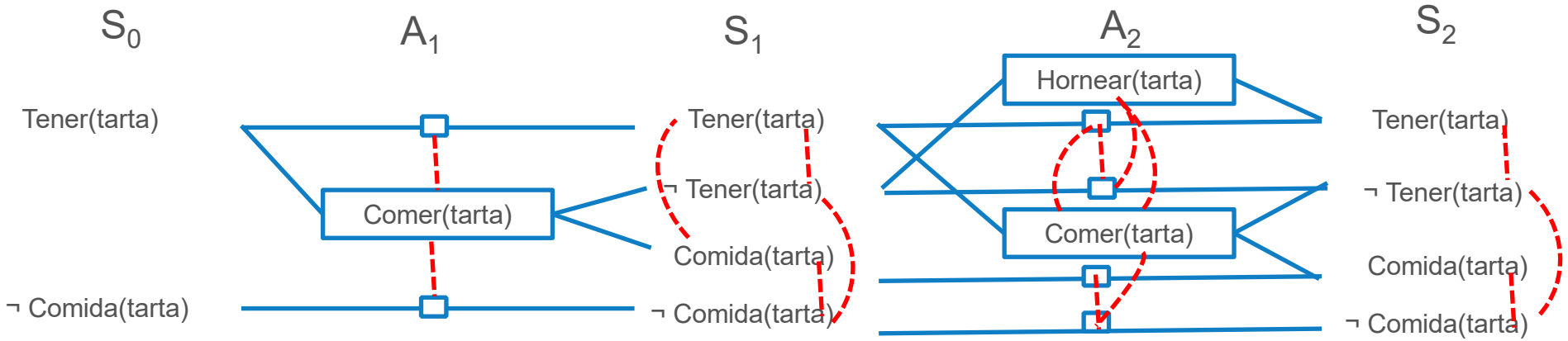
Planificación con SAT

Condiciones de exclusividad mutua entre acciones:

- Efectos de inconsistencia: Una acción niega el efecto de la otra.
- Interferencia: Una acción niega una precondición de la otra
- Necesidades competitivas: Una precondición de una acción es mutuamente exclusiva con una precondición de la otra.

Condiciones de exclusividad mutua entre proposiciones (soporte inconsistente):

- Una proposición es la negación de otra.
- Las acciones que añaden las proposiciones son mutuamente exclusivas.



Planificación con SAT

- La compilación en SAT es sencilla.
- Por cada proposición y acción en cada nivel hay una variable

Proposiciones de I son ciertas en el nivel 0: si p_0 representa la proposición $p \in I$ en el nivel 0, entonces hay que añadir (p_0)

Proposiciones de G son ciertas en el nivel n: si p_n representa la proposición $p \in G$ en el nivel n, entonces hay que añadir (p_n)

Las relaciones de mutua exclusividad entre dos nodos *p* y *q:*
$\neg(p \wedge q) = (\neg p \vee \neg q)$

Las acciones ejecutadas en un nivel dado implican:
- Todas las precondiciones en el nivel anterior son ciertas
- Todos los efectos positivos en ese nivel son ciertos
- Los efectos negativos en ese nivel son falsos.

Búsqueda hacia atrás - Grafo de planificación

Generación del Grafo de Planificación

- Iterativamente añade niveles al gráfico de planificación
- Una vez que las proposiciones que conforman el objetivo se muestran *no-mutex* (sin exclusividad mutua) en un nivel, el algoritmo comprueba si existe una solución a partir de ese nivel.

Búsqueda hacia atrás de la solución

❑El estado inicial es el último nivel del gráfico con el conjunto de proposiciones (no mutex) que cumplen el objetivo.

❑Las acciones disponibles en cada nivel *i* son cualesquiera de ellas dentro del conjunto de acciones en A_{i-1} , cuyos efectos cubran el objetivo del nivel $\boldsymbol{S_i}$, y que no sean mutex entre ellas.

❑El estado que resulta corresponde al nivel S_{i-1} , compuesto por las precondiciones de las acciones seleccionadas en A_{i-1} que se convierten en los nuevos objetivos.

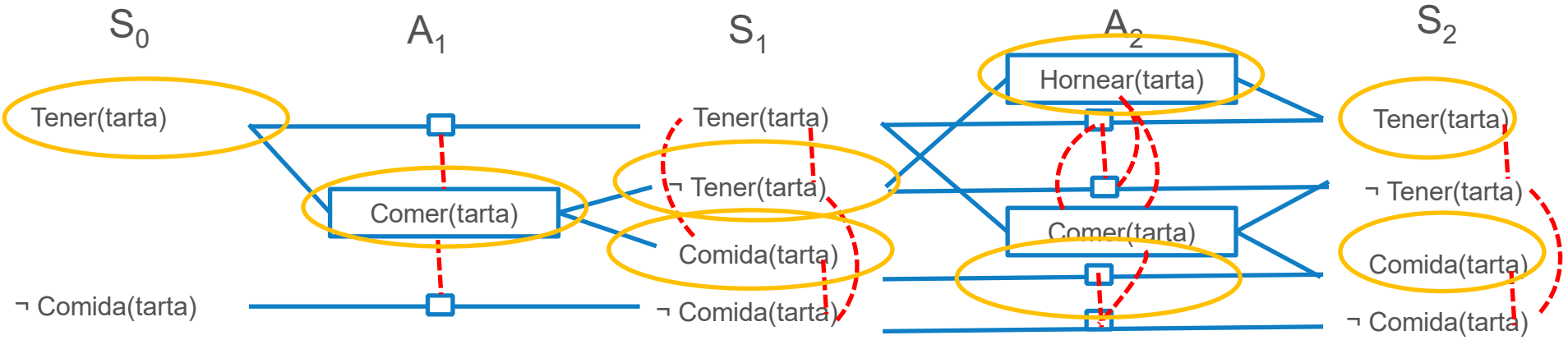
❑El objetivo es llegar al primer nivel S_0

Búsqueda hacia atrás - Grafo de planificación

Estado inicial:
Tener(tarta) Λ ¬ Comida(tarta)

*Meta:*Tener(tarta)ΛComida(tarta)

Acciones:
Comer(tarta)
Precondiciones: Tener(tarta)
Efectos: ¬ Tener(tarta) ΛComida(tarta)
Hornear(tarta):
Precondiciones: ¬ Tener(tarta)
Efectos: Tener(tarta)



Referencias

- Russell, S. y Norvig, P (2014). **Artificial Intelligence**: A **Modern Approach**, 3ª edición, Pearson.

Desafío 7: Referente al tema 4, en menos de 100 palabras responde a las siguientes cuestiones: ¿Qué quiere decir que la planificación automática sea independiente del dominio? ¿Qué ventaja tiene esta independencia del dominio?

Que la planificación automática sea independiente del dominio quiere decir que los planificadores han de ser capaces de resolver problemas de planificación diferentes sin intervención del usuario, que sólo se limita a modelar el problema. La ventaja es que no hay que construir un resolutor para cada problema, sino únicamente modelar cada problema como un problema de planificación y aplicar planificadores o resolutores existentes para resolverlo.

Desafío 8: Referente al tema 5, responde brevemente a las siguientes cuestiones ¿Qué características tienen los juegos en que se aplica Minimax? ¿Qué asume Minimax?

Desafío 9: Si tenemos un problema definido como se indica a continuación:

Estado inicial:

¬Tener(recambio) AND ¬Puesto(recambio)

Meta:

Puesto(recambio)

Acciones:

Poner(recambio):

Precondiciones: Tener(recambio) AND ¬Puesto(recambio)

Efectos: ¬ Tener(recambio) AND Puesto(recambio)

Comprar(recambio):

Precondiciones: ¬ Tener(recambio)

Efectos: Tener(recambio)

Formulario con los desafíos:
https://forms.office.com/r/wwYGWh9ZGp

¿Qué acciones se pueden aplicar a partir del estado inicial? ¿qué proposiciones definirían el estado resultante?

¿Dudas?

¡Muchas gracias por vuestra atención!

¡Feliz y provechosa semana!

www.unir.net