Agentes Inteligentes

Javier García

Departamento de Electrónica y Computación
Universidad de Santiago de Compostela

November 18, 2021

Part V

Planificación Automática

Contenidos

¿Qué es la planificación?

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

¿Qué tienen en común?

Búsqueda en el espacio de problemas

- **Estado o situación**: descripción instantánea
- **Acción u Operador**: transformación de un estado en otro
- **Estado inicial**: situación de partida
- **Objetivo o meta**: descripción de condiciones que se tienen que dar para considerar por terminado el proceso
- **Plan**: secuencia de operadores que permiten pasar del estado inicial a un estado en el que se cumplan los objetivos
- **Heurísticas**: conocimiento que permite obtener eficientemente el plan

Planificación en IA (I)

La planificación es un proceso explícito **deliberativo** que selecciona y organiza las acciones del sistema para alcanzar sus metas anticipándose a los resultados esperados de las acciones

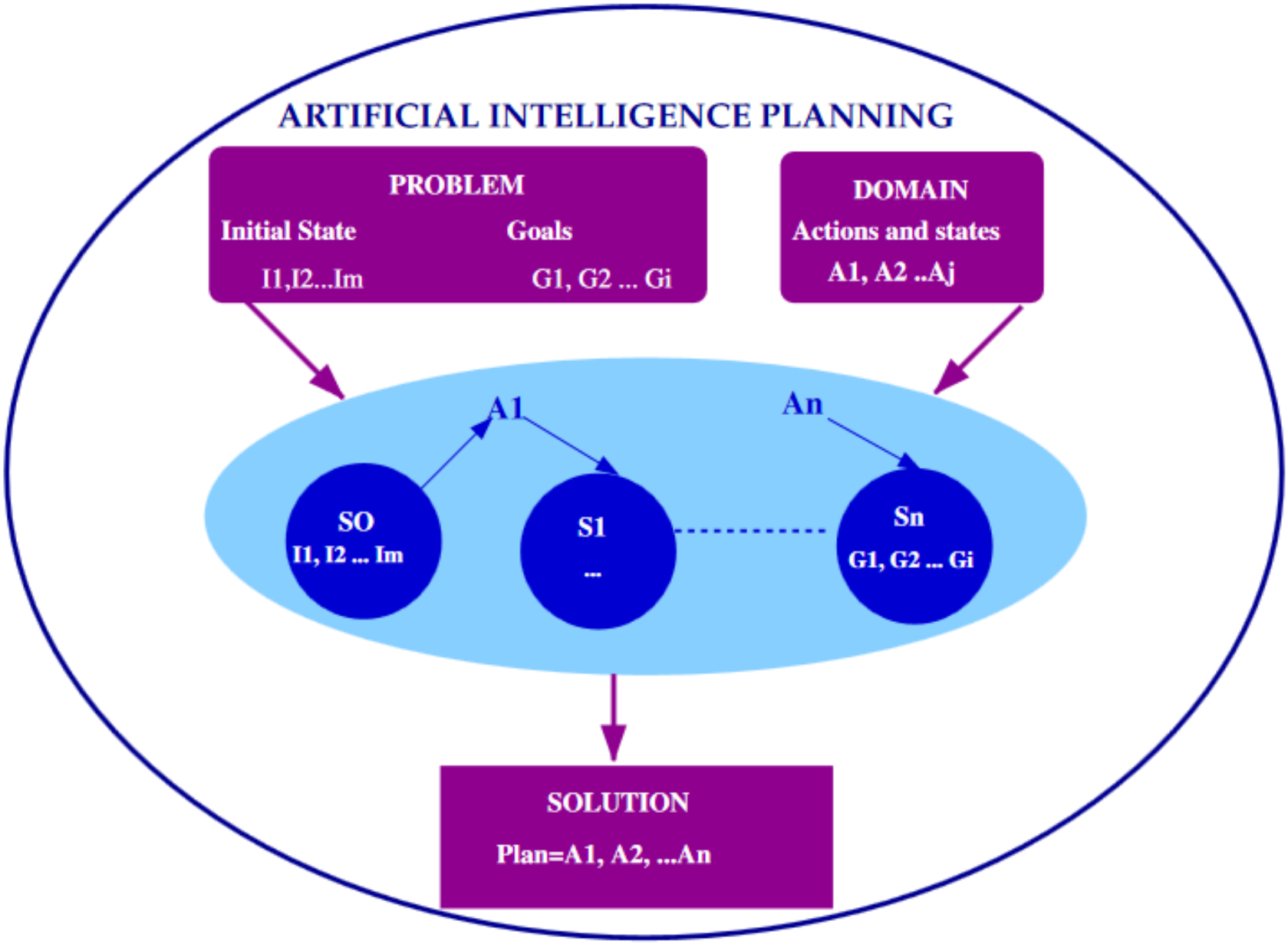
- **Plan**: secuencia de acciones individuales que permiten alcanzar una meta a partir de una situación inicial
- **Planificación en IA**: procedimiento automático para encontrar un plan en un problema concreto
- La **Planificación Automática** estudia computacionalmente este proceso deliberativo
- **La principal diferencia entre búsqueda y planificación es la representación**

Planificación en IA (II)

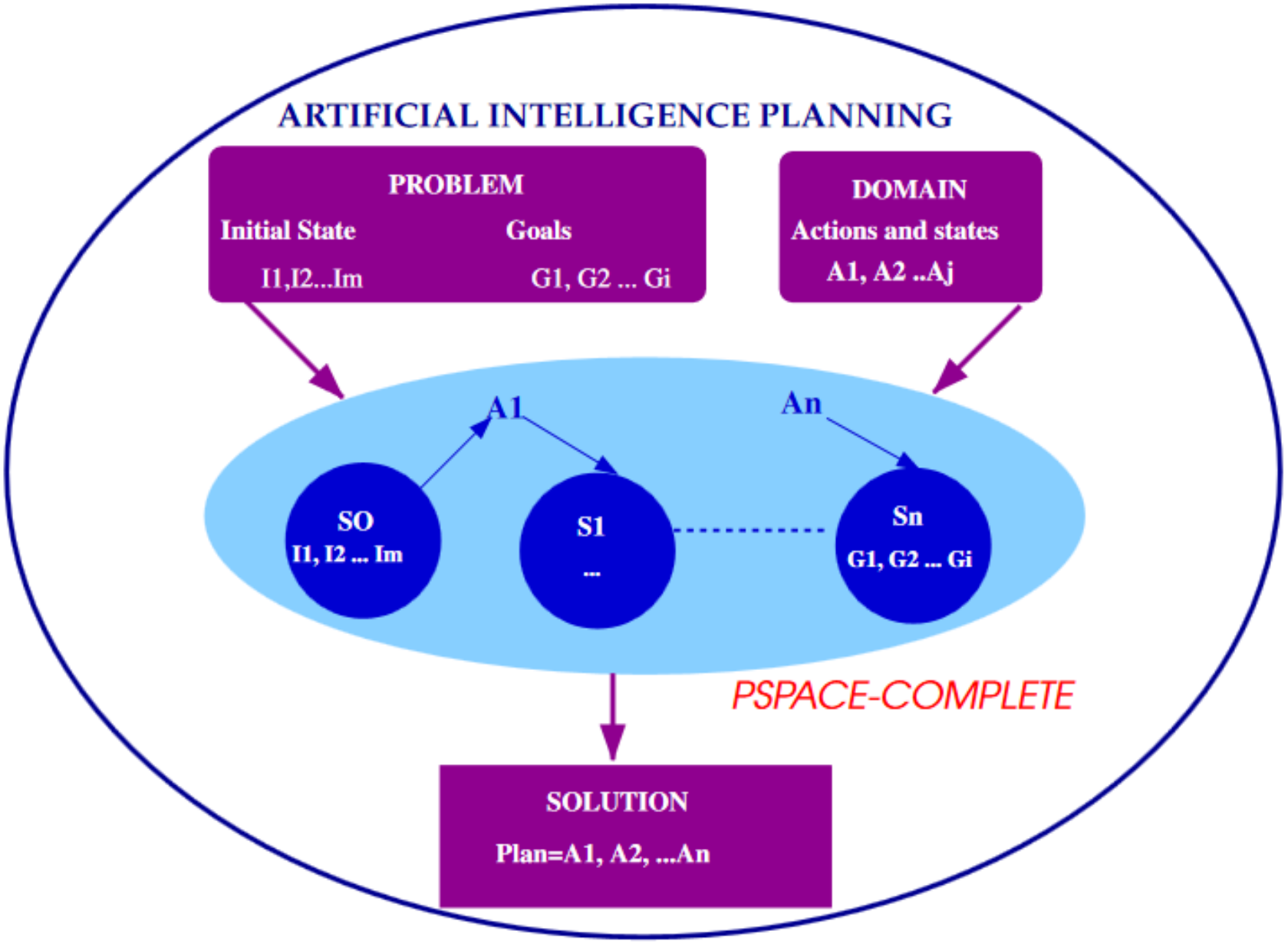
Planificación Automática

- **Dados**:
  - un modelo del dominio (lenguaje PDDL): conjunto de acciones
  - un problema: un estado inicial, un conjunto de metas, una métrica
- **Obtener**: un conjunto ordenado de acciones que consiguen las metas desde el estado inicial (intentando optimizar la métrica)

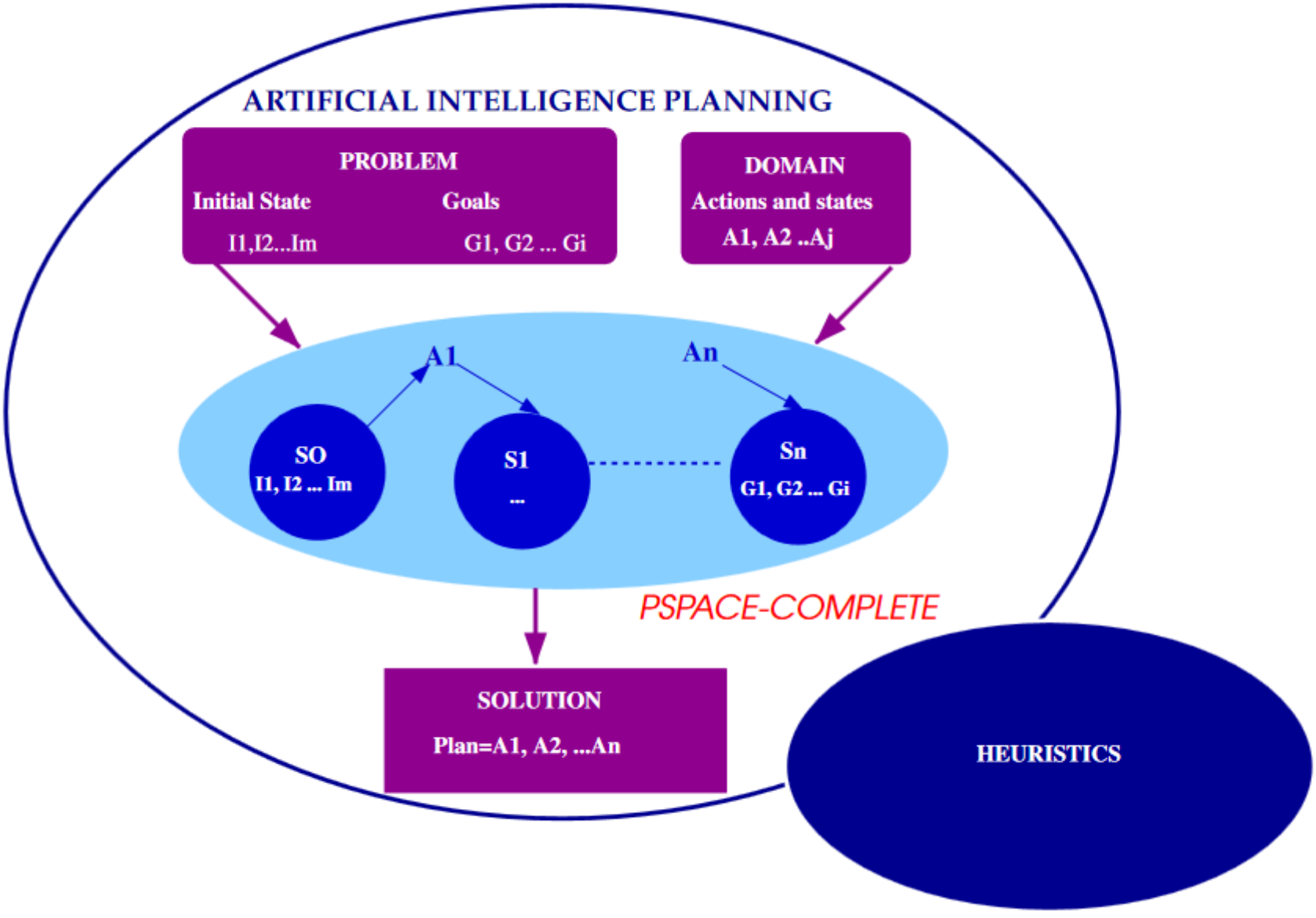
Planificación en IA (III)



Planificación en IA (IV)



Planificación en IA (V)



Ejemplo en turismo (I)

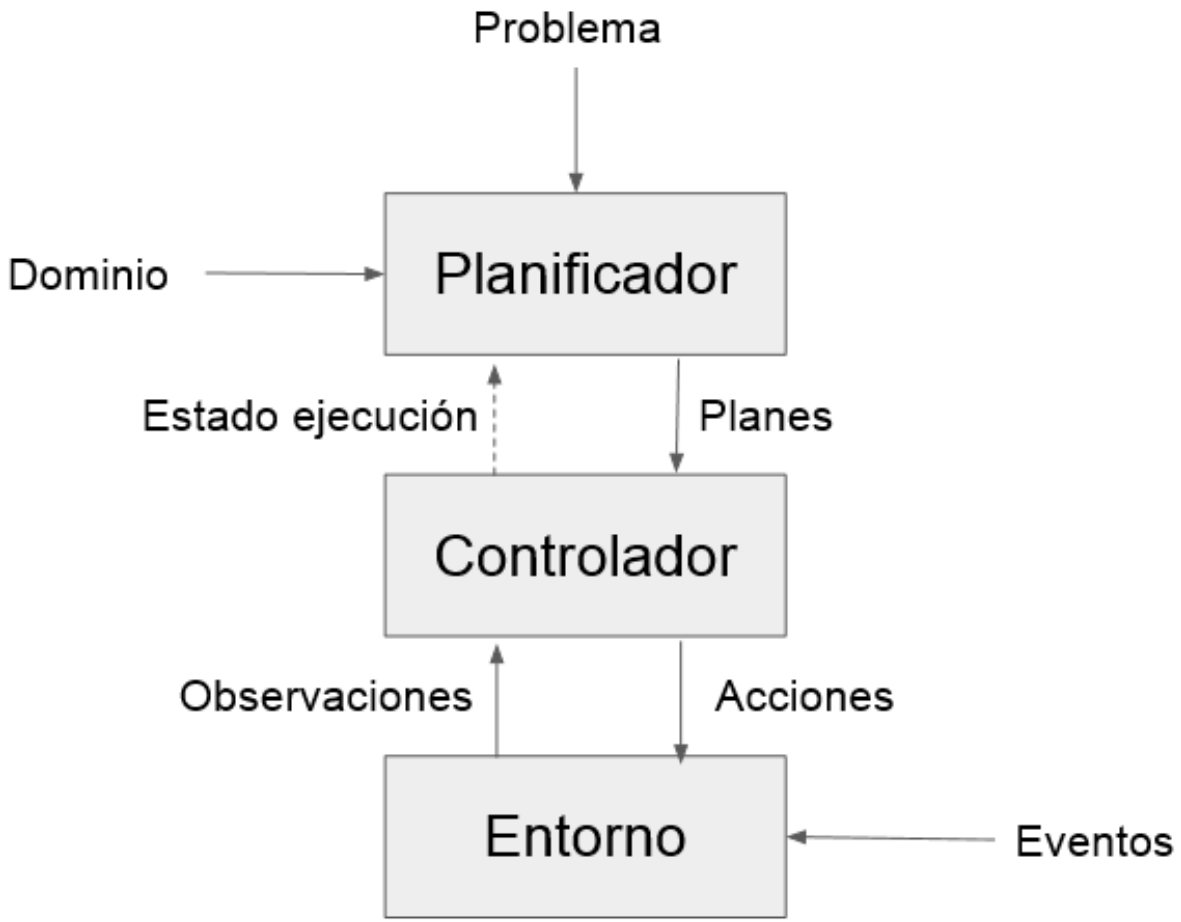
- **Estado**:
  - situación de una persona, restricciones de precio, tiempo,...
  - precios, horarios y disponibilidad de los billetes de avión, tren, barco, autobús, hoteles, etc...de diferentes compañías
- **Operadores**: volar en un determinado vuelo, viajar en un determinado tren, coger un taxi, ir en coche, alojarse en un hotel, alojarse en una casa rural,...
- **Estado inicial**: estamos en Galicia
- **Meta**: quiero pasar una semana en San Francisco

Ejemplo en turismo (II)

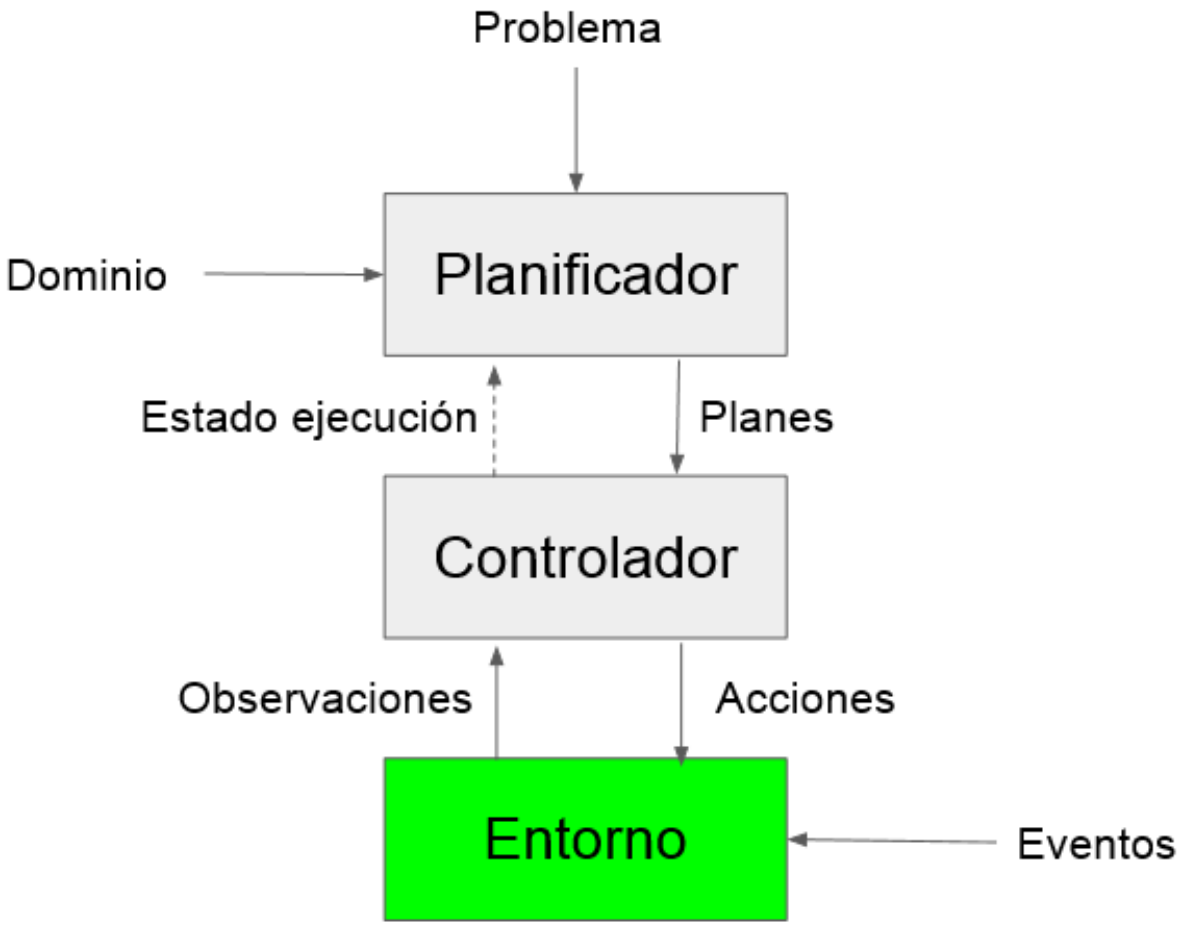
- **Plan**:
  - coger taxi(Casa,AeropuertoSCQ)
  - coger un vuelo(AeropuertoSCQ,AeropuertoJFK)
  - coger un vuelo(AeropuertoJFK,AeropuertoSFO)
  - coger_limusina(AeropuertoSFO,HotelSheraton)
  - estar_en_hotel(HotelSheraton)
- **Heurísticas**: tiempo de viaje de cada transporte,...

Robótica autónoma

Modelo conceptual (I)



Modelo conceptual (II)

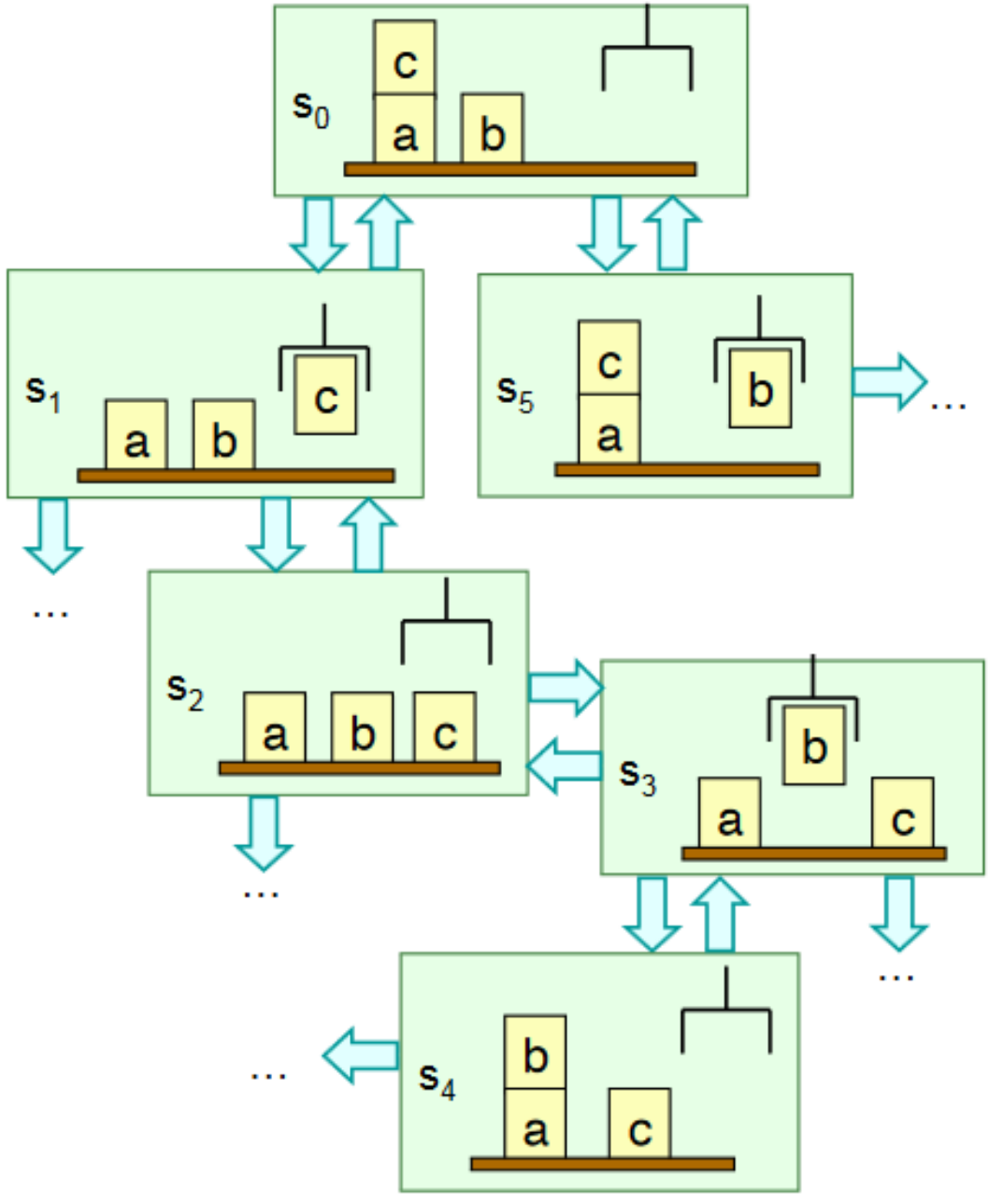


Modelo conceptual (III)

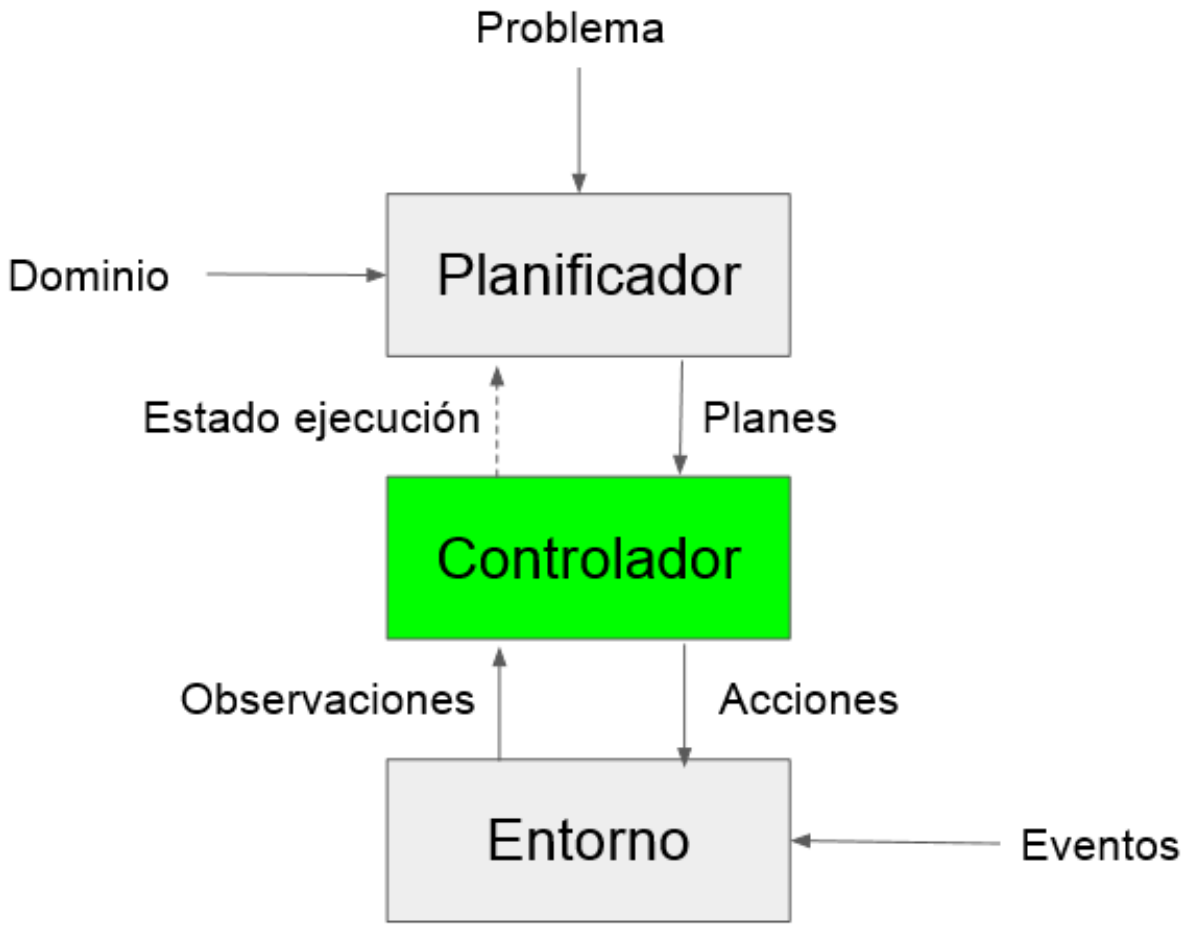
- El entorno (estático) se describe como $\sum = (S, A, \gamma)$
  - S: Conjunto finito de estados
  - A: Conjunto finito de acciones
  - γ: Función de transición $S \times A \rightarrow S$

Modelo conceptual (IV)

- $S = \{s_0, s_1, s_2, \ldots, s_{22}\}$
- $A = \{$*quitar c de a, poner c en mesa*$, \ldots\}$
- γ: Ver las flechas

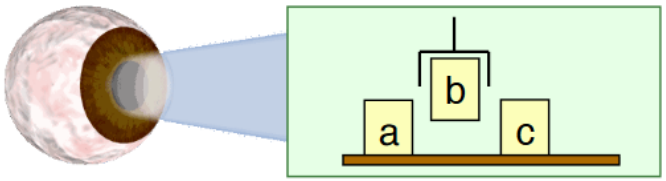


Modelo conceptual (V)

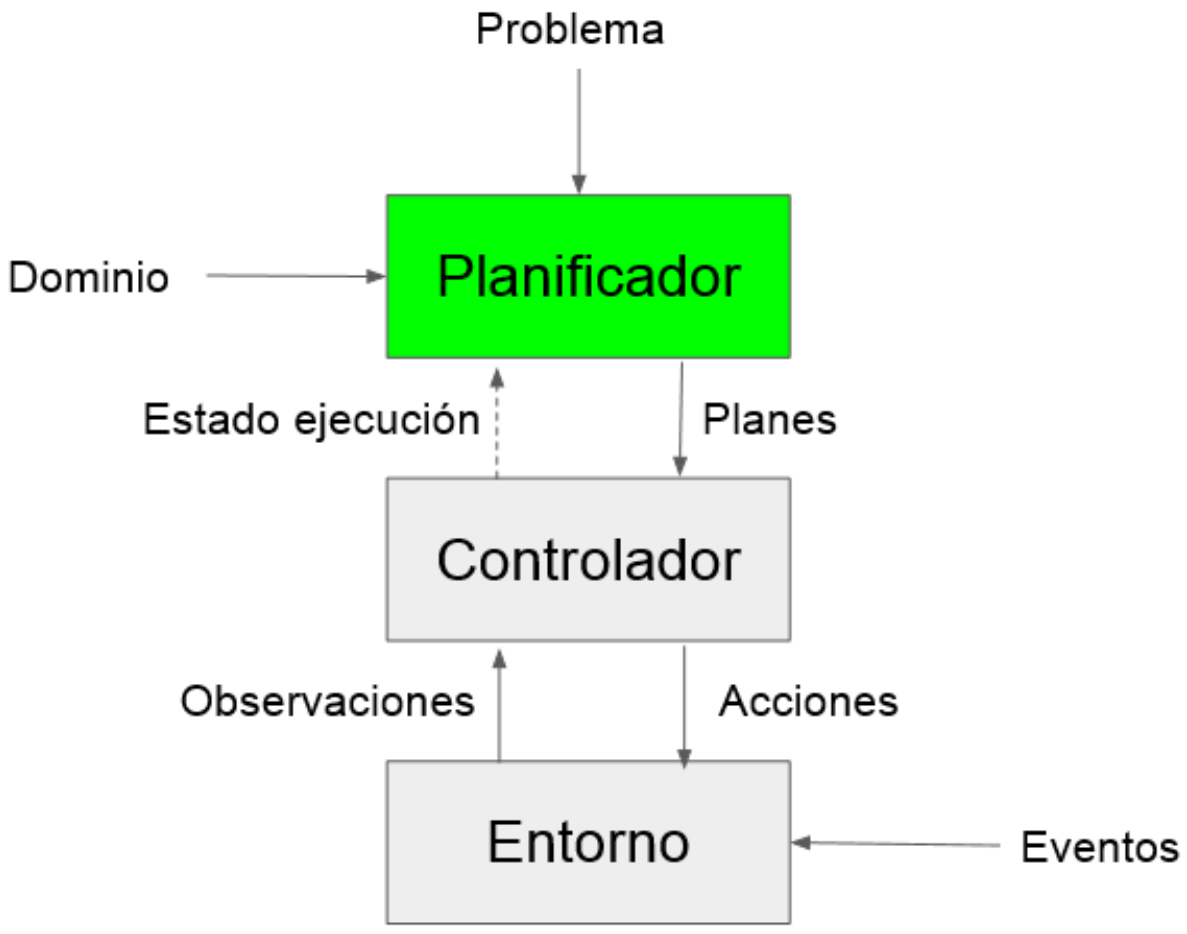


Modelo conceptual (VI)

- El controlador dado un estado/observación devuelve una acción de acuerdo a un plan
  - Estado: Si el entorno es totalmente observable
  - Observación: Si el entorno es parcialmente observable
- Puede trabajar *online* o *offline*
  - *Online*: Continuamente monitoriza el estado del entorno y busca discrepancias entre el estado esperado y percibido. Si las detecta puede ser necesario replanificar o reparar el plan en curso
  - *Offline*: Ejecuta el plan inicialmente generado con independencia de si las acciones conducen a los estados esperados

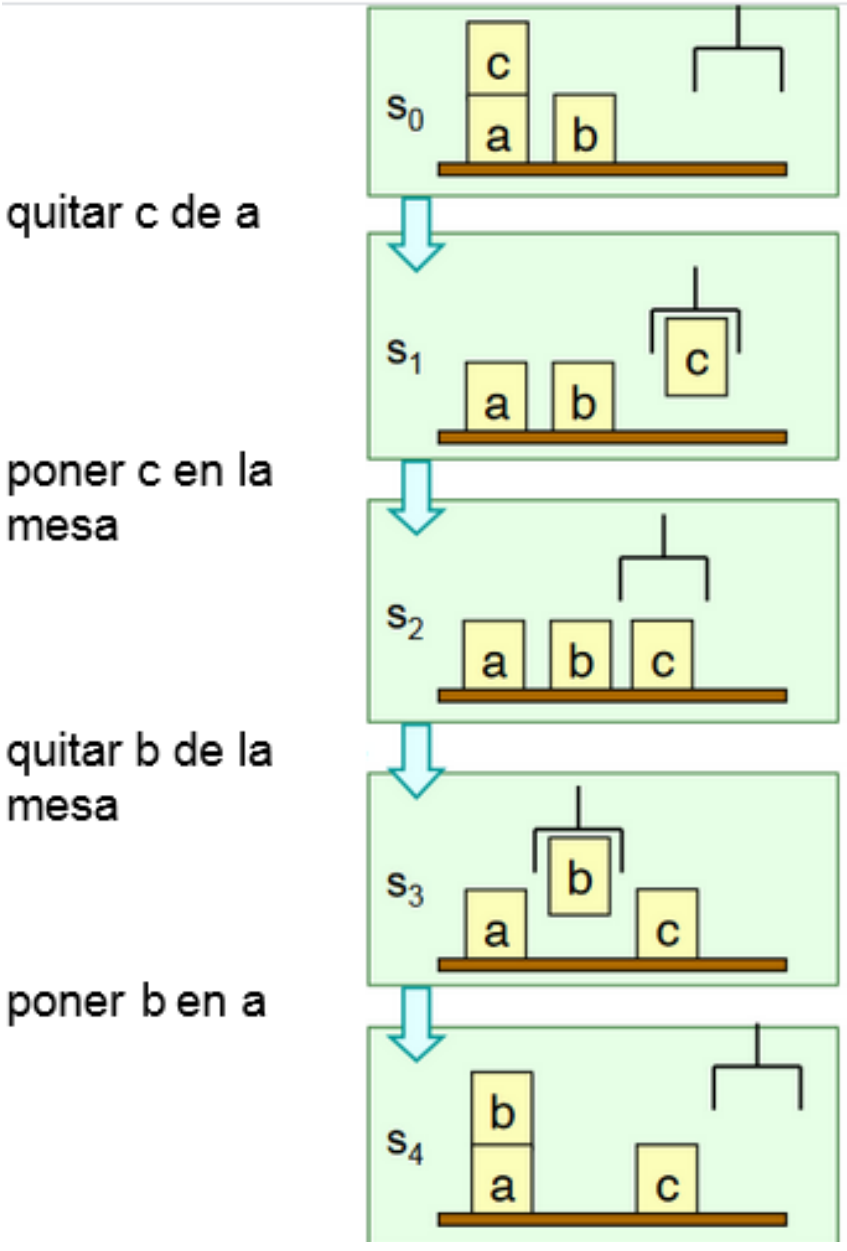


Modelo conceptual (VII)



Modelo conceptual (VIII)

- Dominio: descripción de $\sum$
  - No requiere enumerar todos los posibles estados y acciones
- Problema:
  - Estado inicial (e.g., s_0)
  - Meta o estado final (e.g., s_4)
- Genera un conjunto de acciones/instrucciones para el controlador



Tipos de problema

- Conjunto de estados finito
- Completamente observable
- Determinista
- Estático
- Metas restringidas
- Planes secuenciales
- Tiempo implícito
- Planificación *off-line*

Representación en planificación

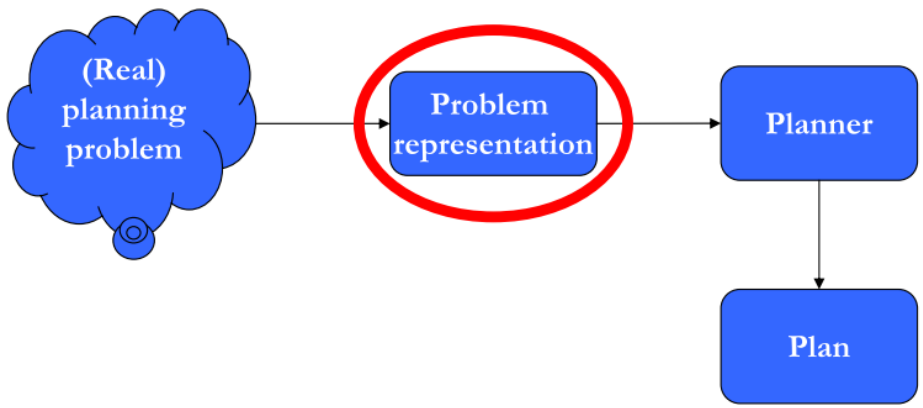
Javier García

Departamento de Electrónica y Computación
Universidad de Santiago de Compostela

November 18, 2021

Representación en planificación (I)

- Para que el ordenador pueda resolver problemas, hace falta decirle qué tiene que resolver en algún lenguaje (al igual que a nosotros)
- Existen muchas formas de suministrar esa información
- La más empleada en planificación automática es la **lógica de predicados**
- Así se representan los estados y los operadores
- La lógica de predicados permite representar las cuestiones ciertas o falsas del mundo mediante: términos, predicados, conectivas, y cuantificadores
- No hay una representación única y válida; cada persona representa los dominios de forma diferente



Representación en planificación (II)

- Necesitamos representar:
  - Estados (estado inicial, final, y estado actual)
  - Acciones
- Múltiples formas de representación pero la lógica de predicados es la más utilizada en planificación

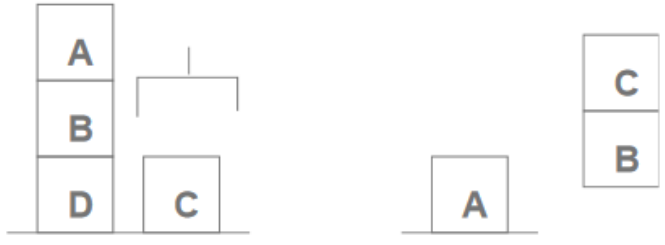
Representación de los estados

- Los estados se describen mediante:
  - Un conjunto de posibles objetos:
    - yo, coche, tren, madrid, galicia
  - Un conjunto de predicados que relacionan objetos:
    - at(user,place), at(monument, place), links(transport, place1, place2), on-table(x),...
  - Un estado se representa por un conjunto de estados instanciados
    - at(yo,madrid),at(bernabeu, castellana), links(C6, leganes, atocha),...
  - Hipótesis del mundo cerrado: Lo que no es explícitamente establecido como verdadero, es falso

Estado en el mundo de los bloques

- Se podrían representar los siguientes predicados:
  - **encima(x,y)**: el bloque x está encima del y
  - **en-mesa(x)**: el bloque x está encima de la mesa
  - **libre(x)**: el bloque x no tiene ningún bloque encima
  - **sujeto(x)**: el brazo del robot tiene cogido al bloque x
  - **brazo-libre**: el brazo del robot no tiene cogido a ningún bloque

Estado inicial:
encima(A,B),encima(B,D),en-mesa(D),
en-mesa(C),libre(A),libre(C),brazo-libre
Metas: en-mesa(A),encima(C,B)



Estado inicial Metas

Representación de los operadores

- Para definir una acción/operador necesitamos saber en qué estados se puede ejecutar y el estado resultante después de ejecutar esa acción
- Representación STRIPS:
  - Objetos envueltos en la acción
  - Cuándo (bajo qué condiciones) la acción puede ser ejecutada
    - Precondiciones: Conjunto de predicados que deben ser ciertos para que la acción pueda ejecutarse
  - Los efectos de la acción
    - añadidos: predicados que pasan a ser ciertos por la ejecución de la acción (hay que añadirlos al estado)
    - borrados: predicados que dejan de ser ciertos por la ejecución del operador (hay que borrarlas del estado)

Operadores en el mundo de los bloques

- levantar(?x): en-mesa(?x) libre(?x) brazo-libre
  sujeto(?x)
  ¬en-mesa(?x) ¬libre(?x) ¬brazo-libre
- dejar(?x): sujeto(?x)
  en-mesa(?x)
  ¬sujeto(?x)
- poner(?x ?y): sujeto(?x) libre(?y)
  encima(?x ?y) libre(?x) brazo-libre
  ¬sujeto(?x) ¬libre(?y)
- quitar(?x ?y): encima(?x ?y) libre(?x) brazo-libre
  sujeto(?x) libre(?y)
  ¬encima(?x ?y) ¬brazo-libre ¬libre(?x)

Problemas y dominios

- El conjunto de predicados y operadores sin instanciar permite representar una gran variedad de problemas
- En cada problema el estado inicial y las metas se representan con un conjunto de objetos y predicados diferentes
- Se separa la información común de la que no lo es
  - Dominio: lista de predicados, tipos de objetos, y operadores
  - Problema: objetos, estado inicial y metas con predicados instanciados

Desde STRIPS a PDDL

- El lenguaje de STRIPS es muy simple
- Se amplió el lenguaje, generando ADL, que incorpora, entre otros:
  - cuantificación universal y existencial
  - efectos condicionales
- Desde 1998 se ha venido organizando una competición de planificadores, IPC (*International Planning Competition*)
- Se necesitaba un lenguaje común: PDDL
- Ha tenido varias versiones, que permiten, entre otros:
  - especificación de costes de ejecución de operadores
  - tipos para variables de operadores
  - acciones durativas

Estructura de un dominio (I)

```
(define (domain <name>)
  (:requirements :<req1> :<req2>)
  (:types <subtype1> ...<subtypen> - <type1>
          <typen>)
  (:predicates <predicate1>
               <predicate2>
               <predicate3>
               ...
               <predicaten>)
(:action <action1>)
...
(:action <actionn>)
```

Estructura de un dominio (II)

- *Requirements*:
  - *:strips*: Permite añadir/borrar predicados en los efectos

    :effect (on ?x ?y)

    :effect (not (on ?x ?y))
  - *:typing*: Admite tipos/subtipos como en herencia, e.g., *truck airplane - vehicle*
  - *:equality*: Admite $=$

    (= (?s1 ?s2))
- Precondiciones:
  - and / or / not

Ejemplo de un dominio

```
(define (domain blocksworld)
  (:requirements :strips :equality)
  (:predicates (clear ?x)
               (on-table ?x)
               (arm-empty)
               (holding ?x)
               (on ?x ?y))
(:action pickup
  :parameters (?ob)
  :precondition (and (clear ?ob) (on-table ?ob) (arm-empty))
  :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob))
(not (arm-empty))))
...
```

Estructura de un problema

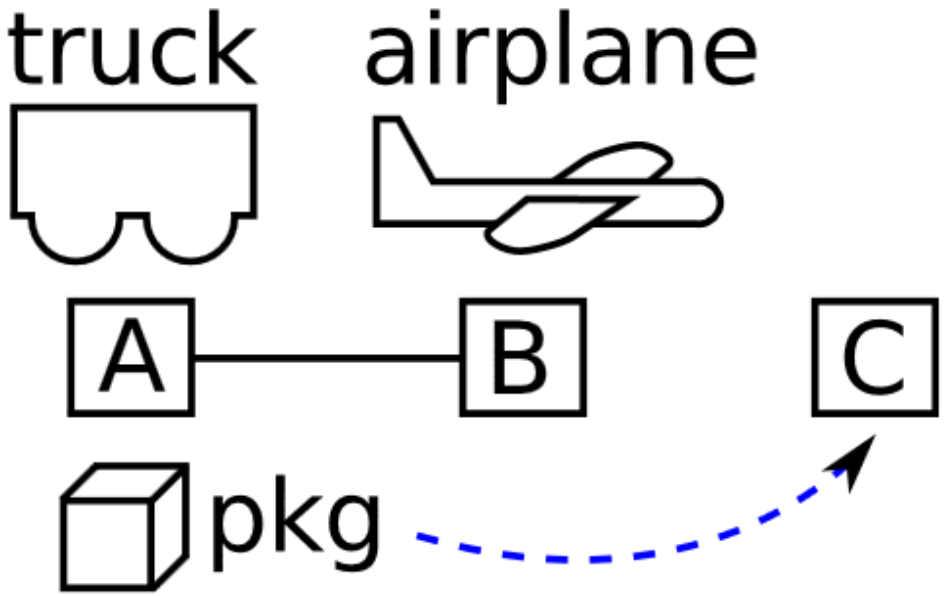
```
(define (problem <name>)
   (:domain <domain_name>)
   (:objects <obj1> <obj2> ...<objn>)
   (:init <pred_inst_1> <pred_inst_2> ...<pred_inst_n>)
   (:goal <formula_logica>)
)
```

Ejemplo de problema

```
(define (problem pb2)
  (:domain blocksworld)
  (:objects a b)
  (:init (on-table a) (on-table b) (clear a) (clear b) (arm-empty))
  (:goal (and (on a b))))
```

Ejemplo de un dominio (I)

Logistics



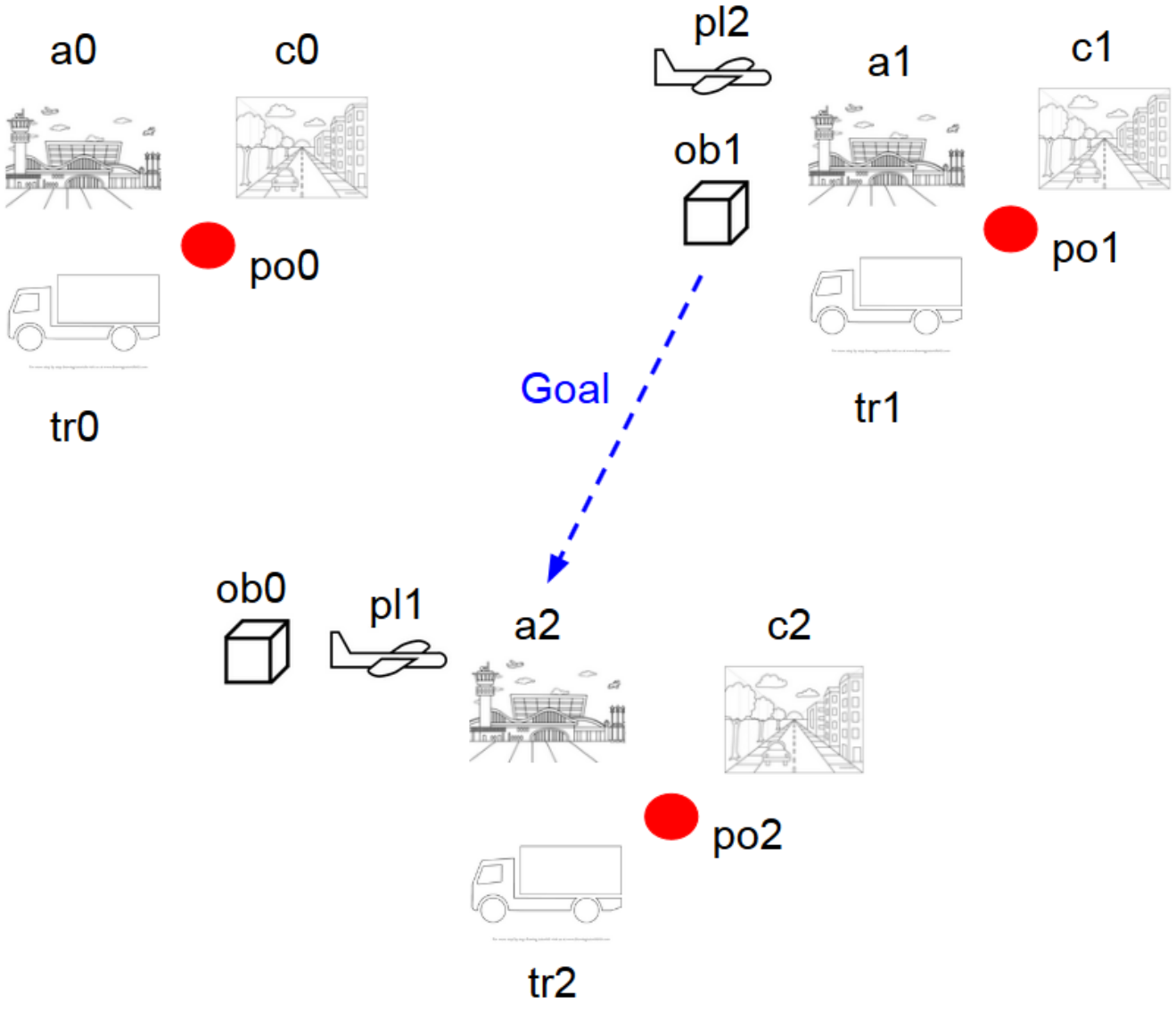
Ejemplo de un dominio (II)

```
(define (domain logistics)
  (:requirements :strips :typing :equality)
  (:types truck airplane - vehicle
          package vehicle - physobj
          airport location - place
          city place physobj - object)
  (:predicates (in-city ?loc - place ?city - city)
               (at ?obj - physobj ?loc - place)
               (in ?pkg - package ?veh - vehicle))
(:action fly-airplane
  :parameters (?p - airplane ?s - airport ?d - airport)
  :precondition (and (at ?p ?s)(not (= ?s ?d)))
  :effect (and (at ?p ?d)(not (at ?p ?s))))
...
```

Ejemplo de un dominio (III)

```
...
(:action drive-truck
:parameters (?truck - truck ?loc-from - place ?loc-to - place ?city -
city)
:precondition (and (at ?truck ?loc-from) (in-city ?loc-from ?city)
(in-city ?loc-to ?city)) :effect (and (not (at ?truck ?loc-from)) (at
?truck ?loc-to))
)
(:action load-vehicle
   :parameters (?pkg - package ?veh - vehicle ?loc - place)
   :precondition (and (at ?veh ?loc) (at ?pkg ?loc))
   :effect (and (not (at ?pkg ?loc)) (in ?pkg ?veh))
)
(:action unload-vehicle
   :parameters (?pkg - package ?veh - veh ?loc - place)
   :precondition (and (at ?veh ?loc) (in ?pkg ?veh))
   :effect (and (not (in ?pkg ?veh)) (at ?pkg ?loc))
)
```

Ejemplo de problema (I)



Ejemplo de problema (II)

```
(define (problem log1)
  (:domain logistics)
  (:objects (ob0 ob1 - package)
            (c2 c1 c0 - city)
            (po2 po1 po0 - location)
            (a2 a1 a0 - airport)
            (tr2 tr1 tr0 - truck)
            (pl1 pl2 - airplane))
  (:init (in-city a2 c2) (in-city po2 c2) (at tr2 po2)
         (in-city a1 c1) (in-city po1 c1) (at tr1 po1)
         (in-city a0 c0) (in-city po0 c0) (at tr0 po0)
         (at ob1 a1) (in ob0 pl1)
         (at pl2 a1) (at pl1 a2))
  (:goal (at ob1 a2)))
```

Operadores más ricos (I)

- Fluents:
  - Permite manejar valores numéricos en PDDL
  - Es necesario introducir *:fluents* en los requisitos
  - En el dominio es nesesario especificar los *fluents* que se van a utilizar con la etiqueta *functions*

    (:functions (fuel-level ?r - rover)
    (fuel-used ?r - rover)
    (fuel-required ?w1 ?w2 - waypoint)
    (total-fuel-used))
  - Se les da valor en el fichero del problema

    ...
    (:init ...((fuel-required W1 W2) = 3) ...)

Operadores más ricos (II)

- Fluents:
  - En los efectos de las acciones se puede incrementar/decrementar su valor
    (increase (fuel-used ?r) 8)
    (decrease (fuel-level ?r) (fuel-required ?w1 ?w2))
  - Ejemplo de acción:

(:action drive
:parameters (?t - truck ?from ?to - place)
:precondition (and (at ?t ?from) (< (fuel-needed ?to ?from)
(fuel-level ?t))
:effect (and (not (at ?t ?from)) (at ?t ?to)
(increase (total-cost) (drive-cost ?from ?to))
(decrease (fuel-level ?t) (fuel-needed ?to ?from))))

Operadores más ricos (III)

- La utilización de *fluents* permite definir *calidades* del plan diferentes del tiempo de ejecución, número de acciones, ...
- Se puede añadir un nuevo campo *:metric* a la definición del problema
- Ejemplo:

  (:metric minimize (fuel-used truck))

Métodos de planificación

Javier García

Departamento de Electrónica y Computación
Universidad de Santiago de Compostela

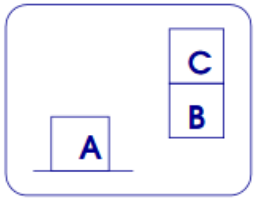
November 18, 2021

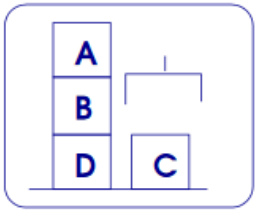
Búsqueda

- Sin conocimiento
  - Amplitud
  - Profundidad
  - Hacia delante/detrás
  - ...
- Con conocimiento
  - Escalada, Haz,...
  - A*, IDA*
  - ...

Ejemplo - Búsqueda

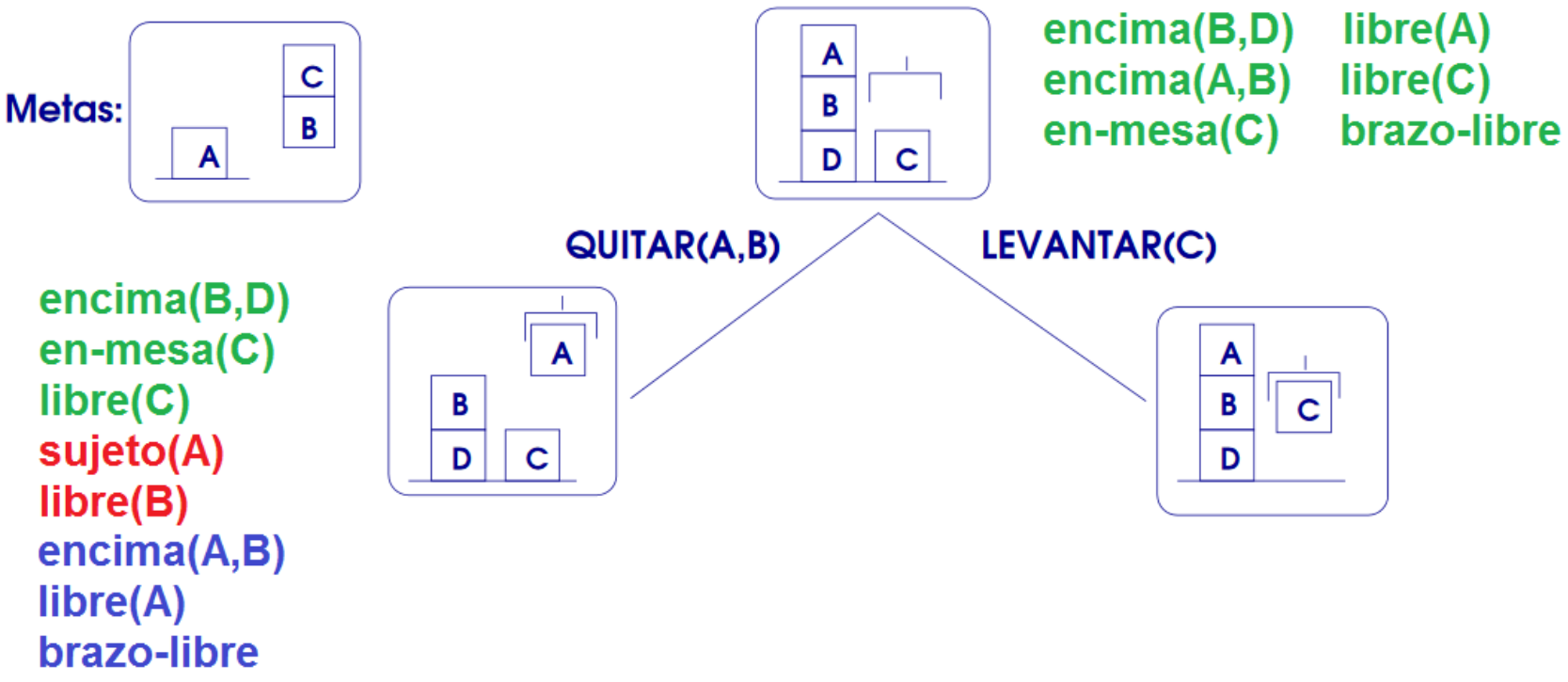
Metas:





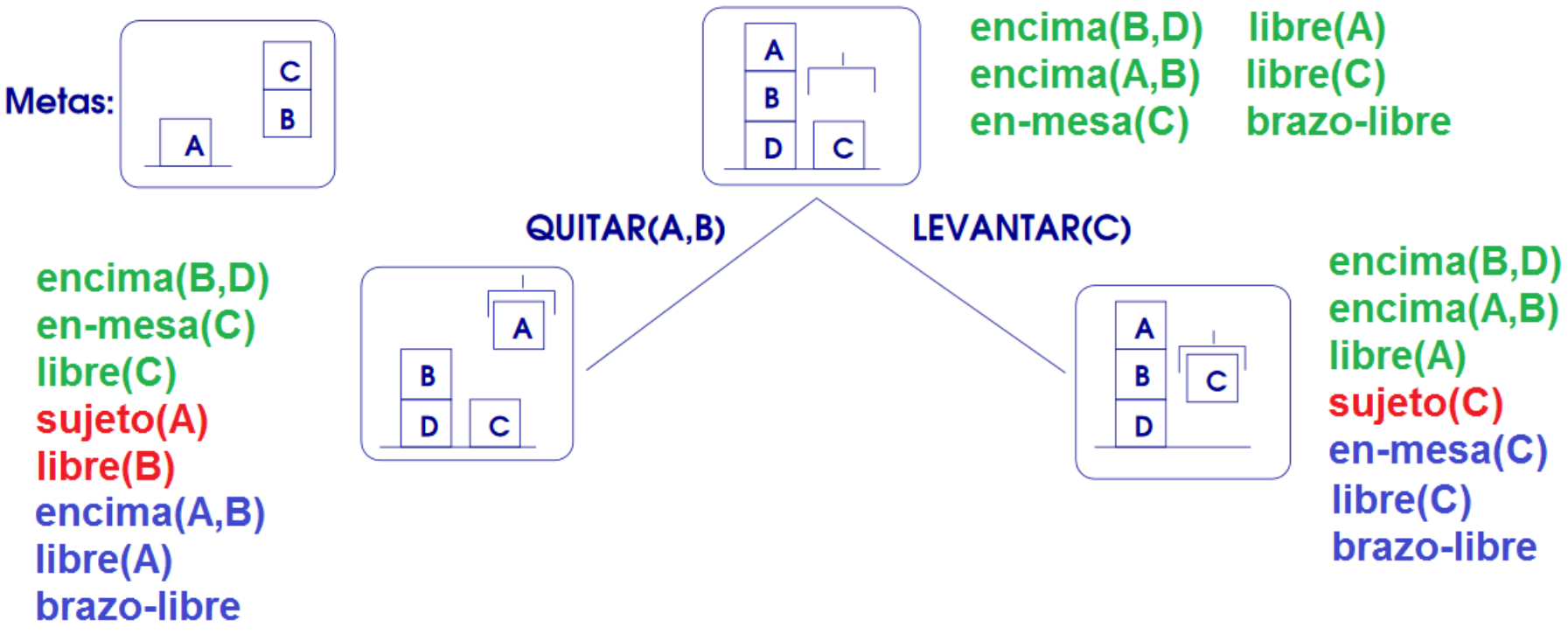
encima(B,D) libre(A)
encima(A,B) libre(C)
en-mesa(C) brazo-libre

Ejemplo - Búsqueda



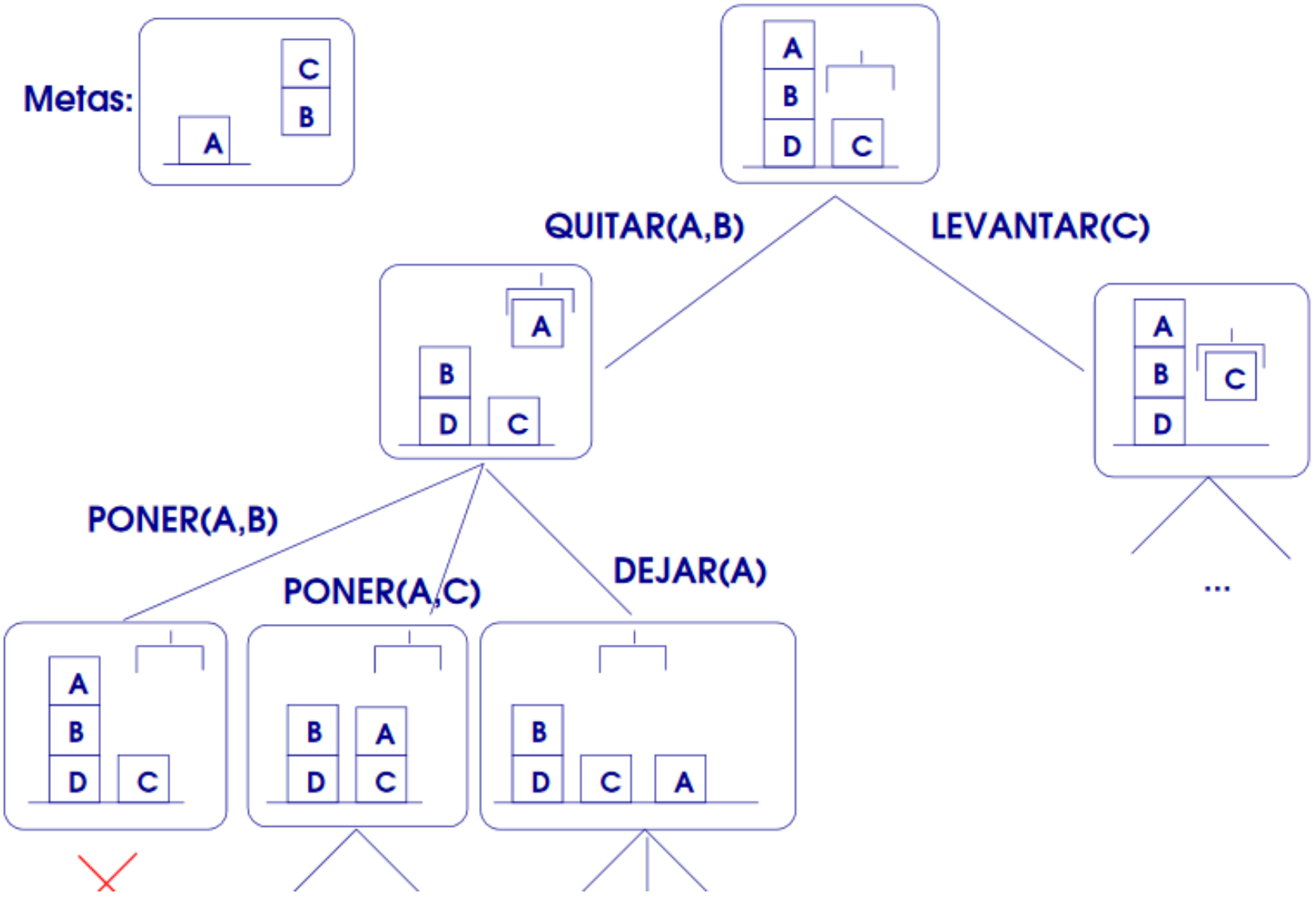
quitar(?x ?y): encima(?x ?y) libre(?x) brazo-libre
sujeto(?x) libre(?y)
¬encima(?x ?y) ¬brazo-libre ¬libre(?x)

Ejemplo - Búsqueda



levantar(?x): en-mesa(?x) libre(?x) brazo-libre
sujeto(?x)
¬en-mesa(?x) ¬libre(?x) ¬brazo-libre

Ejemplo - Búsqueda



Planificación clásica

Javier García

Departamento de Electrónica y Computación
Universidad de Santiago de Compostela

November 18, 2021

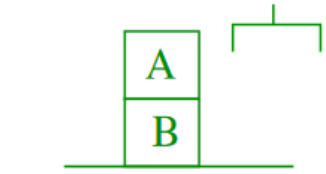
STRIPS

- Los algoritmos clásicos de búsqueda son ineficientes cuando se abordan problemas de planificación
- De ahí que aparezcan algoritmos específicos
- Uno de los primeros: STRIPS

STRIPS

- **Hipótesis**: Sólo cambian las cosas en el mundo que aparecen en las post-condiciones de los operadores
- Utiliza una pila durante el proceso de búsqueda
- **Idea**:
  - Meter en la pila las metas por conseguir y los operadores que consiguen dichas metas
  - Sacar de la pila las metas que sean ciertas en el estado actual y los operadores que se ejecuten

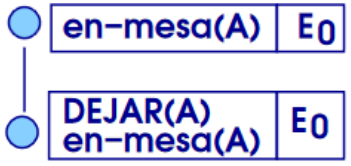
Ejemplo - STRIPS

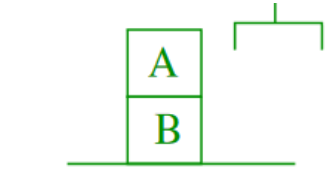


Estado inicial (E_0)

Metas

Ejemplo - STRIPS

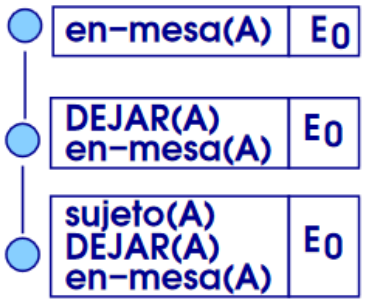


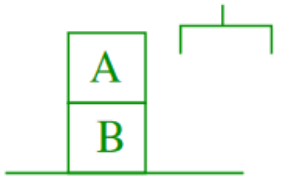


Estado inicial (E_0)

Metas

Ejemplo - STRIPS

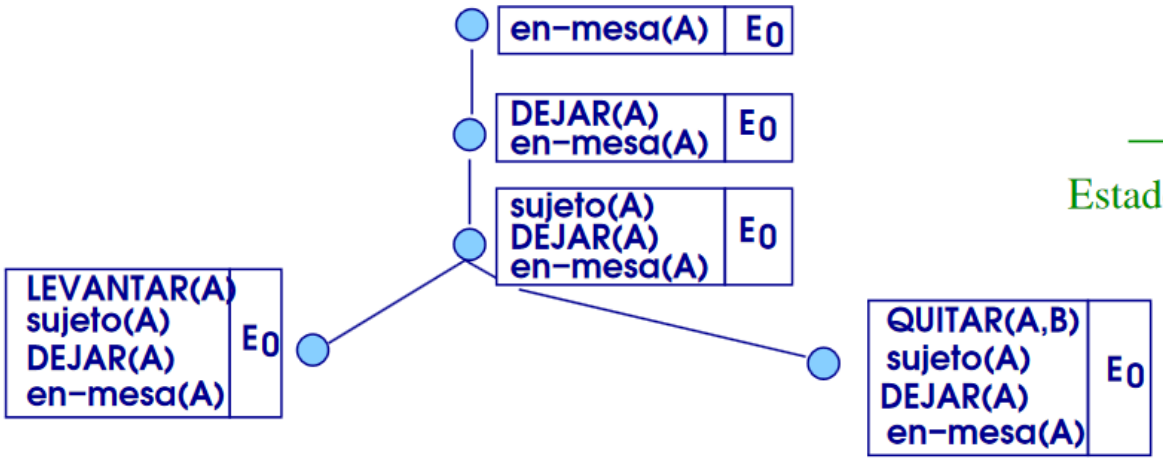


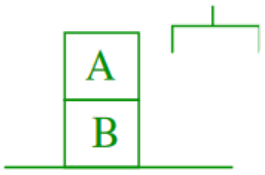


Estado inicial (E_0)

Metas

Ejemplo - STRIPS

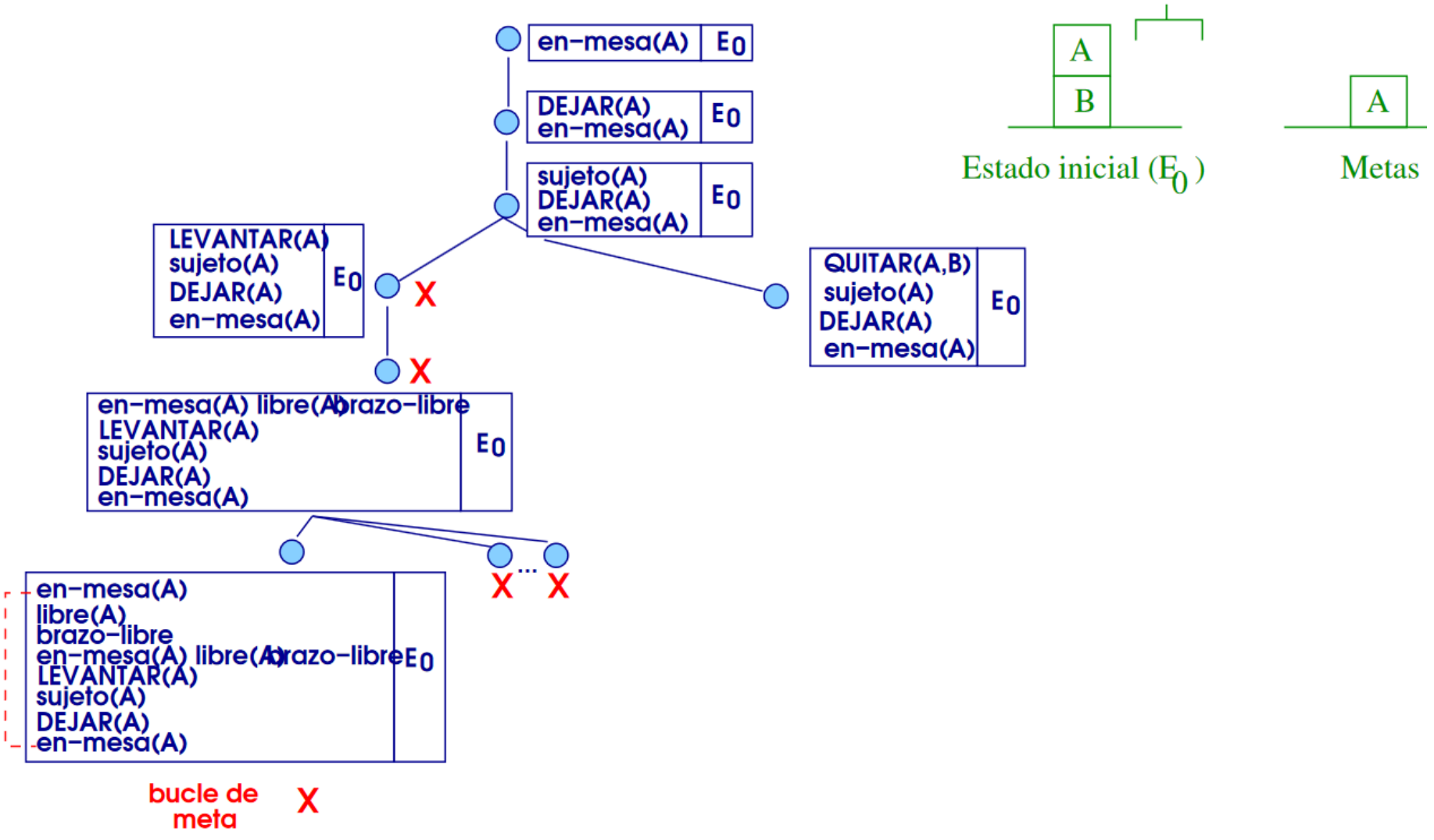




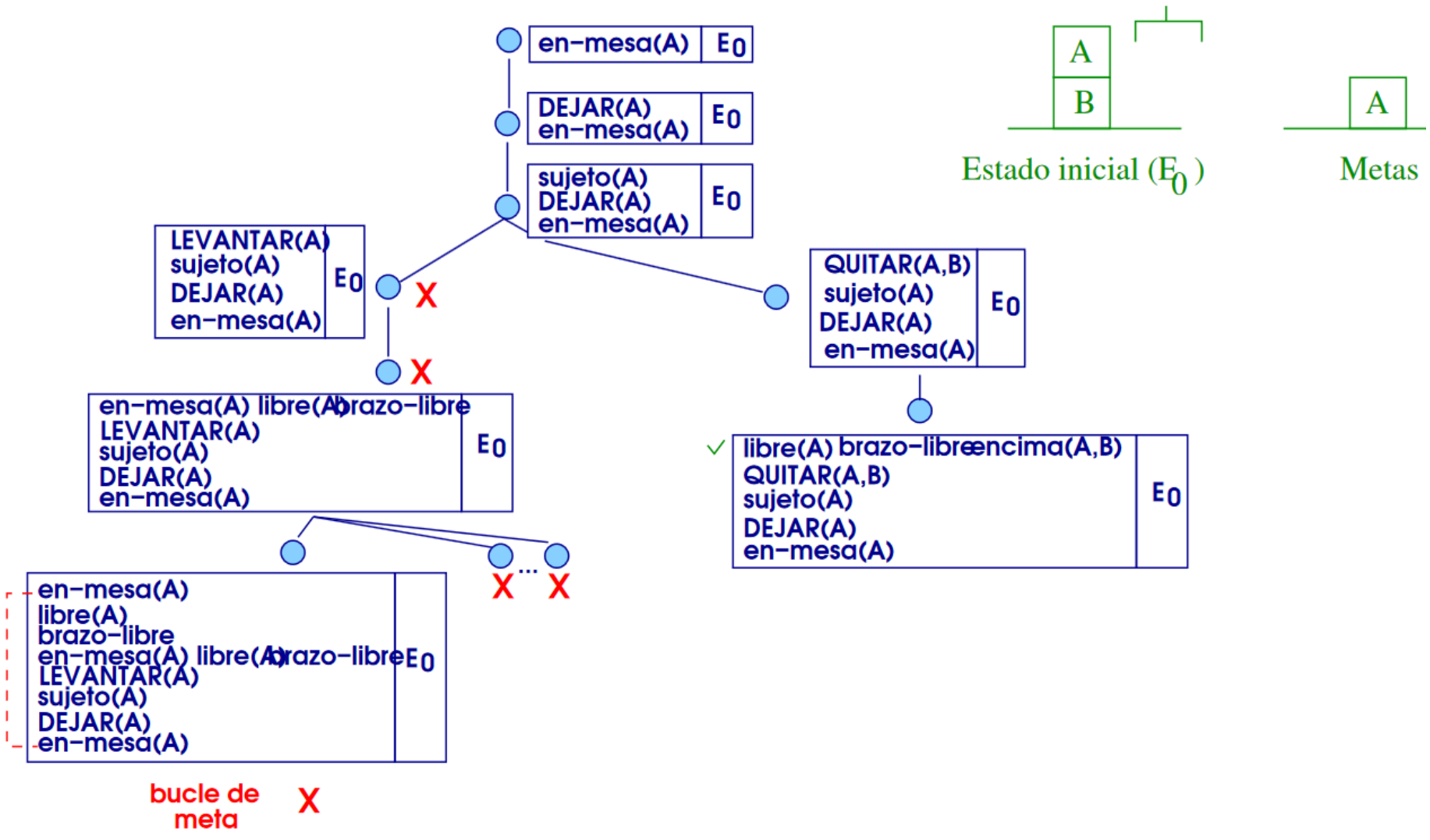
Estado inicial (E_0)

Metas

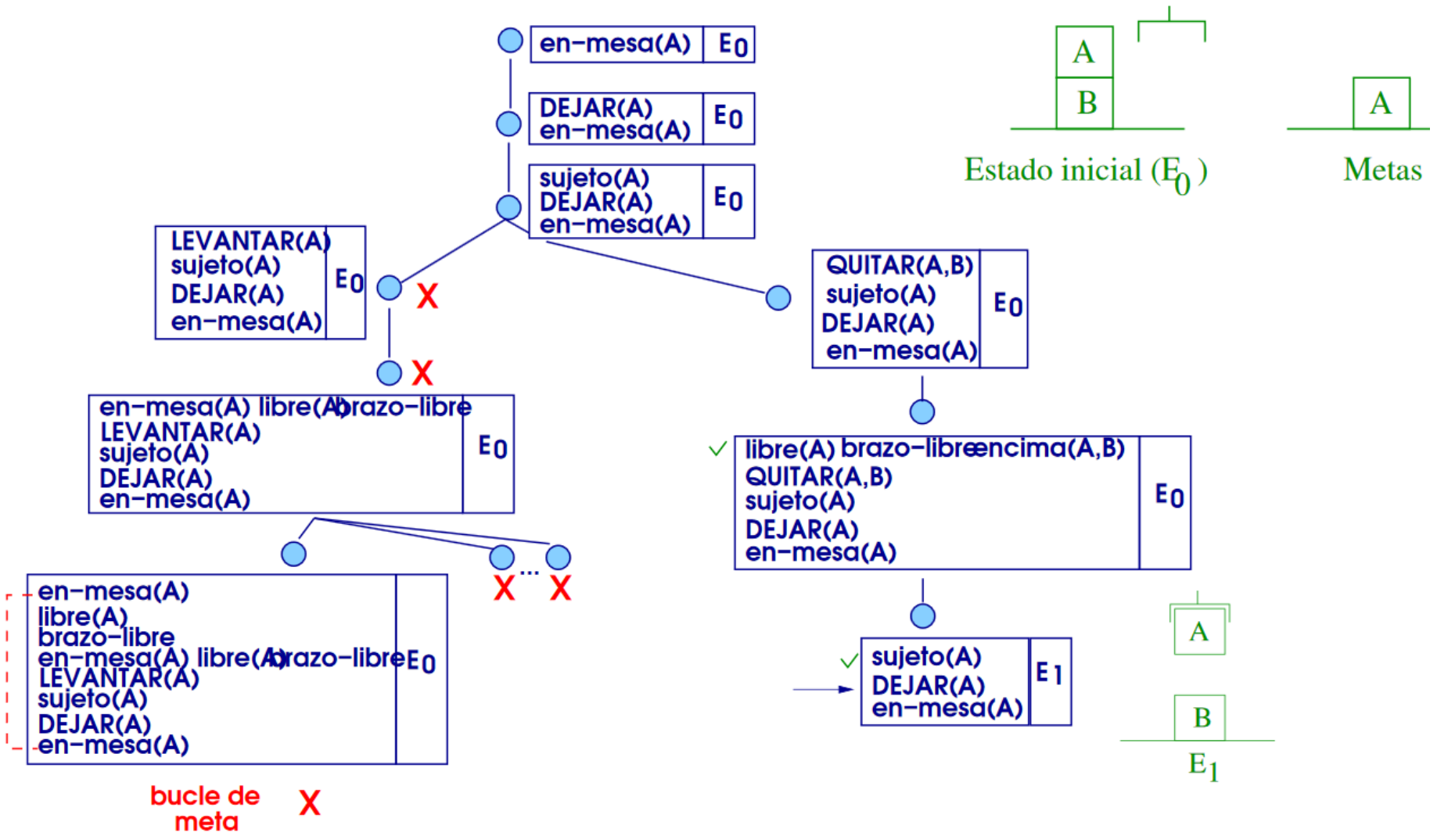
Ejemplo - STRIPS



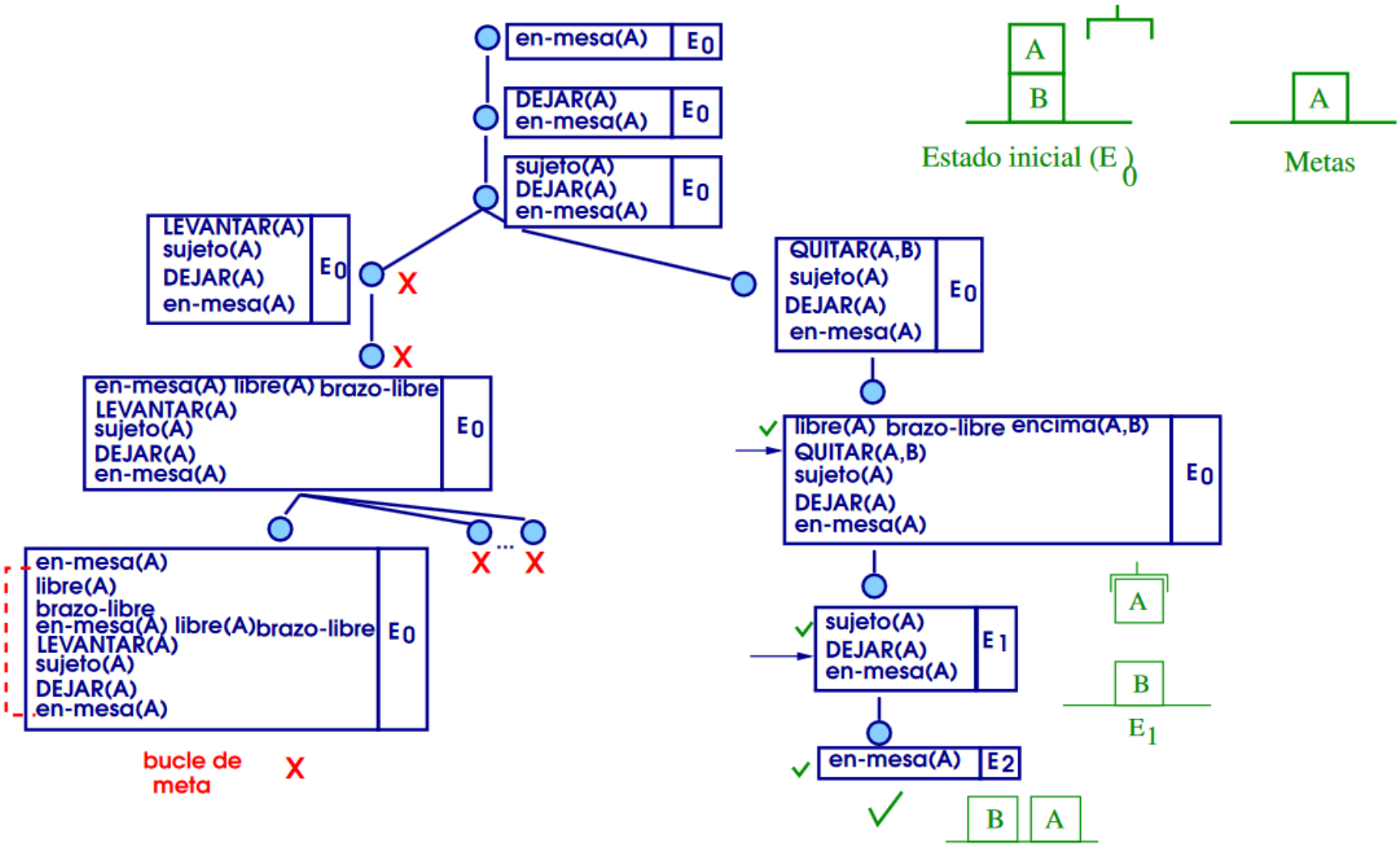
Ejemplo - STRIPS



Ejemplo - STRIPS

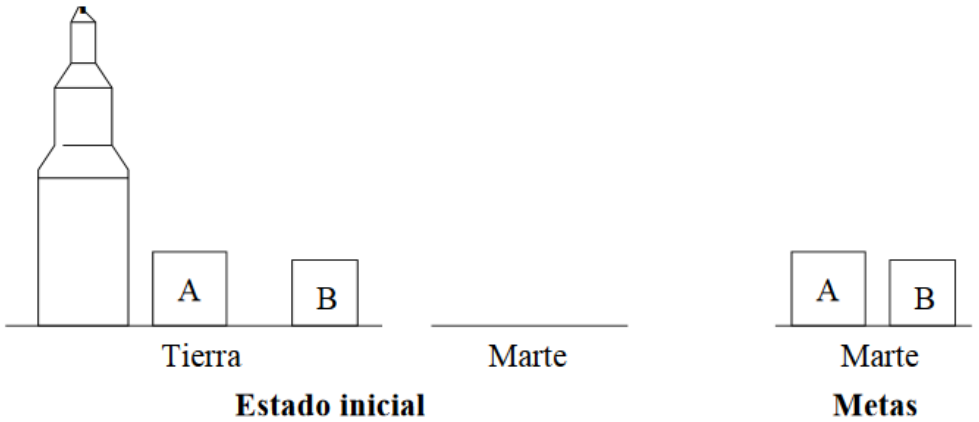


Ejemplo - STRIPS



Problema: linealidad

- STRIPS asume independencia entre las metas, por lo que las trata linealmente: hasta que no encuentra un plan para obtener una meta, no pasa a las siguientes metas
- No funciona cuando hay recursos limitados, por lo que es:
  - incompleta: existe solución, pero no la encuentra



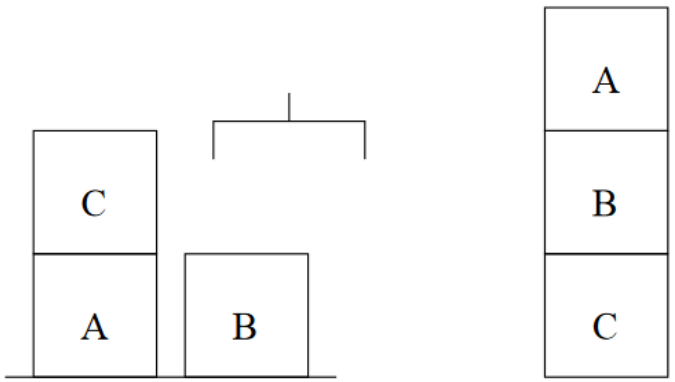
Goal	*Plan*

`(at obj1 locB)`	`(LOAD-ROCKET obj1 locA)`
	`(MOVE-ROCKET)`
	`(UNLOAD-ROCKET obj1 locB)`

`(at obj2 locB)`	*failure*

Problema: linealidad

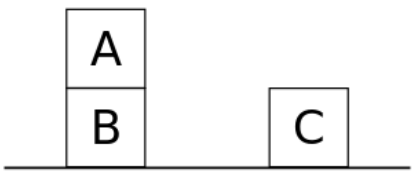
- No funciona cuando hay recursos limitados, por lo que es:
  - no óptima: no encuentra la solución óptima



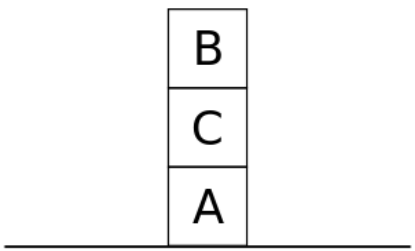
Estado inicial **Metas**

encima(A,B) encima(B,C)

1. encima(A,B)



2. encima(B,C)



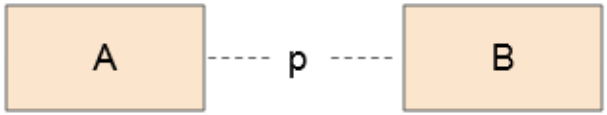
Planes parcialmente ordenados

- Planificadores no lineales que resuelven los problemas de los lineales
- Ordenamiento parcial entre acciones: solo se ordenan las acciones cuando realmente es necesario
- Ejemplo:
  1. Ir a la tienda
  2. Comprar huevos; Comprar harina; Comprar leche
  3. Pagar
  4. Ir a la cocina
- **Plan parcialmente ordenado**: un plan en el que solo se especifican algunas de las precedencias entre sus operadores

Planes parcialmente ordenados: componentes

- Nodos: Acciones, que constituyen los pasos que el plan lleva a cabo, de entre las acciones del problema, cada una con sus precondiciones y efectos
  - Al inicio, dos acciones especiales: INICIO (sin precondiciones y cuyo efecto es el estado inicial) y FIN (sin efectos y cuyas precondiciones son el objetivo final)
- Arcos: restricciones $A < B$ entre las acciones del plan

- Enlaces causales: Tipo especial de arco $A - -p - -B$, que indica que la ejecución de la acción A tiene como efecto p, que a su vez es precondición de B



- Precondiciones abiertas: aquellas que aún no tienen enlaces causales que las consigan

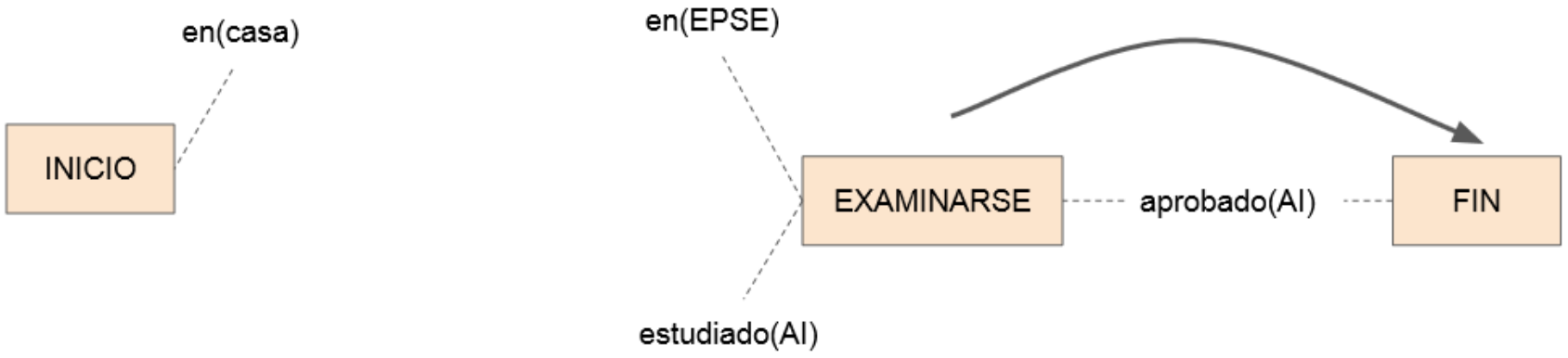
Planes parcialmente ordenados: Ejemplo

- Objetos: `AI`, `EPSE`, `casa`
- Predicados: `estudiado(?x)`, `en(?x)`, `aprobado(?x)`
- Estado inicial:
  `en(casa)`
- Estado final:
  `aprobado(AI)`
- Acciones:

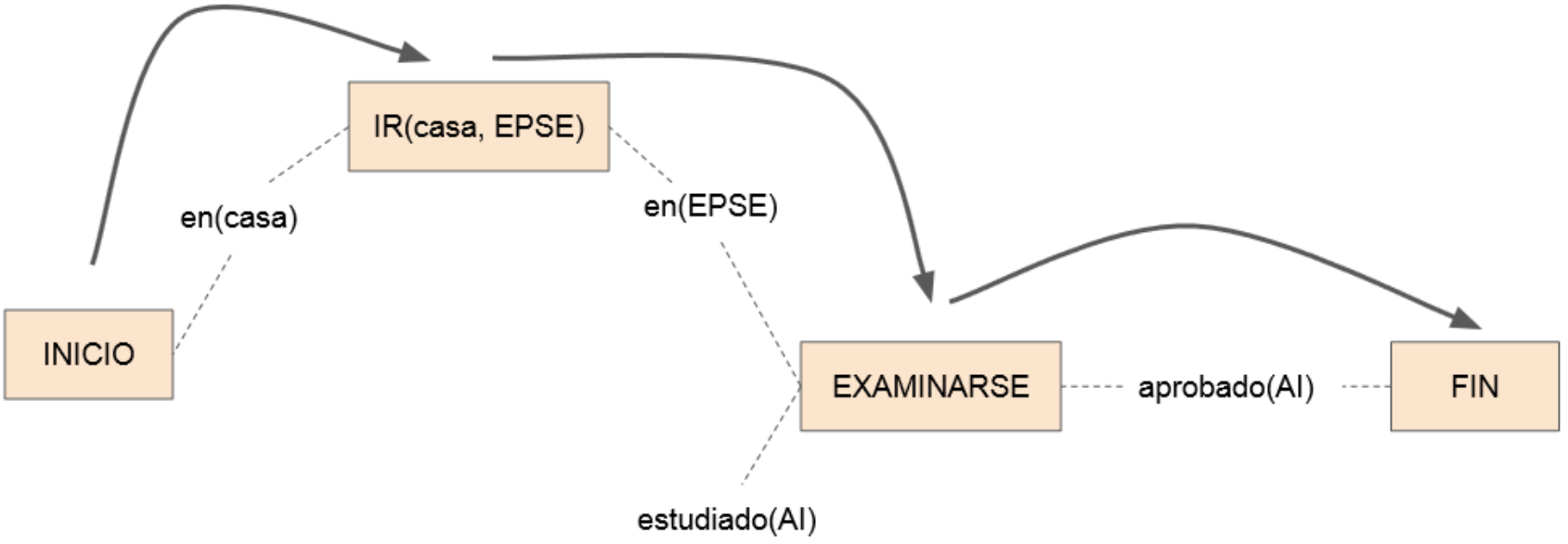
	`IR(?x,?y)`	`ESTUDIAR(?x)`	`EXAMINARSE(?x)`
P:	`en(?x)`		`en(EPSE),estudiado(?x)`
E:	¬`en(?x),en(?y)`	`estudiado(?x)`	`aprobado(?x)`

Planes parcialmente ordenados: Ejemplo

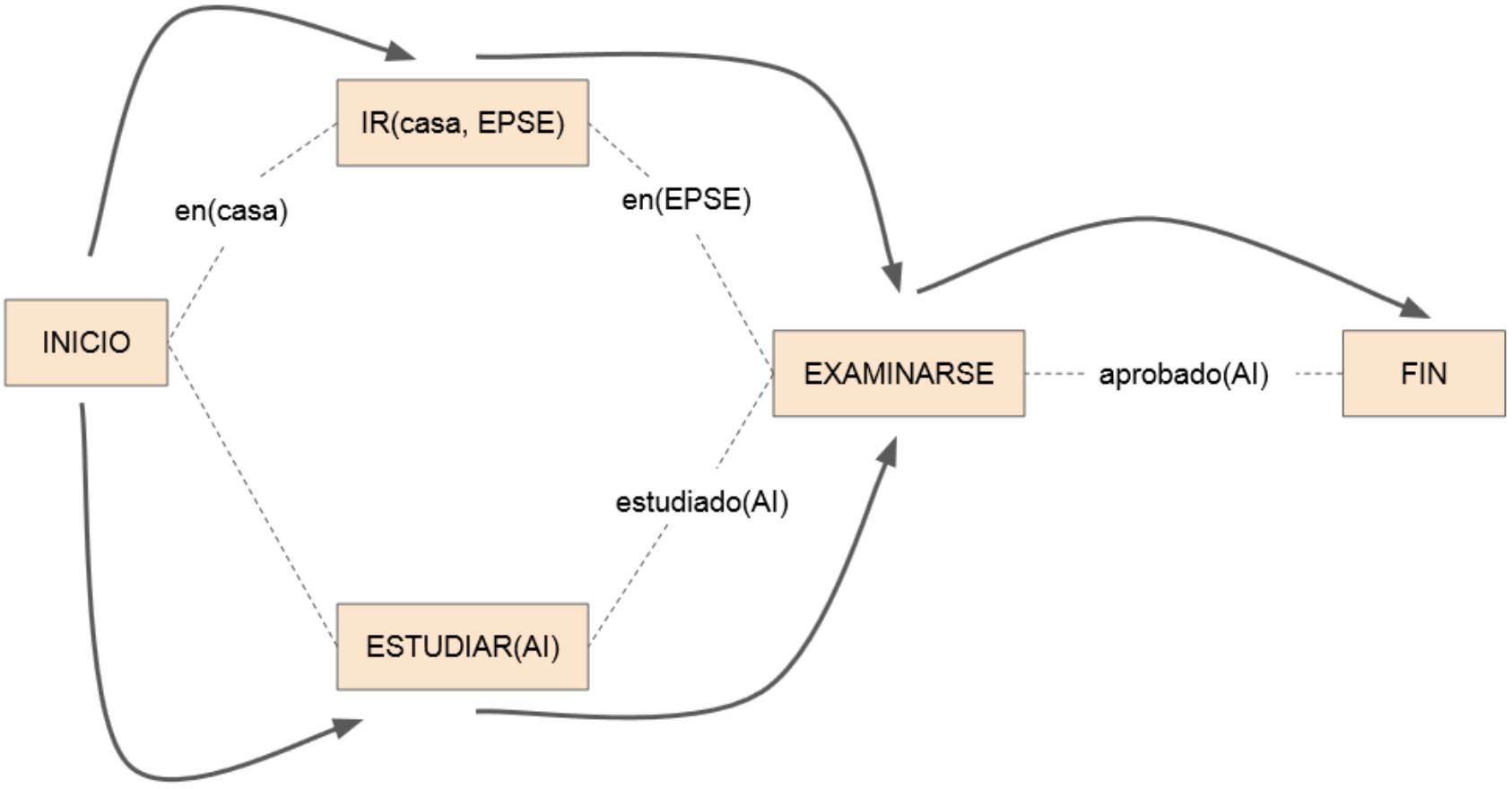
Planes parcialmente ordenados: Ejemplo



Planes parcialmente ordenados: Ejemplo

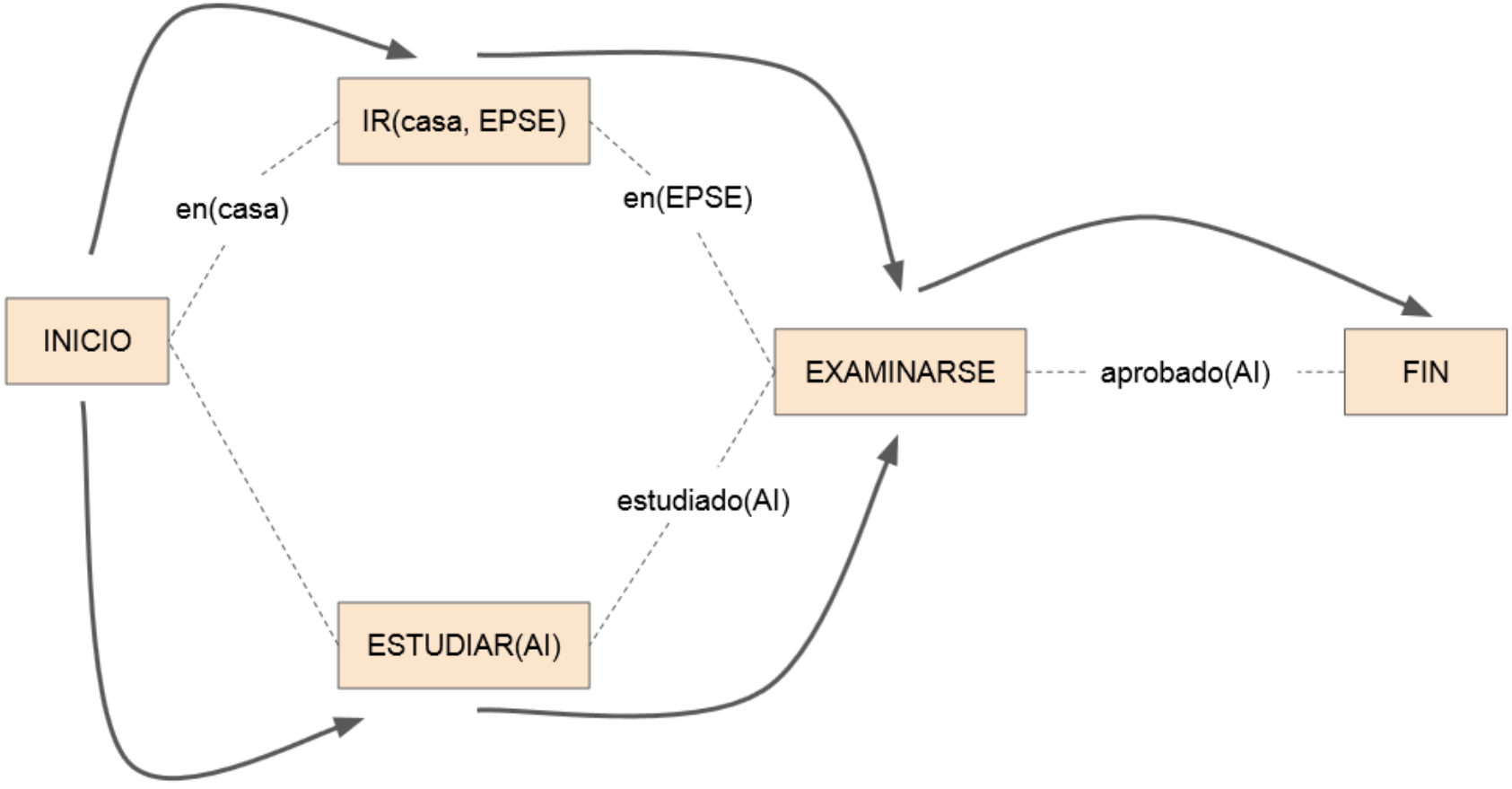


Planes parcialmente ordenados: Ejemplo



IR(casa, EPSE) < EXAMINARSE
ESTUDIAR(AI) < EXAMINARSE

Planes parcialmente ordenados: Ejemplo



IR(casa, EPSE) < EXAMINARSE
ESTUDIAR(AI) < EXAMINARSE

PLAN1: IR(casa,EPSE), ESTUDIAR, EXAMINARSE
PLAN2: ESTUDIAR, IR(casa, EPSE), EXAMINARSE

Planes parcialmente ordenados: Ejemplo

- El proceso de planificación finaliza cuando no quedan precondiciones abiertas
- Durante el proceso pueden aparecer **conflictos**
  - Una acción C amenaza o entra en conflicto con un enlace causal $A - -p - - B$, si C tiene a $\neg p$ en su lista de efectos, y C podría ir después que A y antes que B
  - Puede solucionarse colocando C antes que A, o después que B
  - Si aparecen ciclos buscar otra alternativa
- Una de las grandes ventajas es que permite encontrar acciones que se pueden ejecutar en paralelo:
  - Podemos estudiar mientras vamos de camino a la EPSE a examinarnos

Otros métodos de planificación

Javier García

Departamento de Electrónica y Computación
Universidad de Santiago de Compostela

November 18, 2021

Otros métodos de planificación

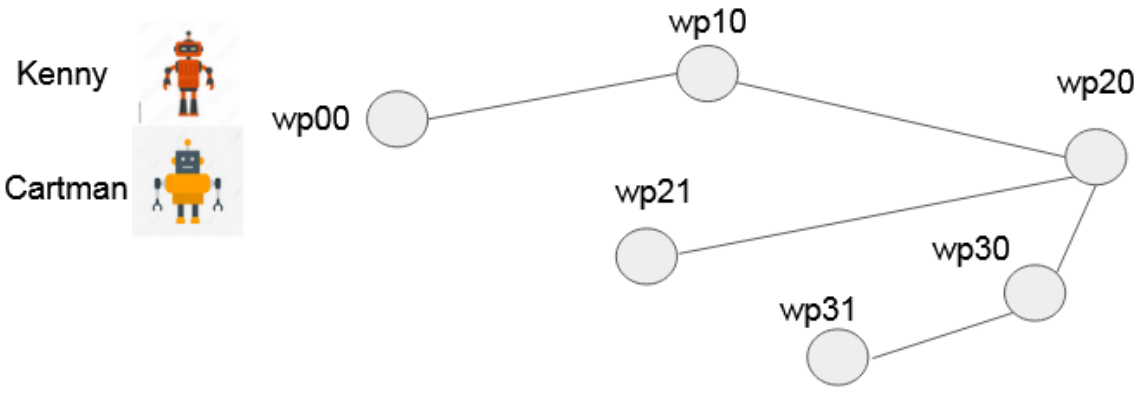
- Grafo de planes: **GRAPHPLAN** (Blum and Furst, 1995)
  - Dos fases:
    - **Generación de alcanzabilidad**: niveles alternados de proposición y acción
    - **Búsqueda de plan válido**: búsqueda hacia atrás de un posible plan paralelo
- Planificación SAT: **SATPlan** (Kautz and Selman, 1996)
  - Codificación de los estados y las operaciones en lógica proposicional
  - Buscar un modelo (una asignación de valores verdadero/falso a las variables) de forma que toda la fórmula se reduzca a verdadero (e.g., Davis-Putnam)
- Planificación heurística: **Metric-FF** (Hoffmann, 2001)
  - Heurísticas dependientes e independientes del dominio
  - Metric-FF: Calcula una heurística independiente del dominio a través del grafo de planificación
  - Utiliza estas heurísticas para guiar el proceso de búsqueda

Otros métodos de planificación

- Planificación temporal: **OPTIC** (Benton et al, 2012), **POPF** (Coles et al, 2010)
  - Capacidad de razonar con el tiempo
  - Acciones durativas
  - Modificadores: *at start*, *over all*, *at end*

```
(:durative-action navigate
:parameters (?x - rover ?y - waypoint ?z - waypoint)
:duration (= ?duration 5)
:condition (and (over all (can_traverse ?x ?y ?z)) (at start (available ?x)) (at start
    (at ?x ?y))  (at start (>= (energy ?x) 8))
              (over all (visible ?y ?z))
       )
:effect (and (at start (decrease (energy ?x) 8)) (at start (not (at ?x ?y))) (at end (at
    ?x ?z))))
```

Otros métodos de planificación



Goal:

(robot_at kenny wp31)
(robot_at cartman wp21)

move ?robot robot ?from ?to - waypoint DURATION 10

POPF:

0.000: (move cartman wp00 wp10) [10.000]
0.000: (move kenny wp00 wp10) [10.000]
10.001: (move cartman wp10 wp20) [10.000]
10.001: (move kenny wp10 wp20) [10.000]
20.002: (move cartman wp20 wp21) [10.000]
20.002: (move kenny wp20 wp30) [10.000]
30.003: (move kenny wp30 wp31) [10.000]