Planificación Automática

Grupo PLG

Universidad Carlos III de Madrid

IA. 2008-09

Indice

Indice

Indice

Indice

Indice

Indice

Indice

En los comienzos...

- Las primeras aproximaciones a la planificación se hicieron desde el cálculo de situaciones (*situation calculus*)
- Usado en sistemas como QA3 [Green, 1969] o por McCarthy-Hayes [McCarthy and Hayes, 1969]
- Todavía activo en comunidades de Representación del conocimiento

Representación de los estados

- utilizaban un predicado `holds(`*fluent*`,estado)`
- *Fluent* (función): literal que se decía era cierto en el estado
- ejemplo: `holds(encima(A,B),S0)`
- utilizaban `result(op,estado)` que devolvía un nuevo estado
- ejemplo:
  `holds(encima(A,B),result(poner(A,B),S0))`

Representación de los operadores

```
holds(sujeto(x),result(quitar(x,y),s)) :-
   holds(encima(x,y),s),holds(libre(x),s),holds(brazo-libre,s)
holds(libre(y),result(quitar(x,y),s)) :-
   holds(encima(x,y),s),holds(libre(x),s),holds(brazo-libre,s)
~holds(encima(x,y),result(quitar(x,y),s)) :-
   holds(encima(x,y),s),holds(libre(x),s),holds(brazo-libre,s)
~holds(libre(x),result(quitar(x,y),s)) :-
   holds(encima(x,y),s),holds(libre(x),s),holds(brazo-libre,s)
~holds(brazo-libre(x),result(quitar(x,y),s)) :-
   holds(encima(x,y),s),holds(libre(x),s),holds(brazo-libre,s)
holds(en-mesa(z),result(quitar(x,y),s)) :-
   holds(en-mesa(z),s)
```

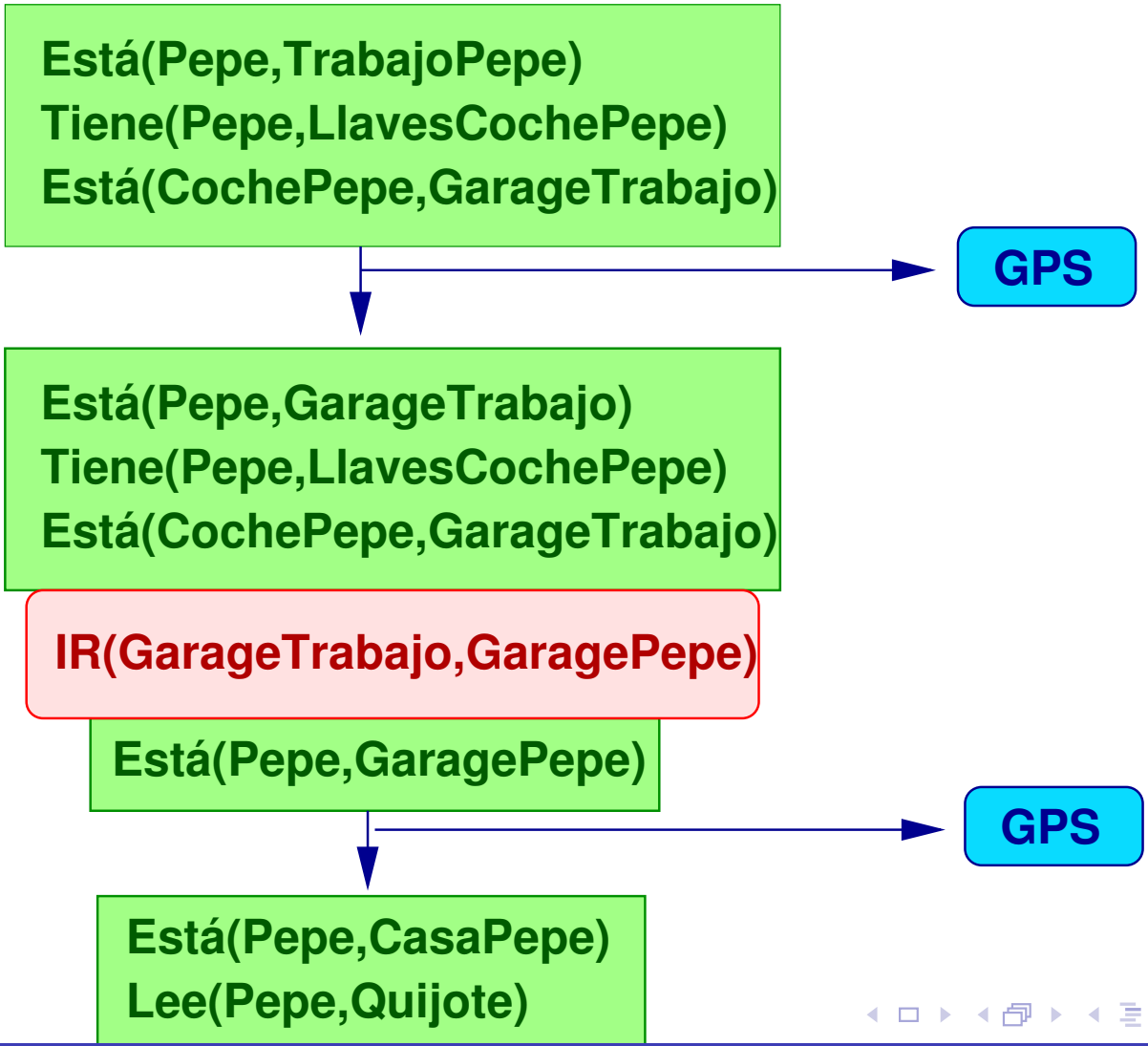
Razonamiento

- Utilizan PROLOG directamente
- Problema del marco (*frame problem*): ¿qué ocurre cuando ejecutamos un operador?
- Solución: axiomas de marco (*frame axioms*) hay que especificar los literales que cambiaban y no cambiaban por la ejecución de los operadores

Análisis de Medios-Fines. GPS

- Idea Intuitiva: Para resolver un problema,
  - analizar qué fines se desean conseguir (`metas`),
  - de qué medios se dispone para conseguirlos (`operadores`),
  - utilizar los medios para reducir, progresivamente, las `diferencias` entre el estado inicial y las metas.
- GPS ("General Problem Solver") de *Newell, Simon* y sus estudiantes fue el primer sistema en proponer esta idea [Ernst and Newell, 1969]
- El objetivo del proyecto era construir un solucionador general que *simulara* la forma de razonamiento humana
- Tenía una fuerte componente psicológica

Ejemplo de reducción de diferencias

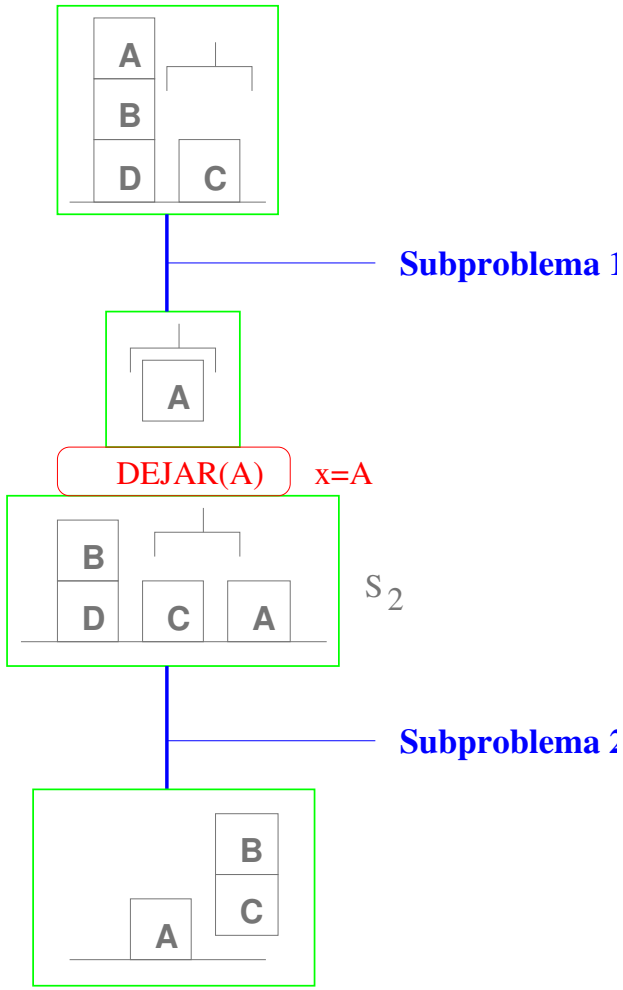


Función GPS

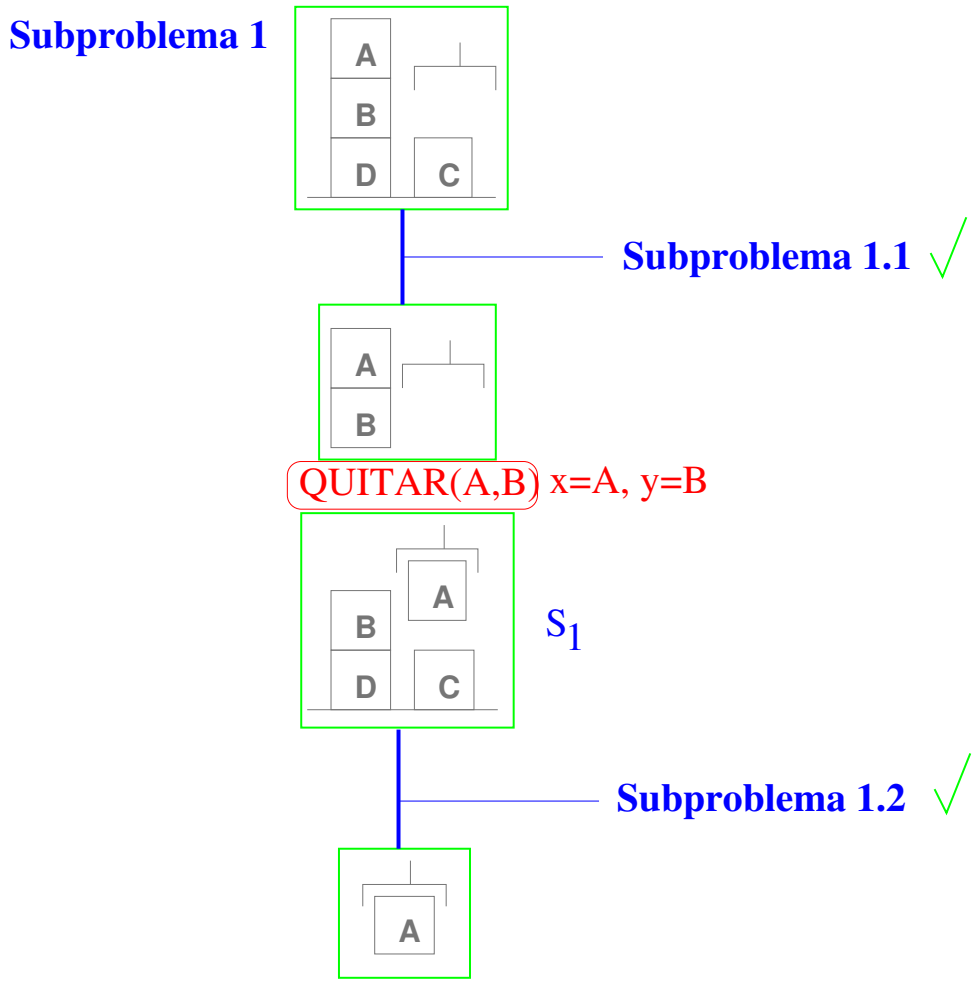
Entradas: Estado inicial (E), Metas (M) y Operadores (O)
Salidas: [Plan o Fallo, Estado Alcanzado o Fallo]

Si $M \subseteq E$ Entonces devuelve [*Verdadero*,*E*]
Si no, *diferencia*:=`elige-diferencia`(E, M);
R:=`operadores-relevantes`(*diferencia*,*O*);
solución:=Falso;
Mientras ($R \neq \phi$) `Y` (*solución=Falso*)
o:=`elige-operador`(*diferencia*,*R*);
$R := R - \{o\}$;
$[P, E']$:=`GPS`(E, `precondiciones`(*o*),*O*);
Case *P* of
Verdadero: $[P_1, E'_1]$:=`GPS`(`ejecuta`(*o*,E'),*M*, *O*);
Si $P_1 \neq$*Falso* Entonces *solución*:=$[o+P_1, E'_1]$
Falso: –
Si no: (P_1, E'_1):=`GPS`(`ejecuta`(o, E'),*M*, *O*);
Si $P_1 \neq$*Falso* Entonces *solución*:=$[P+o+P_1, E'_1]$;
Si *solución=Falso* Entonces devuelve [*Falso,Falso*]
Si no, devuelve *solución*

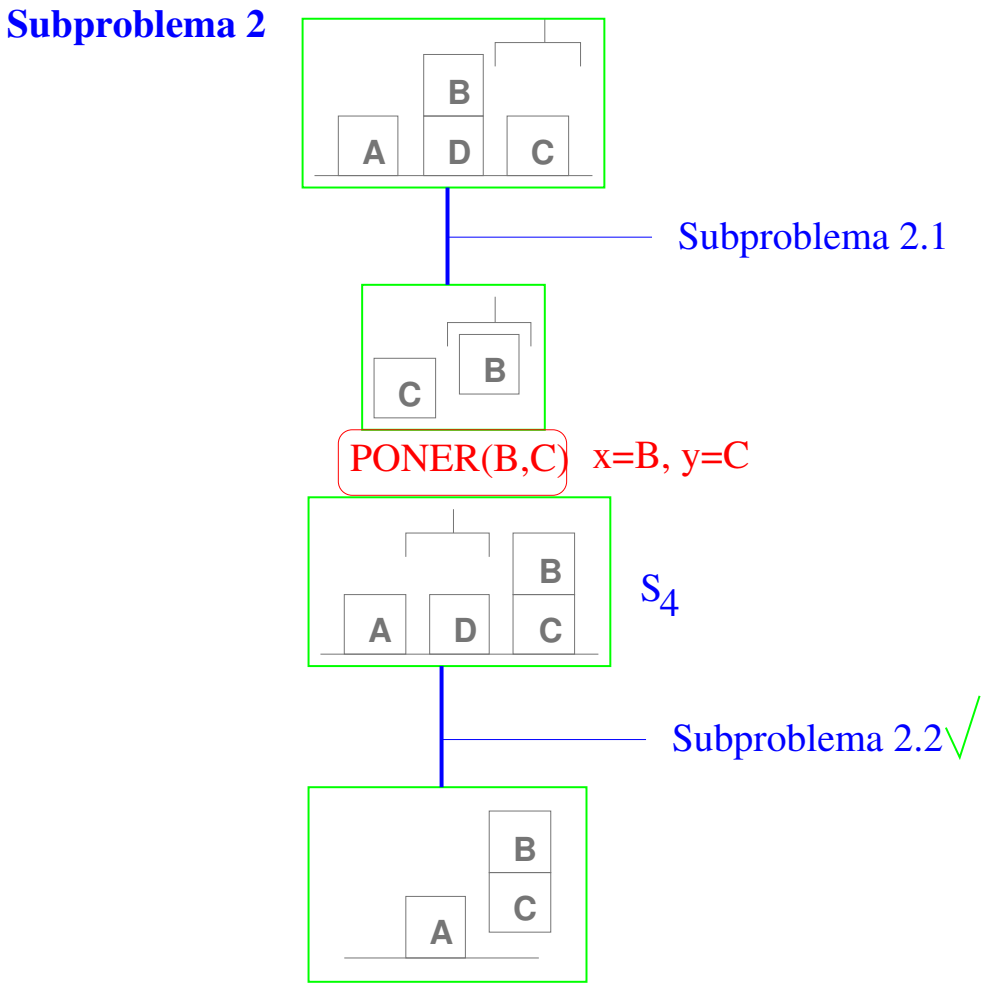
Ejemplo de funcionamiento



Ejemplo de funcionamiento (cont.)



Ejemplo de funcionamiento (cont.)



Ejemplo de funcionamiento (cont.)

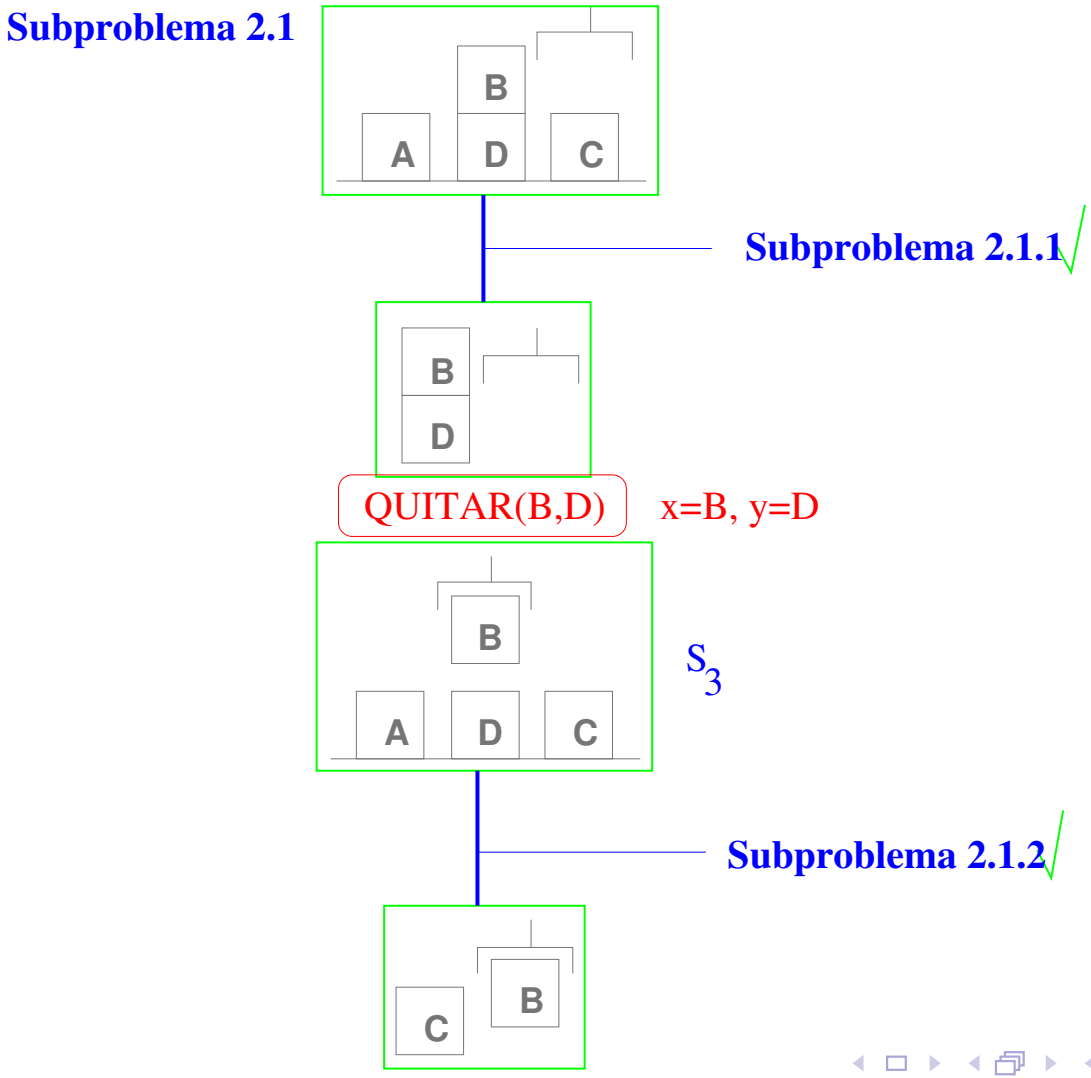


Tabla de diferencias

Operadores	**Diferencias**				
	en-mesa(x)	encima(x,y)	libre(x)	sujeto(x)	brazo-libre
DEJAR(x)	•		•		•
LEVANTAR(x)				•	
QUITAR(x,y)			•	•	
PONER(x,y)					•

Aprendizaje de la tabla de diferencias

- Se inicializan todas las posiciones a (0,0)
- Cada vez que se intenta reducir una diferencia d con un operador o y funciona:

si `tabla[`o`,`d`]=(A,I)` $\longrightarrow$ `tabla[`o`,`d`]:=(A+1,I+1)`

y, además, para cada $o' \neq o$,

si `tabla[`o'`,`d`]=(A',I)` $\longrightarrow$ `tabla[`o'`,`d`]:=(A',I+1)`

Control en GPS

- Tipos de decisión:
  - ¿Qué diferencia reducir?
  - ¿Qué operador utilizar para reducir la diferencia?
  - [¿Qué instanciación de operador utilizar?]
- Se hace retroceso cuando:
  1. No hay operadores que puedan reducir la diferencia
  2. Ningún operador ha podido reducir la diferencia
  3. Reducir una diferencia genera subproblemas más difíciles
  4. Aparece un bucle de meta
  5. Se llega a una profundidad máxima
- Tipos de conocimiento de control
  - Independiente del dominio (1, 2 y 4)
  - Dependiente del dominio (tabla de diferencias y 5)

Más información en

Referencias: [Allen *et al.*, 1990,
Borrajo *et al.*, 1993,
Newell *et al.*, 1972,
Rich and Knight, 1994]

STRIPS

Objetivo: construcción de un sistema de control para el robot Shakey

Representación de operadores

- Problema del marco: ¿qué ocurre con el contexto del mundo cuando se ejecuta una acción?
- Solución (hipótesis STRIPS): sólo cambian las cosas que aparecen en las post-condiciones de cada operador [Fikes and Nilsson, 1971]
- Búsqueda:
  - Nodos: estado actual y pila de metas-operadores
  - Nodo raíz: estado inicial y conjunción de metas
  - Heurística: seleccionar siempre alguno de los sucesores de cada nodo
- 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

Algoritmo de STRIPS

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

- Si la cima de la pila del nodo es una conjunción de metas
  - Si la conjunción es cierta en el estado Entonces se elimina de la pila
  - Si no, generar como sucesores todas las posibles combinaciones de las metas
    seleccionar una de ellas
- Si la cima de la pila del nodo es una meta
  - Si la meta es cierta en el estado Entonces se elimina de la pila
  - Si no, Si hay bucle de meta Entonces retroceder
    - Si no, generar un sucesor por cada instanciación de operador
      que añade dicha meta
      - Si hay sucesores Entonces elegir uno
      - Si no, retroceder
- Si la cima de la pila del nodo es un operador instanciado
  - Si el operador instanciado se puede ejecutar
  - Entonces ejecutar operador, quitarlo de la pila y añadirlo al plan
  - Si no, se introducen sus precondiciones en la pila

Representación de operadores

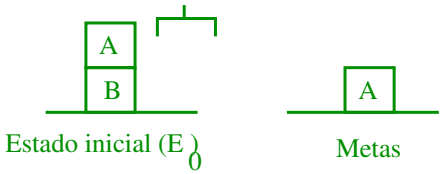
QUITAR(x, y)
precondiciones: encima(x, y),libre(x),brazo-libre
añadidos: sujeto(x),libre(y)
borrados: encima(x, y),brazo-libre,libre(x)

LEVANTAR(x)
precondiciones: en-mesa(x),libre(x),brazo-libre
añadidos: sujeto(x)
borrados: en-mesa(x),brazo-libre,libre(x)

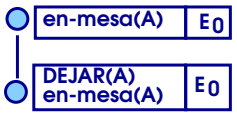
PONER(x, y)
precondiciones: sujeto(x),libre(y)
añadidos: encima(x, y),libre(x),brazo-libre
borrados: sujeto(x),libre(y)

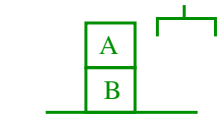
DEJAR(x)
precondiciones: sujeto(x)
añadidos: en-mesa(x),libre(x),brazo-libre
borrados: sujeto(x)

Ejemplo de STRIPS



Ejemplo de STRIPS

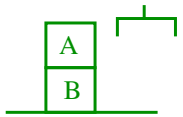




Estado inicial (E_0)

Metas

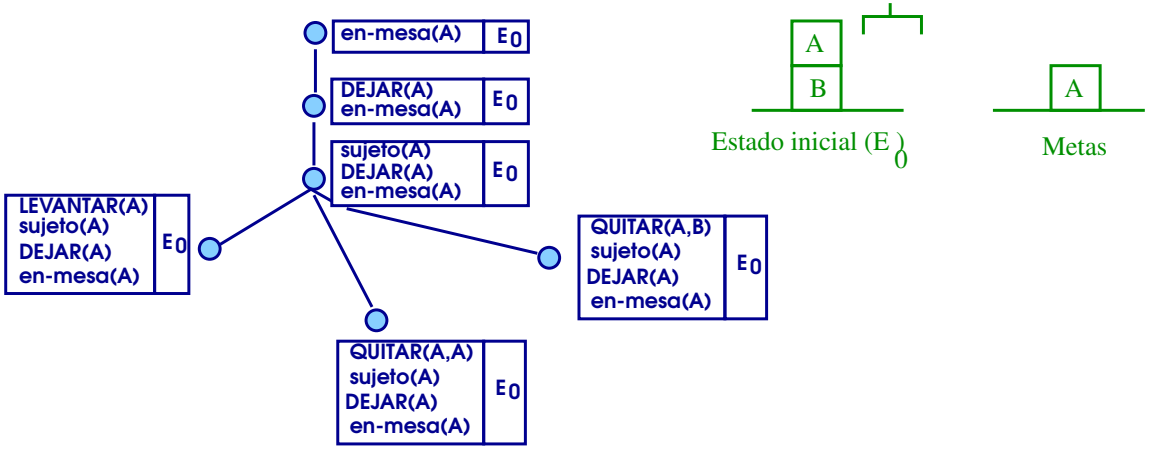
Ejemplo de STRIPS



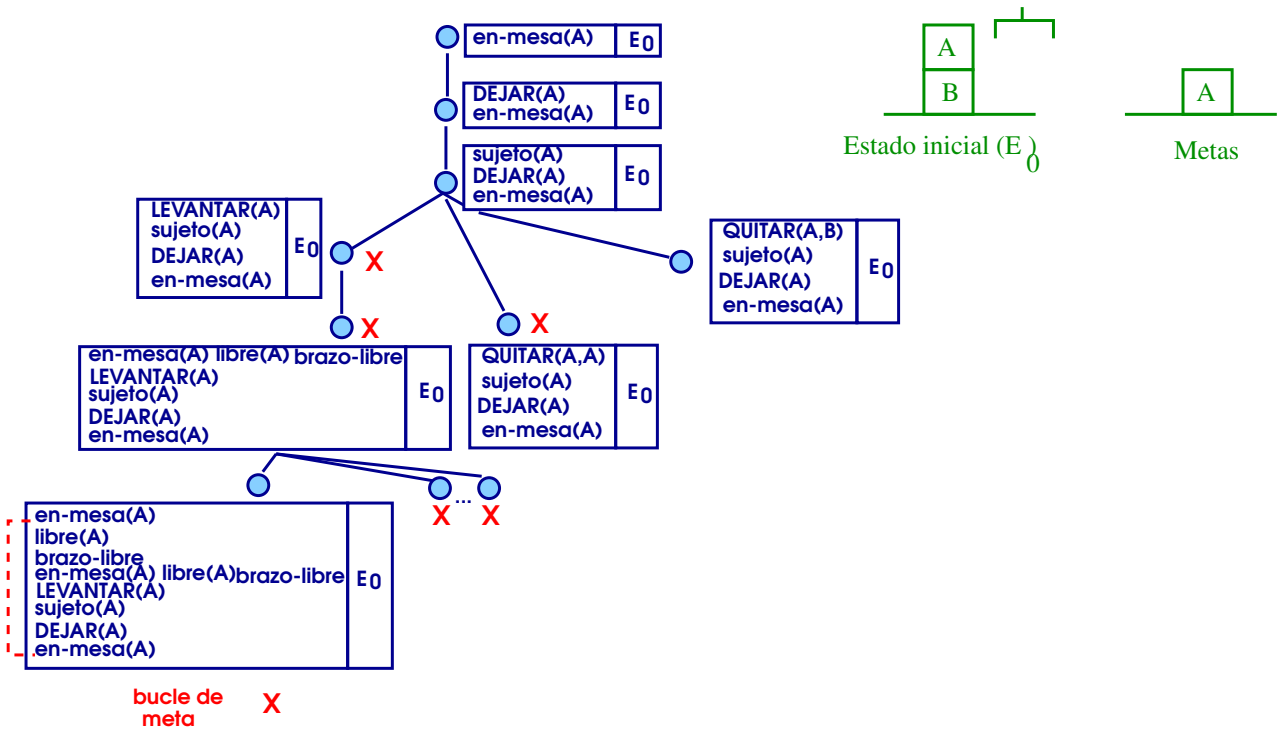
Estado inicial (E_0)

Metas

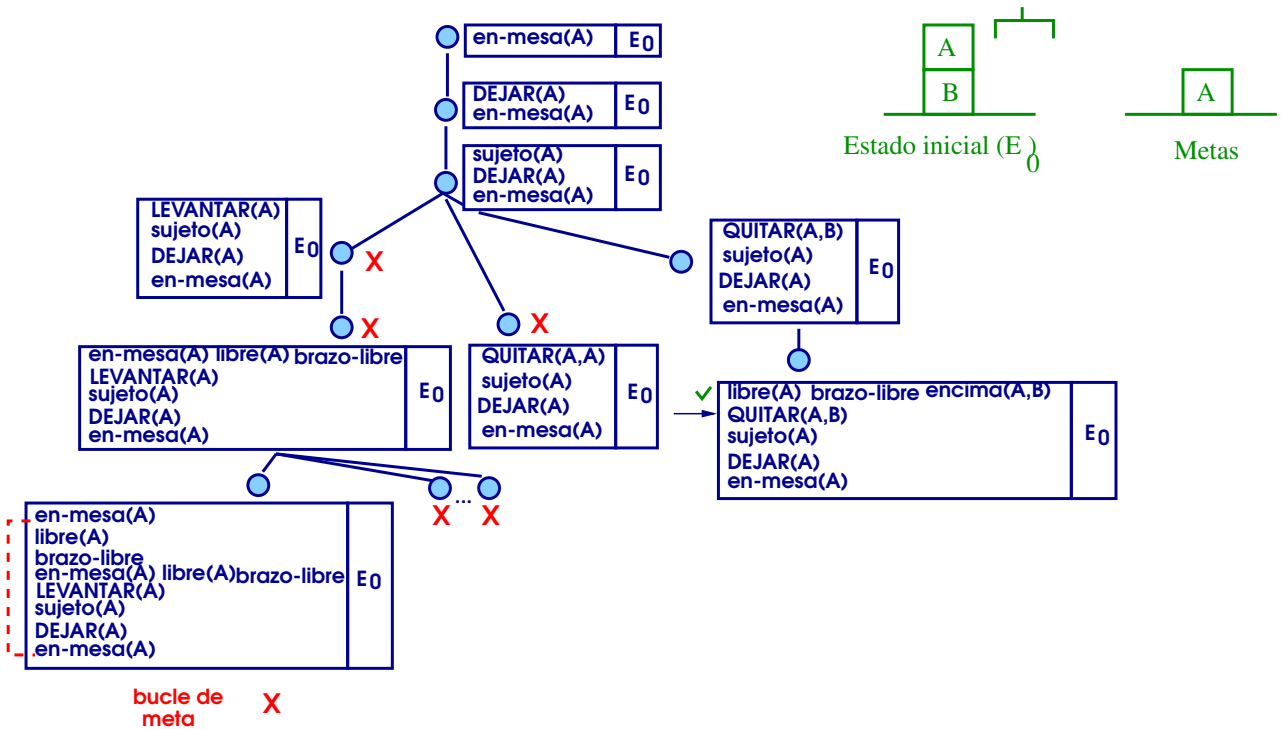
Ejemplo de STRIPS



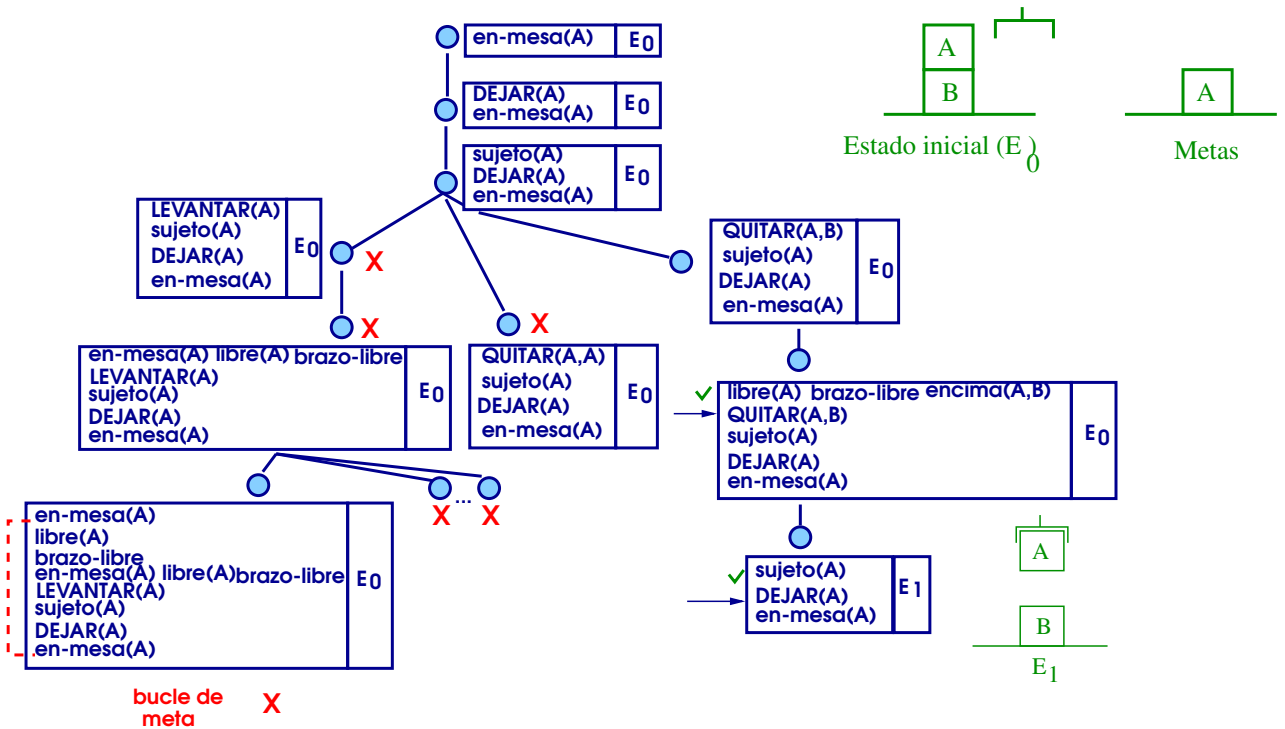
Ejemplo de STRIPS



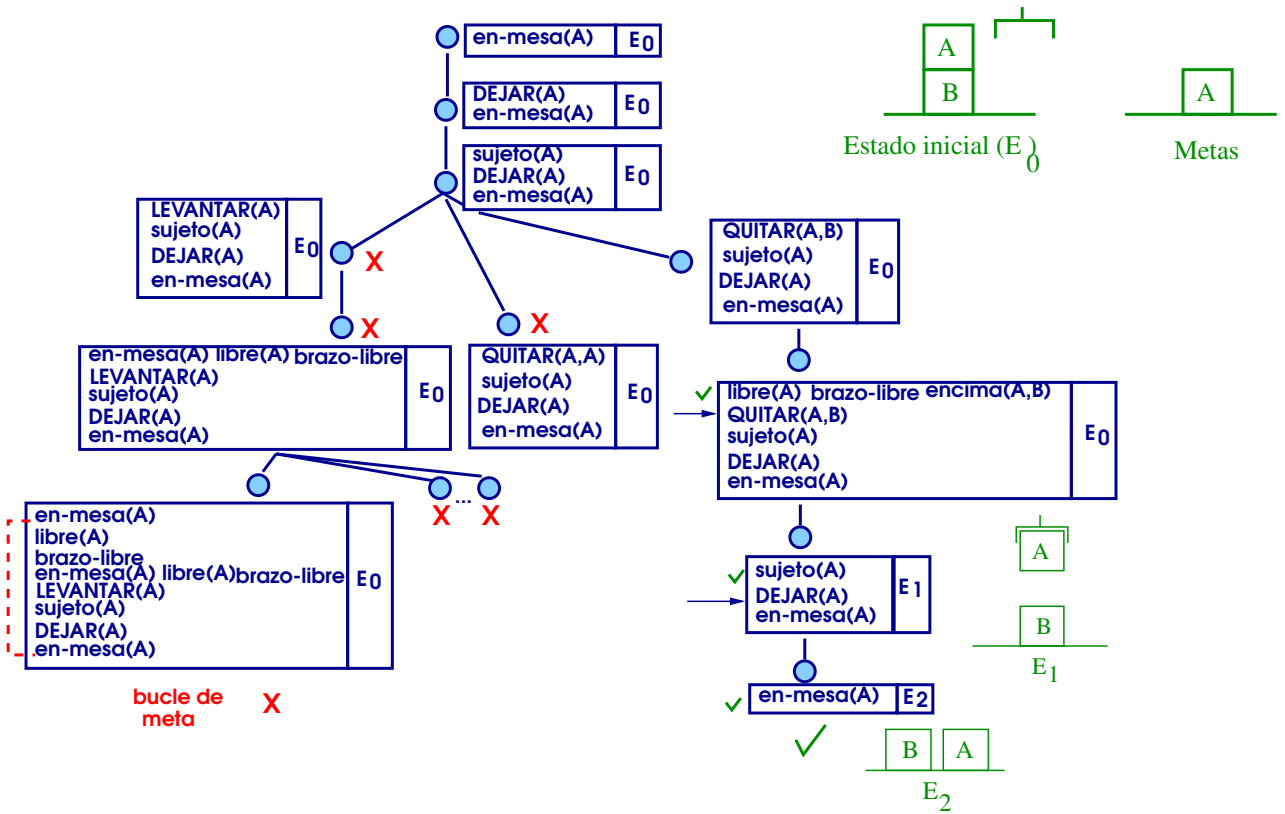
Ejemplo de STRIPS



Ejemplo de STRIPS



Ejemplo de STRIPS



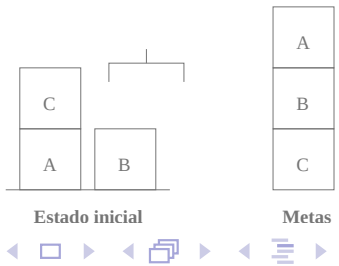
Problema: linealidad

- 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

Problema del cohete chino

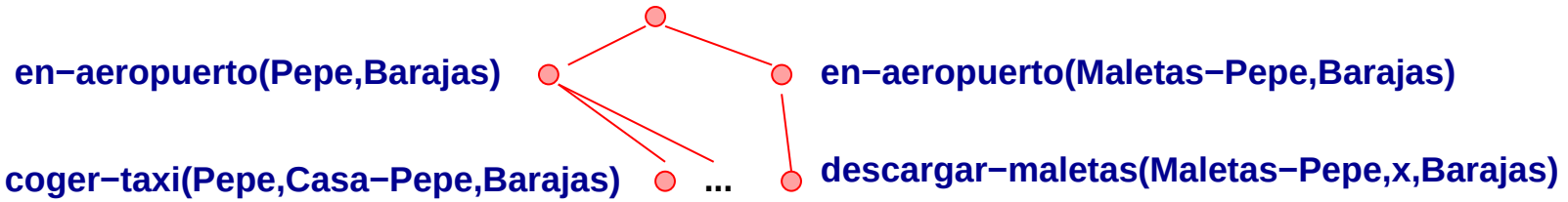
  - no óptima: no encuentra la solución óptima

Anomalía de Sussman

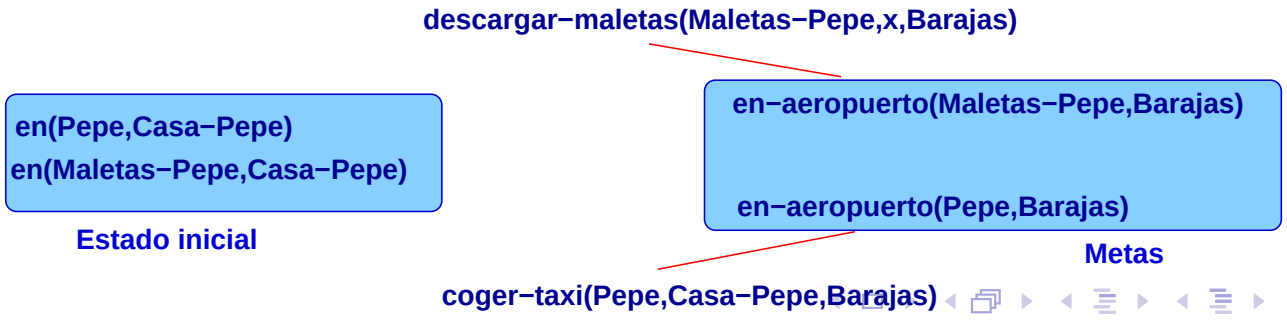


Planificación no lineal

- Consideración de varias metas al mismo tiempo
  - Pensar con un conjunto de metas: no es necesario generar completamente un plan de una meta para estudiar al mismo tiempo el resto



  - Expandir un grafo en el que las metas (y los operadores) son nodos y se pueden seleccionar las metas en cualquier orden



Planificación no lineal

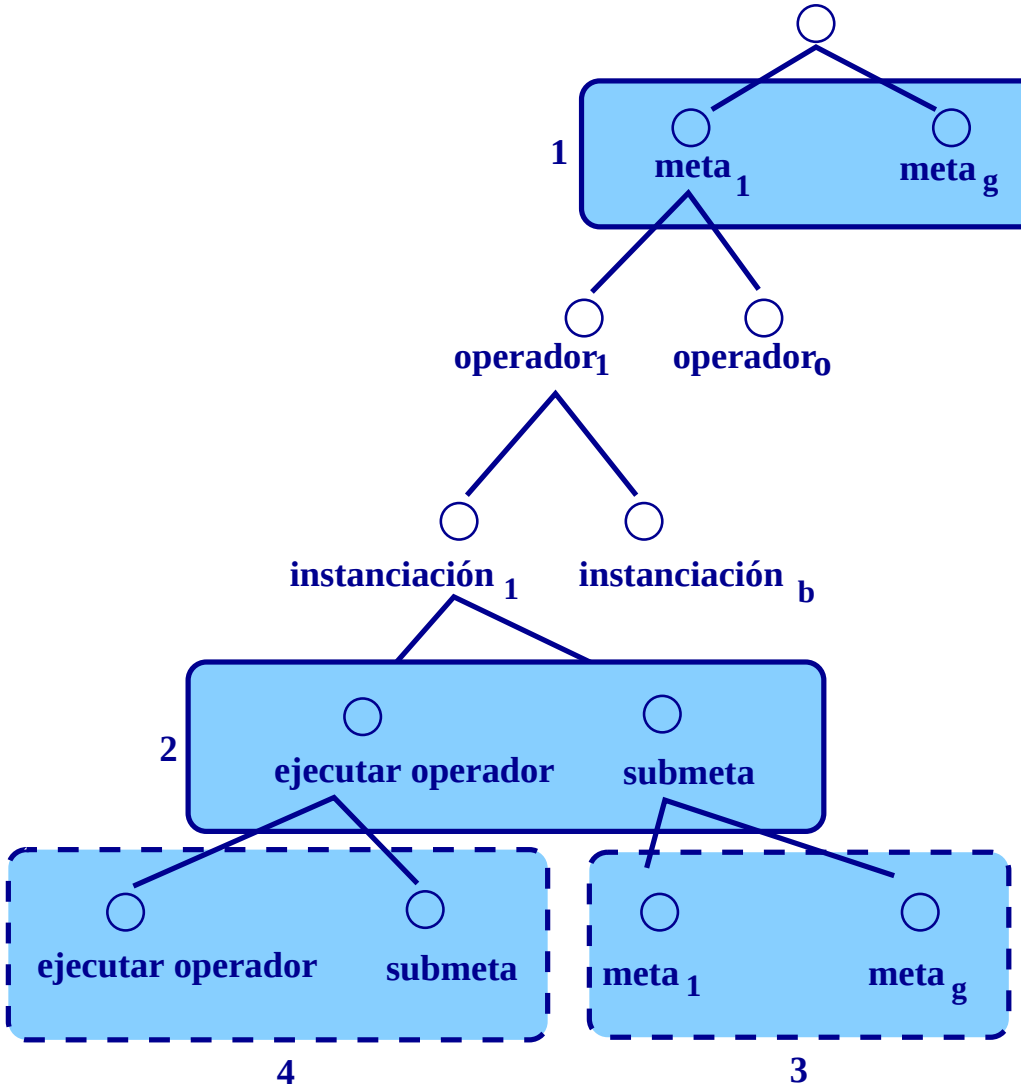
- Según espacio de problemas:
  - Estados (STRIPS, PRODIGY): nodos del árbol representan estados
  - Planes (NOAH, TWEAK, UCPOP, SNLP, O-PLAN): nodos del árbol representan planes
- Según plan generado:
  - Orden total: secuencia única de operadores
  - Orden parcial: múltiples secuencias posibles
- Según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

No hay una técnica mejor que otra

PRODIGY

- Análisis medios-fines con búsqueda hacia atrás (bidireccional) [Veloso *et al.*, 1995]
- Las metas se tratan como un conjunto
- Decisiones:
  - Meta: qué meta escoger
  - Operador: qué operador utilizar para obtener una meta
  - Instanciación de operador: qué valores asignar a las variables del operador
  - Ejecutar un operador o trabajar en alguna submeta
- Se puede definir conocimiento de control explícito para tomar las decisiones

Árbol de búsqueda genérico de PRODIGY



Elige una meta

Elige un operador

Elige una instanciación

Decide si ejecutar un operador o trabajar en una submeta

Algunas definiciones

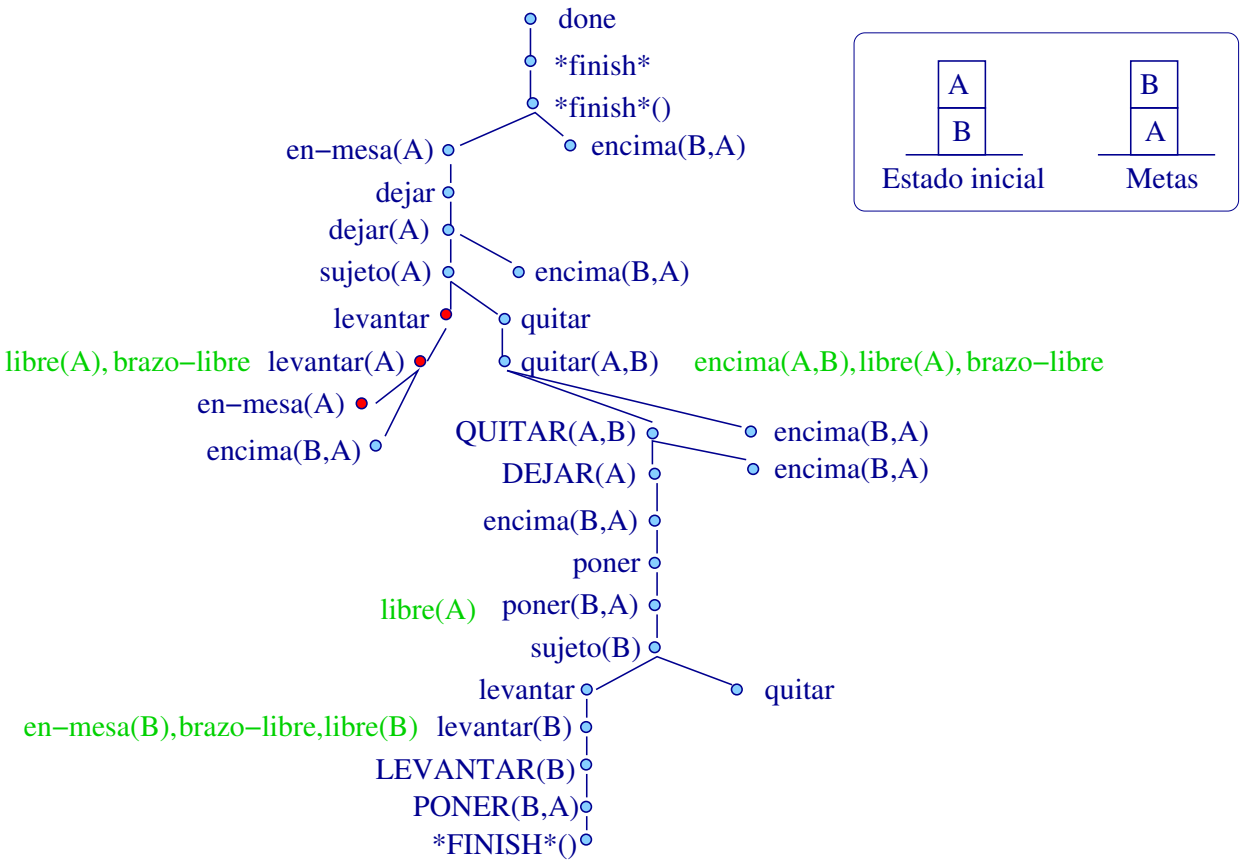
- **Una meta está pendiente** si es una precondición de un operador seleccionado (incluyendo las metas iniciales) que no es cierta en el estado actual
- **Un operador es aplicable** cuando todas sus precondiciones son ciertas en el estado actual
- **Operadores relevantes a una meta** son aquellos que, si se ejecutara una de sus instanciaciones, conseguiría que la meta fuera cierta en el estado
- **Camino sin salida**
  - Bucle de meta
  - Todas las opciones han sido caminos sin salida
  - No hay ninguna opción
  - Se ha sobrepasado el límite de profundidad o tiempo

Algoritmo de planificación de PRODIGY4.0

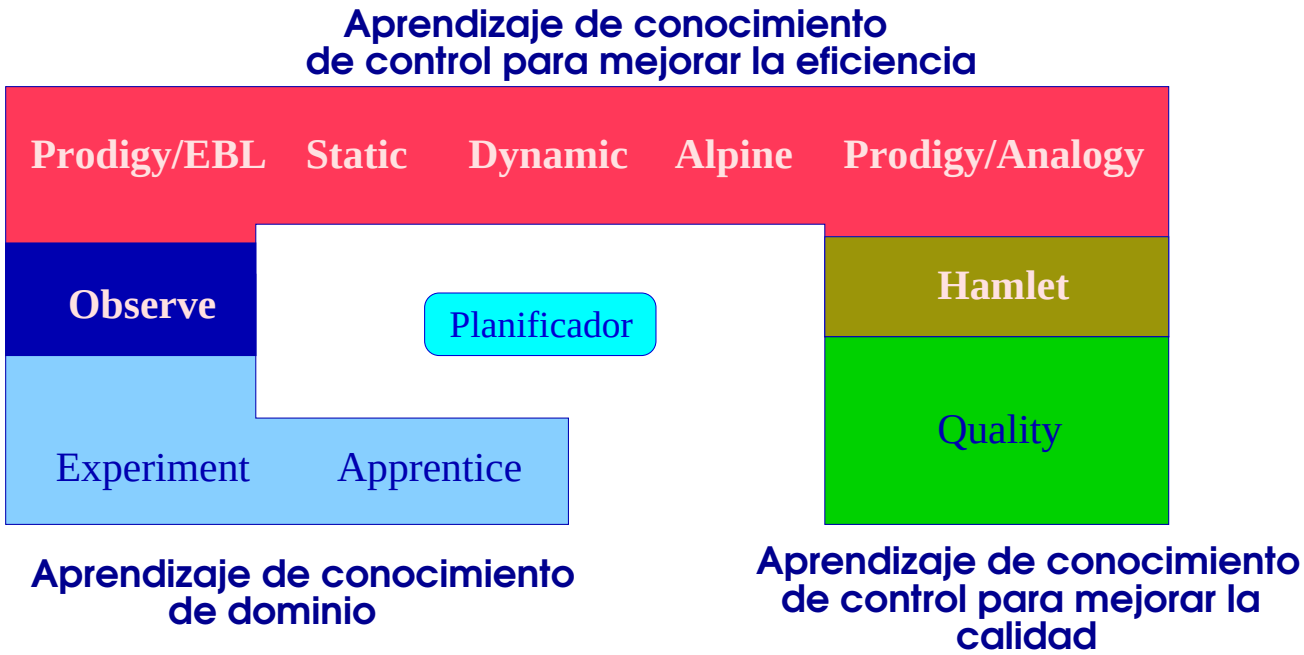
Prodigy $(E, \mathcal{M}, \mathcal{O}, \mathcal{C})$

Mientras que las metas $\mathcal{M}$ no sean ciertas en el estado E Y
no se hayan sobrepasado los recursos límite (tiempo o nodos) Y
no se hayan explorado todos los posibles nodos
Si hay un camino sin salida, entonces retroceder
Si no, Calcular el conjunto de operadores aplicables, $\mathcal{A}$
Elegir una meta $M \in \mathcal{M}$ o un operador $A \in \mathcal{A}$ de acuerdo a $\mathcal{C}$
Si se ha seleccionado M, entonces:
Generar el conjunto $\mathcal{O}' \in \mathcal{O}$ de los operadores relevantes
Elegir un operador $O \in \mathcal{O}'$, de acuerdo a $\mathcal{C}$
Generar el conjunto de instanciaciones del operador O, $\mathcal{O}_i$
Elegir un operador instanciado $O_i \in \mathcal{O}_i$, de acuerdo a $\mathcal{C}$
Si no, (se ha seleccionado un A):
Asignar E al estado después de *Ejecutar A*
Calcular el conjunto de metas pendientes $\mathcal{M}$

Ejemplo de PRODIGY



Arquitectura de PRODIGY



Más información en

- Libros: [Allen *et al.*, 1990, Ghallab *et al.*, 2004]
- Otras referencias: [Fikes and Nilsson, 1971, Newell *et al.*, 1972, Veloso *et al.*, 1995, Weld, 1994]

Indice

Indice

Indice

Referencias

James F. Allen, James Hendler, and Austin Tate (eds.).

Readings in Planning.

Morgan Kaufmann, 1990.

Daniel Borrajo, Natalia Juristo, Vicente Martínez, and Juan Pazos.

Inteligencia Artificial. Métodos y Técnicas.

Centro de Estudios Ramón Areces, Madrid, 1993.

George W. Ernst and Allen Newell.

GPS: A Case Study in Generality and Problem Solving.

ACM Monograph Series. Academic Press, New York, NY, 1969.

Richard E. Fikes and Nils J. Nilsson.

Strips: A new approach to the application of theorem proving to problem solving.

Artificial Intelligence, 2:189–208, 1971.


Malik Ghallab, Dana Nau, and Paolo Traverso.

Automated Task Planning. Theory & Practice.

Morgan Kaufmann, 2004.


Cordell C. Green.

Application of theorem proving to problem solving.

In *Proceedings of the First International Joint Conference on Artificial Intelligence*, pages 219–239, Washington DC, 1969.

Also in Readings in Planning.


John McCarthy and Patrick Hayes.

Some philosophical problems from the standpoint of artificial intelligence.

Machine Intelligence, 4:463–502, 1969.

Also in Readings in Planning.


Allen Newell, Herbert A. Simon, and J. Shaw.

Human Problem Solving.
Prentice-Hall, Englewood Cliffs, NJ, 1972.


Elaine Rich and Kevin Knight.
Inteligencia Artificial.
McGraw-Hill, Inc., 1994.
Segunda edición.


Manuela Veloso, Jaime Carbonell, Alicia Pérez, Daniel Borrajo, Eugene Fink, and Jim Blythe.
Integrating planning and learning: The PRODIGY architecture.
Journal of Experimental and Theoretical AI, 7:81–120, 1995.


Daniel S. Weld.
An introduction to least commitment planning.
AI Magazine, 15(4):27–61, 1994.