Razonamiento y Planificación Automática

Dr. Luis Peña

Planificadores STRIPS

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

La Universidad en Internet

¿Qué hay en la sesión?

- Planificación STRIPS
  - Componentes del lenguaje STRIPS
  - Planificación “hacia-adelante”
  - Heurística STRIPS

Planificación: Representación de estados tipo STRIPS

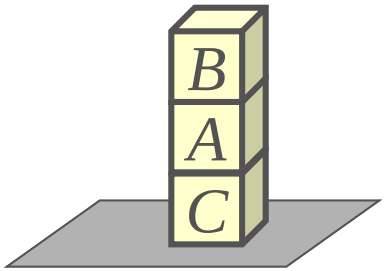
Representación básica tipo STRIPS:

- Un estado se describe por el conjunto de propiedades relevantes que se dan en él
- Propiedades relevantes para caracterizar estados en un mundo con 3 bloques

```
{ on(A,B), on(A,C), on(A,T), on(B,A), on(B,C), on(B,T),
  on(C,A), on(C,B), on(C,T), clear(A), clear(B), clear(C) }
```

- Ejemplo: propiedades del estado *S* de la derecha:

```
{ on(B,A), on(A,C), on(C,T), clear(B) }
```



- No todas las combinaciones de propiedades son "consistentes":

```
{ on(B,A), on(A,B) }
```
no representa un estado válido

Operadores STRIPS

Operador STRIPS: <nombre>(<PC>,<A>,>E>)

- **Precondición *PC***: conjunto de propiedades de un estado que han de darse *antes* de poder aplicar el operador
- **Lista de adiciones *A***: conjunto de las *nuevas* propiedades del estado resultado de aplicar el operador
- **Lista de eliminaciones *E***: conjunto de las propiedades del estado que *dejan* de darse *después* de aplicar el operador (subconjunto de *PC*)
- Ejemplos:

`move(A,B,C)`

PC: `On(A,B), Clear(A), Clear(C)`

E: `On(A,B), Clear(C)`

A: `On(A,C), Clear(B)`

`move(B,A,T)`

PC: `On(B,A), Clear(B)`

E: `On(B,A)`

A: `On(B,T), Clear(A)`

- Operadores del mundo con 3 bloques:

{ `move(A,B,C), move(A,B,T), move(A,C,B), move(A,C,T),`
`move(B,A,C), move(B,A,T), move(B,C,A), move(B,C,T),`
`move(C,B,A), move(C,B,T), move(C,A,B), move(C,A,T),`
`move(A,T,B), move(A,T,C), move(B,T,A), move(B,T,C),`
`move(C,T,A), move(C,T,B)` }

Ejecución de operadores STRIPS

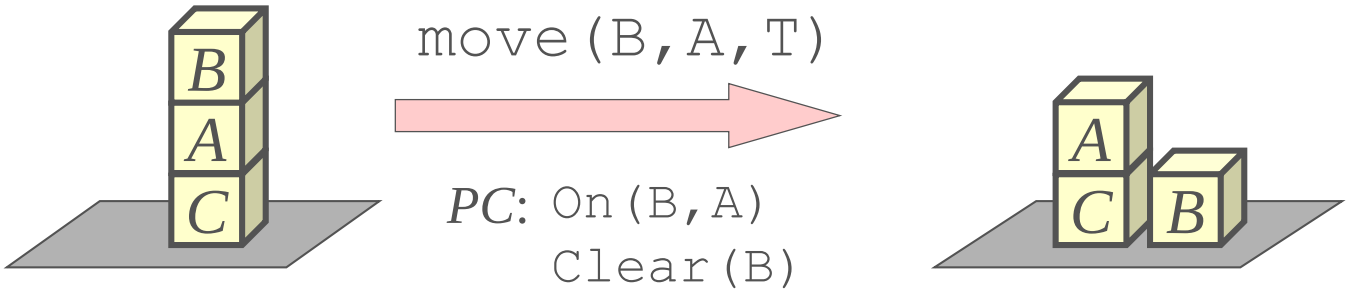
- Operadores *aplicables*:
  - un operador *Op* es *aplicable* en un estado *S* si se cumple su PC_{Op} en *S*, es decir si

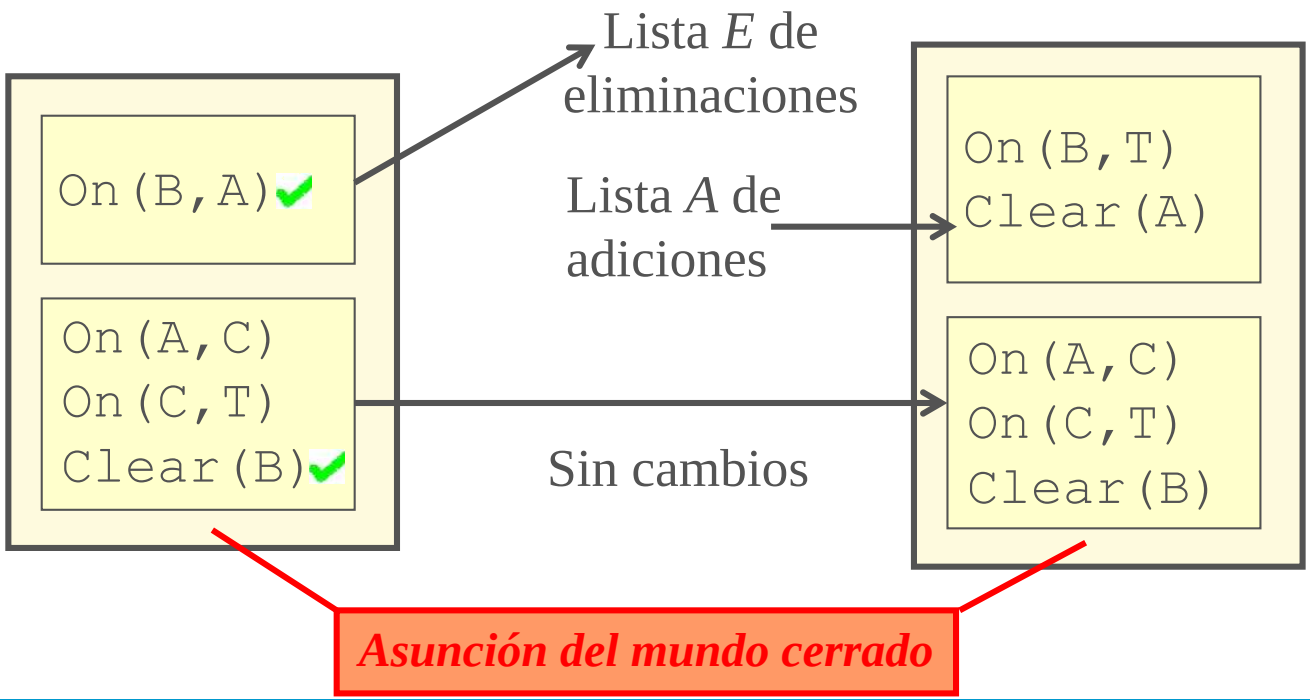
$$PC_{Op} \subseteq S$$

- *Ejecución* de operadores:
  - al ejecutar un operador aplicable *Op* en un estado *S*, se eliminan las propiedades de la lista *E* del estado *S* y se añaden las propiedades de la lista *A*

$$S' \leftarrow S - E_{Op} \cup A_{Op}$$

Ejemplo: ejecución de un operador STRIPS



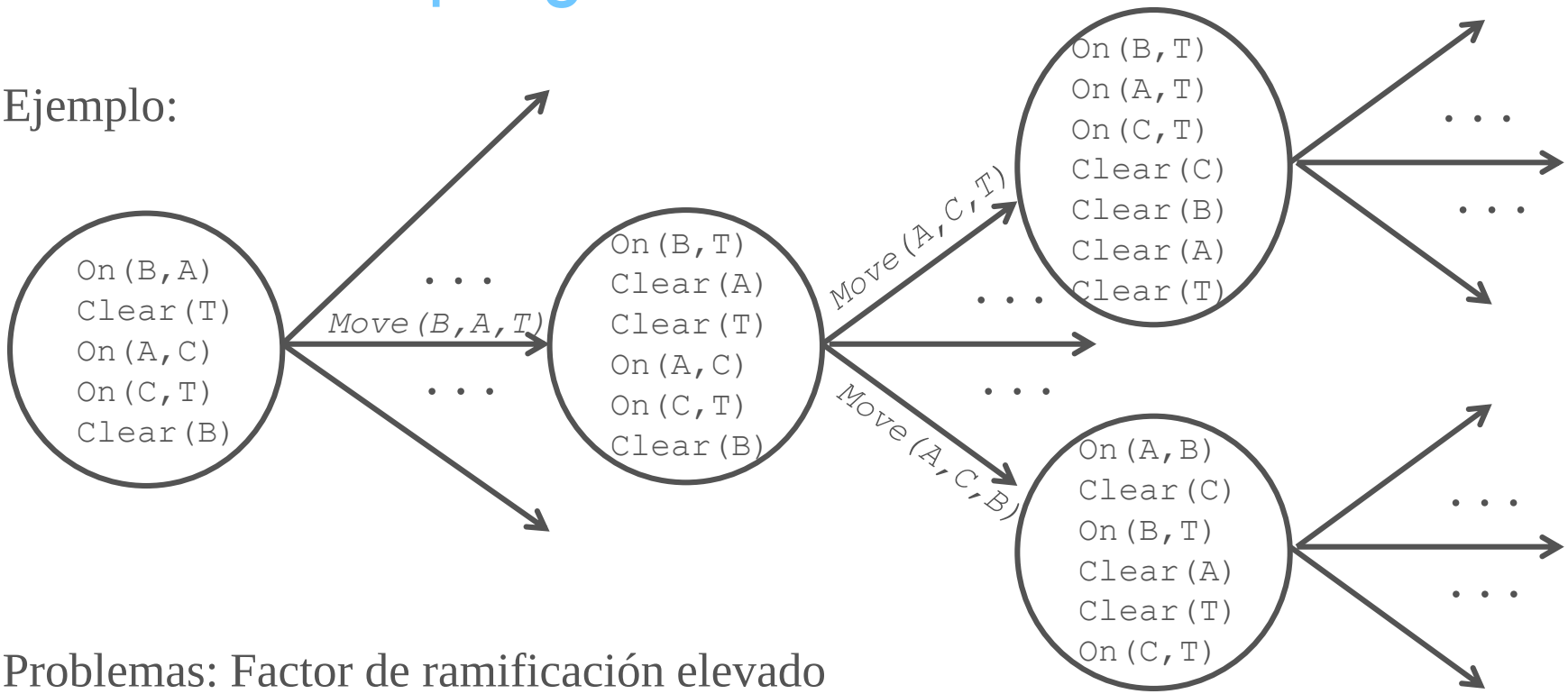


Algoritmos: Planificación progresiva

- Espacio de búsqueda:
  - Partiendo del estado inicial, se van ejecutando los operadores aplicables
  - Los operadores sirven para implementar la función *expandir*
    - *Expandir(S)*: Conjunto cuyos elementos son los conjuntos de propiedades obtenidas al ejecutar los operadores aplicables en *S*
- Algoritmos:
  - Búsqueda no informada: amplitud, profundidad, ...
  - Búsqueda heurística: A^*, …

Planificación progresiva

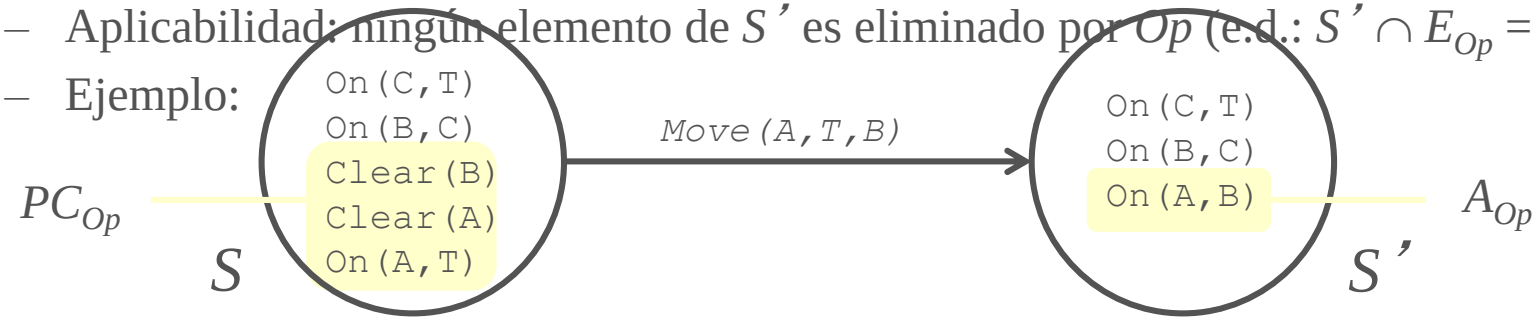
- Ejemplo:



- Problemas: Factor de ramificación elevado
  - El estado inicial se describe por *todas las propiedades* que podrían ser relevantes (p.e. las posiciones de 10 bloques)
  - La descripción del estado meta suele ser parcial, es decir sólo contiene lo que *realmente* importa (p.e. las posiciones de 3 bloques)

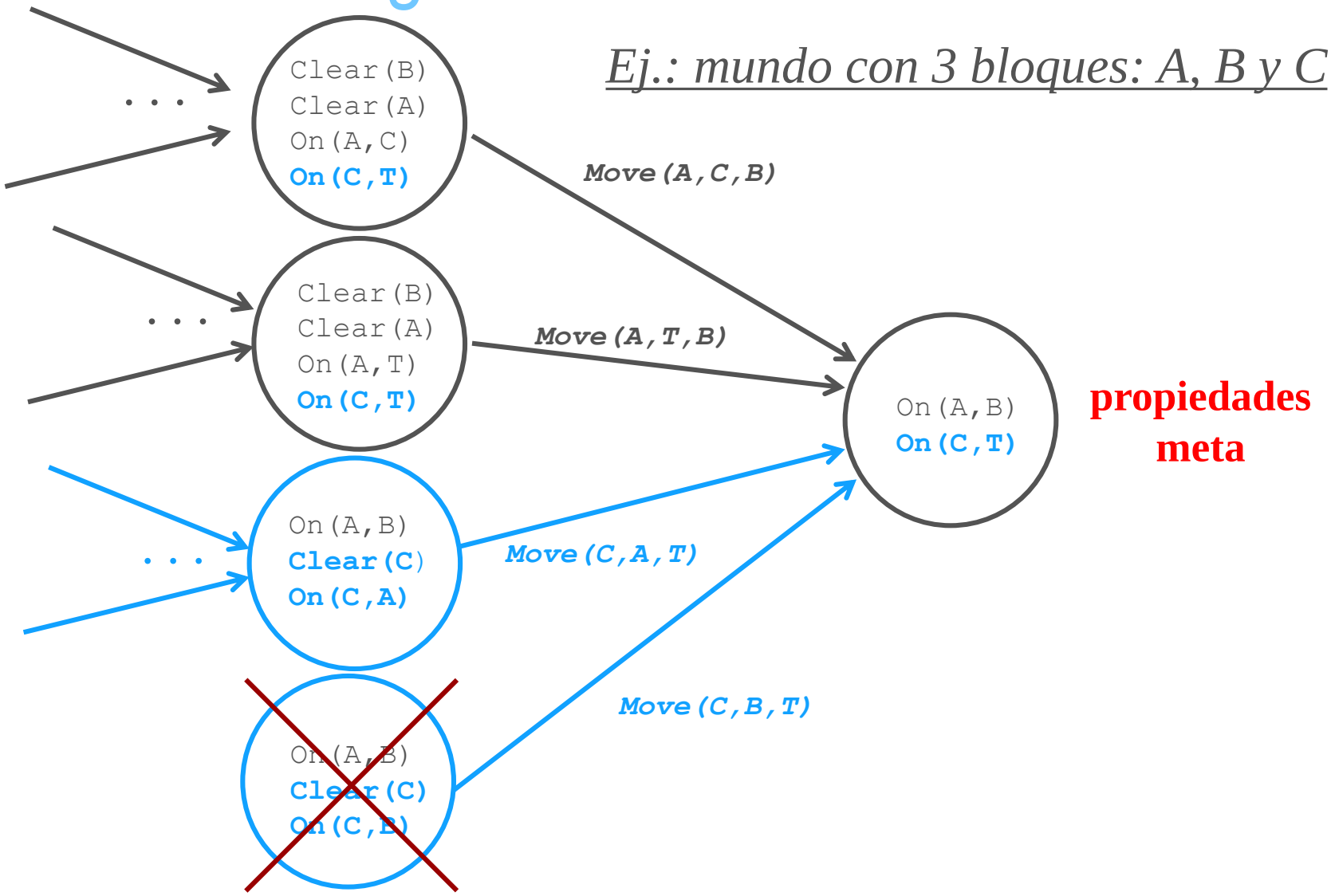
Planificación regresiva

- **Idea**: aplicar operadores "al revés"
  - *regresar* conjuntos de propiedades meta por operadores, dando lugar a otros conjuntos de propiedades meta
  - reducir la "diferencia" entre los conjuntos de propiedades meta y el estado inicial, hasta que todas las propiedades meta se den en el estado inicial
- **Regresar** un estado S' por un operador *Op:* Calcular el conjunto de propiedades más débil S para que la aplicación de *Op* lleve a S' (*precondición más débil*):
  - Eliminar la lista A de S' y añadir los elementos de PC: $S \leftarrow S' - A_{Op} \cup PC_{Op}$
  - Aplicabilidad: ningún elemento de S' es eliminado por *Op* (e.d.: $S' \cap E_{Op} = \varnothing$)
  - Ejemplo:



  - Cuidado: No todos los estados que resultan de la regresión son "consistentes"

Planificación regresiva



Heurística STRIPS

¿Cómo combatir aún más la complejidad de la planificación?

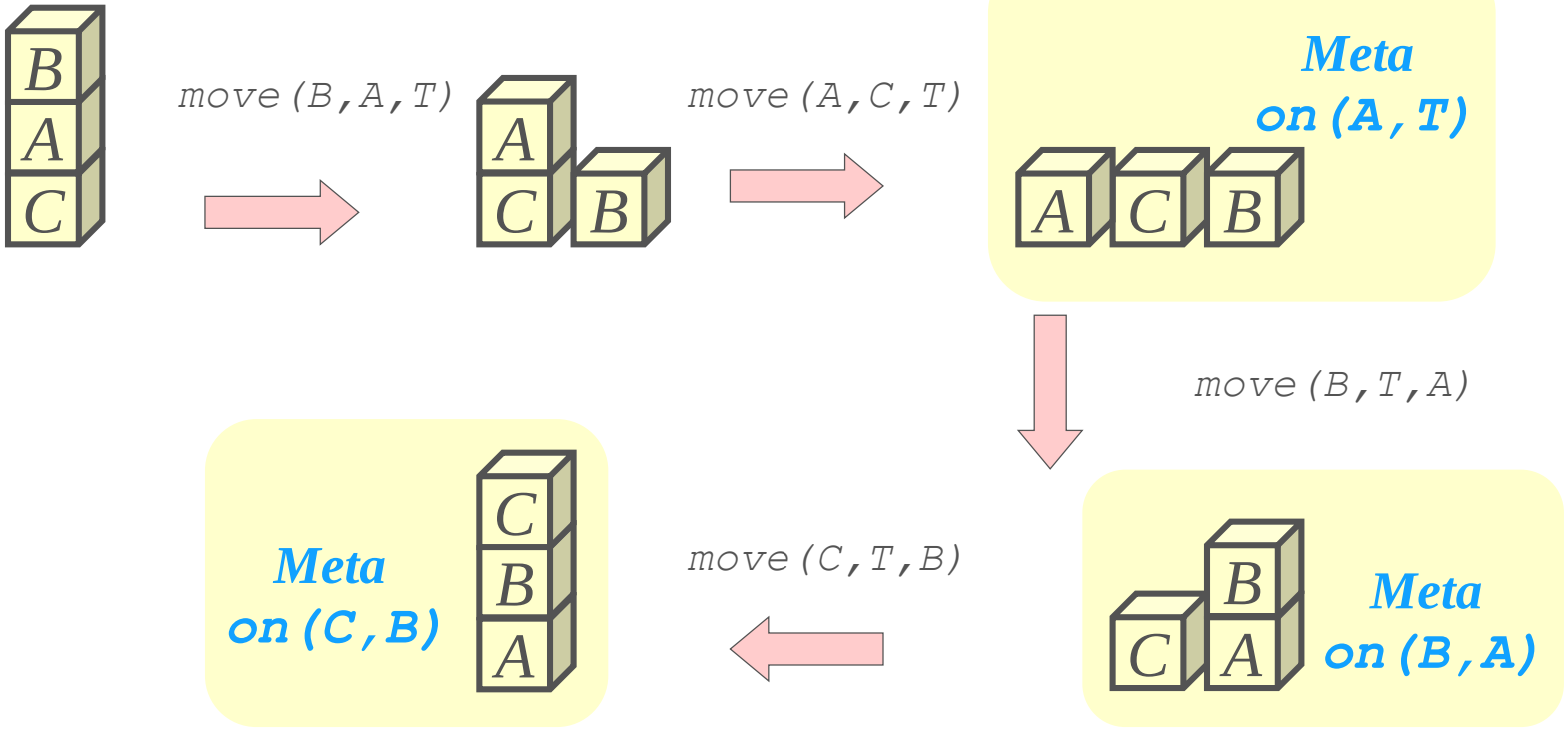
- **Heurística STRIPS**:
  - Generar "subplanes" para alcanzar *cada una de las propiedades meta aparte*
  - El plan final es una *concatenación* de estos subplanes
- **Planificación tipo STRIPS**:
  - Elegir una propiedad P_i que no se cumple en S_0
  - Generar un subplan para al propiedad meta P_i a partir del estado inicial S_0
    - genera una secuencia de operadores de forma regresiva (si es posible)
    - ejecutar esta secuencia a S_0, dando lugar a una nueva situación S_1
  - Elegir propiedad P_j que no se cumple en S_1
  - Generar un subplan para la propiedad meta P_j a partir del estado actual S_1
    - genera una secuencia de operadores de forma regresiva (si es posible)
    - ejecutar esta secuencia a S_1, dando lugar a una nueva situación S_2
  - . . .
  - Si una propiedad meta P_k (alcanzada por un subplan anterior) ha sido "destruida" al alcanzar posteriormente una propiedad meta P_l, entonces "rehacer" P_k

Planificación con la heurística STRIPS

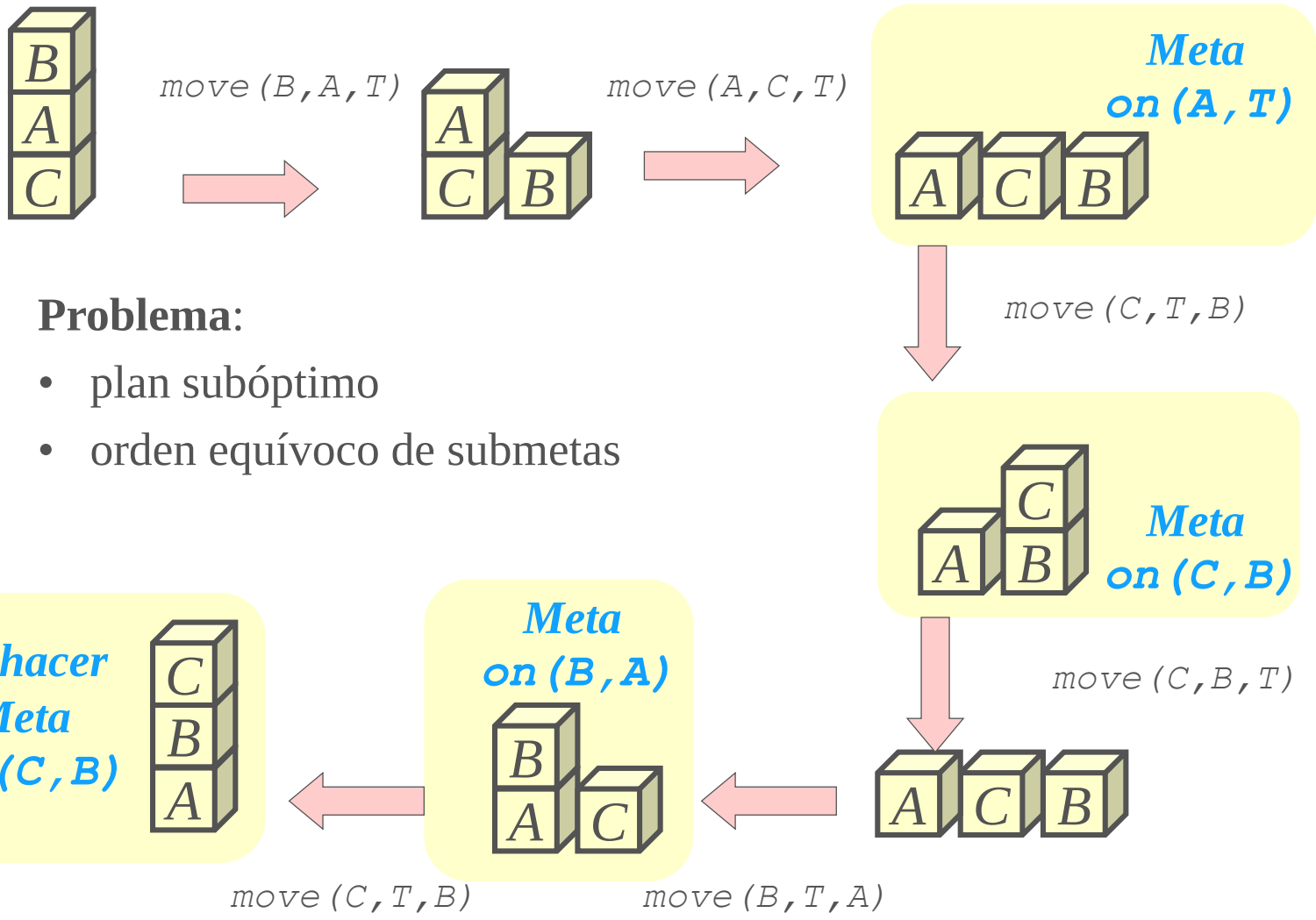
Estado inicial

Estado meta

on (A, T)
on (B, A)
on (C, B)



Planificación con la heurística STRIPS: Problemas



Problema:

- plan subóptimo
- orden equívoco de submetas

Planificación con la heurística STRIPS

Algoritmo recursivo no determinista:

- Entradas:
  - El estado actual *S*
  - Estado meta *Metas*
  - *Plan* generado para llegar a *S*
- Consulta: `STRIPS(`S_0`,Metas,[])`

- Salidas:
  - Un estado *SMeta* en el que se cumplen todos los objetivos de *Metas*
  - *Plan* para transformar *S* en *SMeta*
  - Si no hay tal plan, devuelve false

STRIPS(*S*,*Metas*,*Plan*) devuelve (*SMeta*, *PlanMeta*) ó *false*

Mientras que $Metas \not\subseteq S$ **Hacer** % Mientras haya *Metas* no satisfechas en *S*

(1) ***Elegir*** $M \in Meta$ tal que $M \notin S$ % *M* no está satisfacha en *S*

(2) ***Elegir*** operador *Op* tal que $M \in A_{Op}$ % *Op* puede alcanzar *M*

(3) $(S,Plan) \leftarrow$ STRIPS$(S,PC_{Op},Plan)$ % Alcanzar precond. de *Op*

(4) **Si** STRIPS devuelve *false* **Entonces *falla***

(5) $S \leftarrow S - E_{Op} \cup A_{Op}$ % aplicar el *Op* al nuevo estado *S* (resultado de (3))

(6) $Plan \leftarrow Plan + Op$ % añadir el *Op* al final de nuevo *Plan* (resultado de (3))

Fin

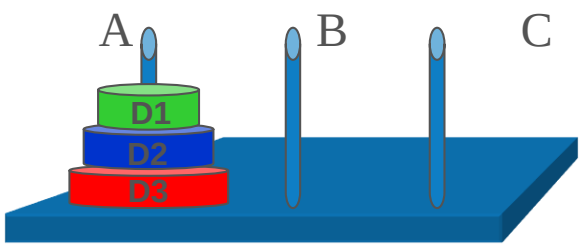
Devolver(*S*,*Plan*)

Planificación con la heurística STRIPS

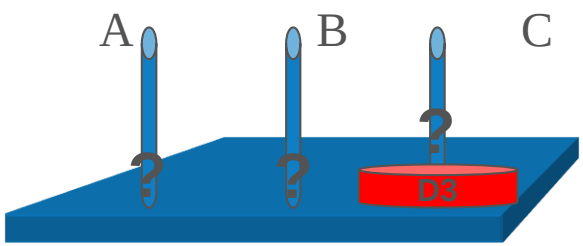
Simulación "a mano" del la planificación con la heurística STRIPS

- Generación de un **Árbol "*y*/*o*"**
  - Nodos "*y*": **propiedades**
  - Nodos "*o*": **operadores**

- **Ejemplo**: Problema (simplificado) de las Torres de Hanoi



Estado inicial



Estado meta

Ejemplo: Dominio de las Torres de Hanoi

- 3 discos (D1, D2, D3) y 3 agujas (A, B, C)

- Conjunto de **propiedades**:
  - *on*(D1,D2), *on*(D1,D3), *on*(D1,A), *on*(D1,B), *on*(D1,C), *on*(D2,D3), *on*(D2,A), …
  - *clear*(D1), *clear*(D2), *clear*(D3): "no hay nada encima del disco"
  - *clear*(A), *clear*(B), *clear*(C): "no hay ningún disco en la aguja"
  - **NO** incluye propiedades no compatibles con el dominio:
    - *on*(D2,D1), *on*(D3,D1), *on*(D2,D1), *on*(D3,2), *on*(A,D1), *on*(B,D3), …

- Conjunto de **operadores**:
  - P.e. : *mover*(D1,D2,D3)

    PC: { *clear*(D1), *clear*(D3), *on*(D1,D2) }

    E: { *on*(D1,D2), *clear*(D3) }

    A: { *on*(D1,D3), *clear*(D2) }

Ejemplo: Dominio de las Torres de Hanoi (más formal)

- **Individuos**: I = {D1, D2, D3, A, B, C}
- Predicados auxiliares:
  - *menor*(D1,D2), *menor*(D2,D3), *menor*(D3,A), *menor*(D3,B), *menor*(D3,C)
  - *menor** es el cierre transitivo de *menor*
- Conjunto de **propiedades**:
  - $P = \{ on(x,y) : x,y \in I \wedge menor^*(x,y) \} \cup \{ clear(x) : x \in I \}$
- Conjunto de **operadores**:
  - $O = \{ mover(x,y,z) : x \in \{D1, D2, D3\} \wedge y,z \in I \wedge menor^*(x,y) \wedge menor^*(x,z) \wedge y \neq z \}$
  - Plantilla para los efectos de los **operadores**:

mover(“x”,“y”,“z”)

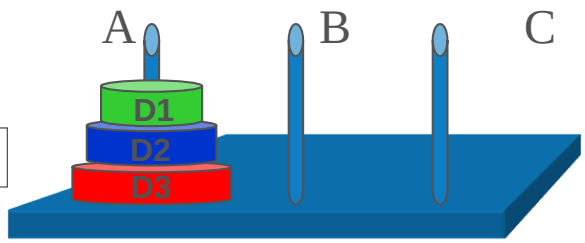
PC: { *clear*(“x”), *clear*(“z”), *on*(“x”,“y”) }

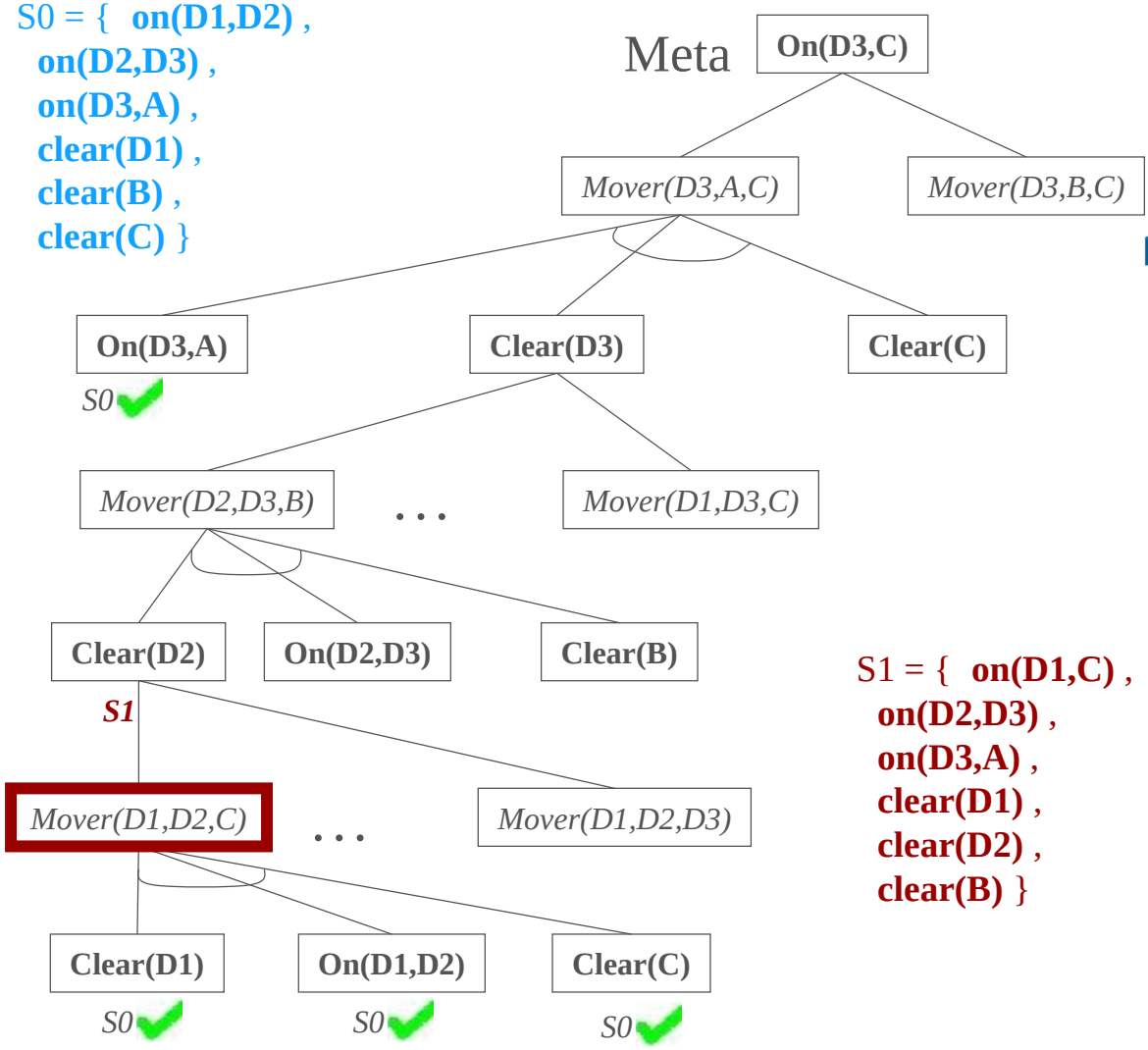
E: { *on*(“x”,“y”), *clear*(“z”) }

A: { *on*(“x”,“z”), *clear*(“y”) }

Ejemplo: Torres de Hanoi

S0 = { **on(D1,D2) ,**
on(D2,D3) ,
on(D3,A) ,
clear(D1) ,
clear(B) ,
clear(C) }

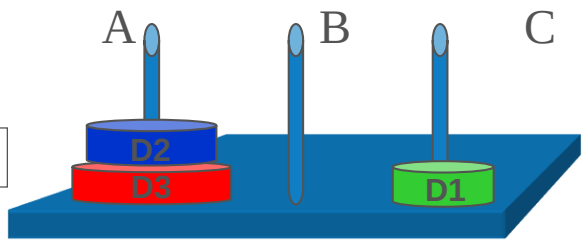


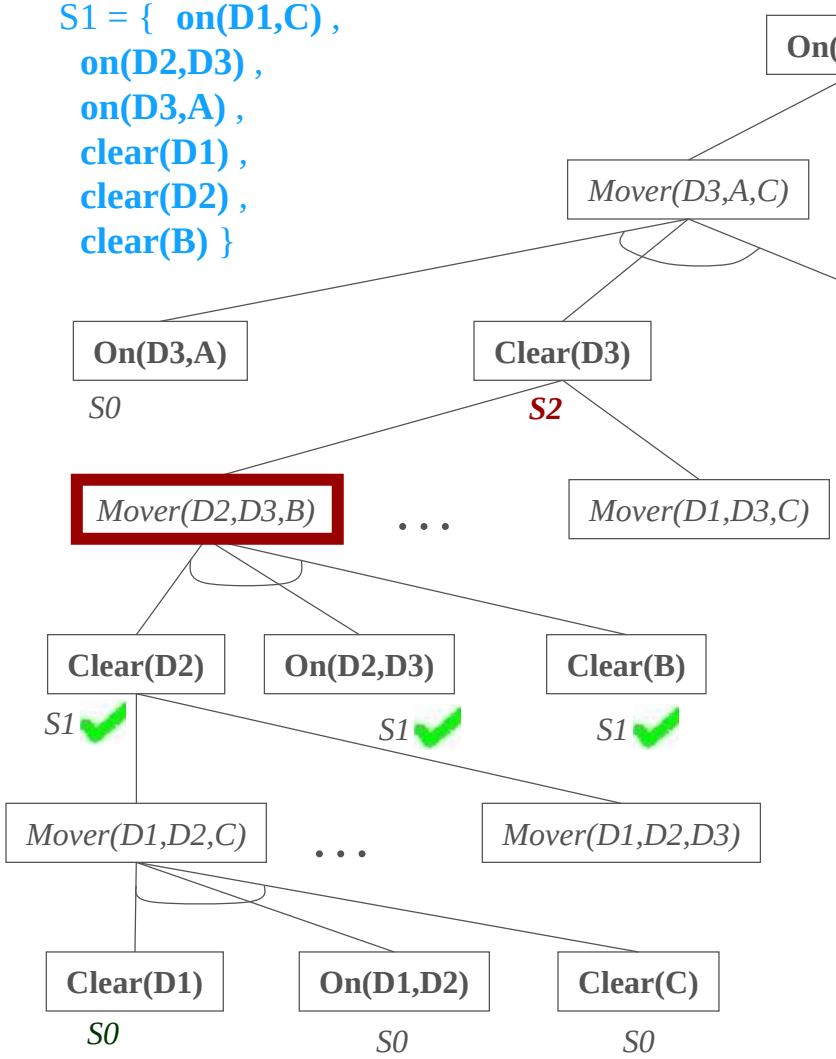


S1 = { **on(D1,C) ,**
on(D2,D3) ,
on(D3,A) ,
clear(D1) ,
clear(D2) ,
clear(B) }

Ejemplo: Torres de Hanoi

S1 = { **on(D1,C) ,**
on(D2,D3) ,
on(D3,A) ,
clear(D1) ,
clear(D2) ,
clear(B) }

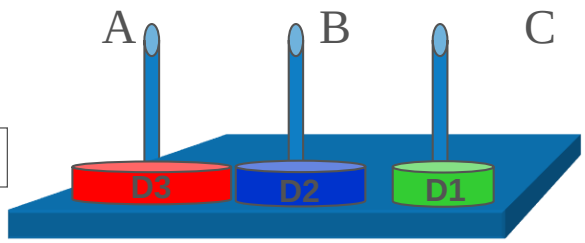




S2 = { **on(D1,C) ,**
on(D2,B) ,
on(D3,A) ,
clear(D1) ,
clear(D2) ,
clear(D3) }

Ejemplo: Torres de Hanoi

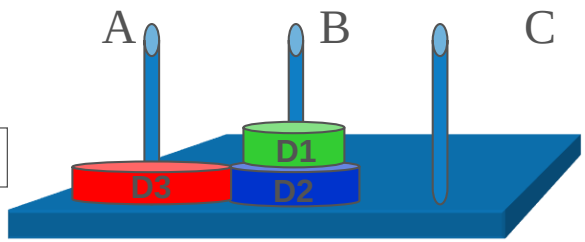
S2 = { **on(D1,C) ,**
on(D2,B) ,
on(D3,A) ,
clear(D1) ,
clear(D2) ,
clear(D3) }



S3 = { **on(D1,D2) ,**
on(D2,B) ,
on(D3,A) ,
clear(D1) ,
clear(D3) ,
clear(C) }

Ejemplo

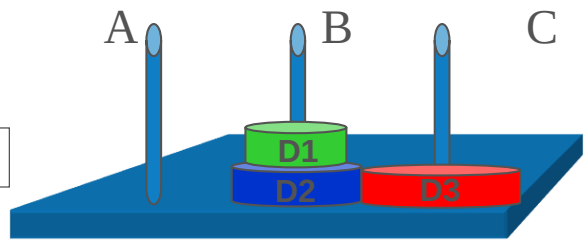
S3 = { **on(D1,D2) ,**
on(D2,B) ,
on(D3,A) ,
clear(D1) ,
clear(D3) ,
clear(C) }



S4 = { **on(D1,D2) ,**
on(D2,B) ,
on(D3,C) ,
clear(D1) ,
clear(D3) ,
clear(A) }

Ejemplo: Torres de Hanoi

S4 = { **on(D1,D2) ,**
on(D2,B) ,
on(D3,C) ,
clear(D1) ,
clear(D3) ,
clear(A) }



Planificación con la heurística STRIPS: Problemas

Implementación real:

- En las implementaciones reales (deterministas) no hay oráculos
- Problema:
  - una mala elección de metas y/o operadores puede llevar a planes subóptimos
- Solución:
  - aplicar algoritmos de búsqueda para determinar el orden óptimo en la elección de operadores y metas
  - a costa de un aumento en la complejidad

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

unir

www.unir.net