Planificación con STRIPS

Bibliografía

- Nilsson, *Artificial Intelligence: A New Synthesis.* Cap. 22.
- Russell & Norvig, *Artificial Intelligence: A Modern Approach*. Cap. 11.

Introducción

- Objetivo: Desarrollar un proceso de planificación
  - » La planificación es un proceso de búsqueda y ejecución de una serie de acciones que permitan alcanzar un objetivo
  - » Ejemplos de agentes planificadores ya vistos
    - – Agentes basados en búsquedas
    - – Agentes basados en lógica
      - Cambiar una rueda, embarques de vuelos, etc.
- El lenguaje empleado en los problemas de planificación es STRIPS
  - » STRIPS: STanford Research Institute Problem Solver. [Fikes y Nilsson 1971]
  - » Los algoritmos que usan STRIPS aprovechan la estructura de problema gracias a su capacidad expresiva
- La Planificación clásica se restringe a entornos
  - » Observables
  - » Deterministas
  - » Finitos
  - » Estáticos
    - – los cambios sólo se producen al actuar los agentes
  - » Discretos
    - – En tiempo, acciones, objetos y efectos

Sintaxis

- Un método de planificación es un método de búsqueda sobre una estructura de datos (estados del proceso de planificación) que evoluciona con las acciones de un agente de planificación

- ESTADOS
  - » Los estados en un proceso de planificación se denominan DESCRIPCIONES DE ESTADO (DE)
    - – Compuestos de condiciones lógicas (predicados)
      - [En(RP,E), En(RR,M)]
    - – Sólo admiten literales simples, sin dependencias funcionales y sin variables no ligadas.
      - En(x, y), En(padre(Pedro), Madrid) -> NO ADMITIDOS
      - En(Avión, Barcelona), En(Tren, Ávila) -> ADMITIDOS

- OBJETIVOS
  - » Un objetivo es una *conjunción de literales básicos afirmados*
  - » Los objetivos se restringen a conjuntos de literales cuantificados existencialmente
    - – Una DE *s* satisface un objetivo *w* si *s* contiene todos los elementos en *w*
      - DE *s*: rico ^ famoso ^ miserable
      - Goal *w*: rico ^ famoso (se satisface)

Sintaxis, II

- OPERADORES o ACCIONES
  - » Un operador o regla STRIPS es un patrón que posee variables libres
  - » se especifica en términos de
    - – Las **precondiciones**, PC, que deben cumplirse antes de ser ejecutado
      - El conjunto PC está formado por literales.
        - » La acción correspondiente a un operador sólo se puede ejecutar en un estado si todos los literales en PC pertenecen también a la DE previa a la acción
        - » On(x,y) ^ Clear(x) ^ Clear(z)
    - – Las **consecuencias** (efectos) que se derivan cuando se ejecuta
      - Lista de borrado: Un conjunto, D, de literales
      - Lista de adición: Un conjunto, A, de literales
    - – Una forma alternativa de representación de los efectos de una regla es mediante una conjunción de literales positivos y negativos
  - » PC, D y A o, alternativamente, PC y EFECTOS constituyen un *esquema de acción.*

Obs.: en lo que sigue los efectos los consideraremos como listas de borrado y adición

Semántica

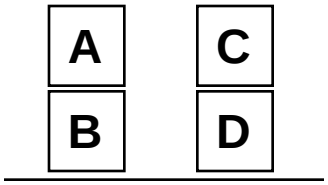
- Descripción de cómo los operadores afectan a los estados
- Existen diferentes métodos de búsqueda fundamentados en el uso de operadores STRIPS aplicables a los procesos de planificación
  - » Hacia delante: *progression planning*
  - » Hacia atrás: *regression planning*
- Dada una FBF objetivo *w*, los métodos de búsqueda tratan de encontrar una secuencia de acciones que proporcionen un estado del mundo descrito mediante alguna DE Δ tal que $\Delta \models w$. Si esto sucede se dice que la DE satisface el objetivo.
- $\Delta \models w$ si existe una sustitución σ tal que $w\sigma$ es una conjunción de literales todos los cuales están en Δ

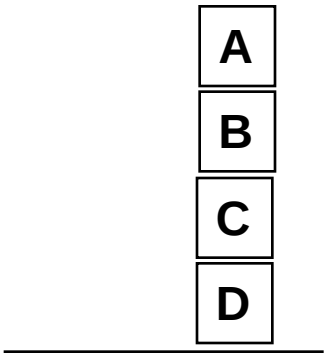
Semántica, II

- Un operador es aplicable si lo es en cualquier DE que satisfaga sus PC. En caso contrario es no aplicable (sin efecto)
- Partiendo de una DE *s* (*before-action*), el resultado de ejecutar un operador aplicable a *s* nos lleva a otra DE *s'* (*after-action*). En el proceso:
  - » Comprobado que los literales de PC están, a su vez, en la DE *s* (condición de aplicación del operador STRIPS)
  - » Borrados de la DE *s* los literales de D
  - » Añadidos a la DE *s* los literales en A
  - » Todos los literales no mencionados en D se acarrean a la DE s'. Este acarreo se denomina HIPÓTESIS STRIPS
    - – Hipótesis STRIPS: Cada literal no mencionado en los efectos permanece sin modificar

Semántica, III (ejemplo)

- Mundo de Bloques
  - » Dominio:
    - conjunto de bloques sobre un tablero
    - Sólo es posible situar un bloque sobre otro
    - Los bloques los mueve un brazo mecánico, uno cada vez
  - » Objetivo:
    - Construir uno o más montones de bloques





  - » Operadores:
    - On(bloque, posición)
      - On (C, Fl)
    - Move(bloque, pos_x, pos_y)
      - Requiere PC. Para mover un bloque requiere que no haya otro sobre él. Con el predicado CLEAR garantizaremos esta restricción
      - Move(B, Fl, C)
      - Move (B, C, C) acción espuria o degenerada (se puede solventar añadiendo PC de igualdad)

Ejemplo (cont.)

- Ej.: move(x,y,z)
  - » PC: On(x,y) ^ Clear(x) ^ Clear(z)
  - » D: Clear(z), On(x,y)
  - » A: On(x,z), Clear(y), Clear(Fl)

– Clear(posición)

- Predicado que toma valor T si ningún bloque está sobre *posición*
- Se debe interpretar este operador como
  - » “existe un espacio vacío sobre *posición* para trasladar un bloque”
  - » En tal caso Clear(Fl) = T siempre
- Dificultad 1: si en el operador *move* y o z corresponden al tablero (Fl), este no se puede despejar. Para evitarlo se podría incluir un operador adicional *moverSobreTablero*
- Ej.: moverSobreTablero(x, y)
  - » PC: On(x,y) ^ Clear(x)
  - » D: Clear (x)
  - » A: On(x, Fl), Clear(y)

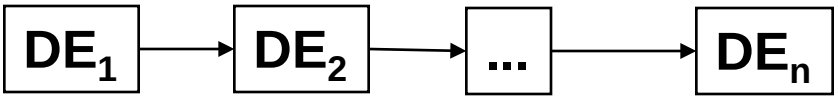
Ejemplo (cont.)

- Dificultad 2:
  - » O bien se permite mover *Fl* → espacio de búsqueda mayor
  - » Si no se permite hay que incluir un nuevo predicado, bloque(x), e incluir, como precondiciones de *move*, bloque(x) ^ bloque(z) (se evita poder mover *Fl*)

Forward Search Methods (*FSM*)

- Búsqueda en el espacio de estados desde el estado inicial
  - » El estado inicial del problema de planificación está descrito en términos de un conjunto de literales simples positivos (los literales que no aparecen se consideran F)
  - » Los operadores STRIPS aplicables deben cumplir las precondiciones. La DE resultante, a-a, se obtiene aplicando los conjuntos D y A sobre la DE b-a

  - » Se debe incluir un test objetivo
  - » El coste de aplicación de los operadores habitualmente es unitario
- Los *FSM* parten de una DE *before-action* y mediante un operador STRIPS generan una DE *after-action*. Este proceso lo realizan de forma iterativa (hacia delante), generando una secuencia de DEs, la última de las cuales se desea que sea el objetivo



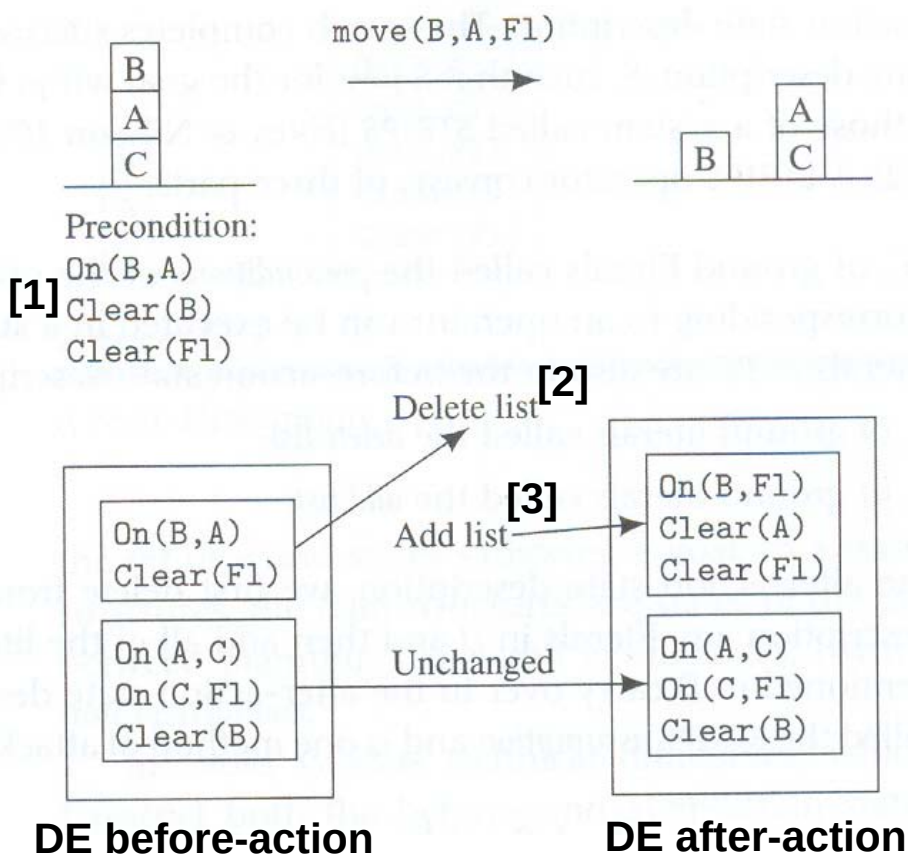
FSM, II

- Al no existir funciones, el espacio de estados es finito. Cualquier algoritmo completo para búsqueda en grafos lo es también para planificación (por ejemplo A*)
- En la práctica suelen ser ineficaces en problemas de tamaño medio. Por ejemplo, para un problema de planificación de carga aérea con
  - » 9 aeropuertos
  - » 50 aviones
  - » 200 piezas a transportar
  - » Objetivo: llevar la carga de aeropuerto A a aeropuerto B
  - » El factor de ramificación es de aproximadamente 1000 y el árbol de búsqueda tiene aproximadamente 1000^{41} nodos

FSM, III

- Ejemplo de aplicación del patrón de regla visto anteriormente en búsqueda hacia adelante:
  - » move(x,y,z)
    - – PC: On(x,y) ^ Clear(x) ^ Clear(z) [1]
    - – D: Clear(z), On(x,y) [2]
    - – A: On(x,z), Clear(y), Clear(Fl) [3]
  - » [1]: PC en una regla STRIPS están en forma de objetivo: conjunción de literales. Se asumen cuantificadas existencialmente las variables de PC.
    - – Una instancia de regla STRIPS (instancia de patrón) se puede aplicar a la DE Δ si una instancia de PC se satisface mediante la DE. Tal instancia se obtiene mediante la unificación, por etapas, de los literales en PC con un literal en la DE.
    - – La instancia del operador aplicable se obtiene aplicando la sustitución a todos los elementos de la regla.

FSM, IV ejemplo operador STRIPS (Nilsson, pp. 376)



$$PC = On(x,y) \wedge Clear(x) \wedge Clear(z)$$

$$DE_{b-a} = \begin{Bmatrix} On(B,A), Clear(B), Clear(Fl) \\ On(A,C), On(C,Fl) \end{Bmatrix}$$

$$A_1 = \{On(x,y), On(B,A)\}, D_1 = \{x, B\}, \sigma_1 = \{x := B\}$$

$$A_1\sigma_1 = \{On(B,y), On(B,A)\}$$

…

$$\sigma = \{x := B, y := A, z := Fl\}$$

FSM, V (aplicación cont.)

- La instancia de la regla y la DE *after-action* quedan, una vez aplicado el unificador σ:

$$move(B, A, Fl)$$

$$PC = On(B, A) \wedge Clear(B) \wedge Clear(Fl)$$

$$D = \{On(B, A), Clear(Fl)\}$$

$$A = \{On(B, Fl), Clear(A), Clear(Fl)\}$$

$$y$$

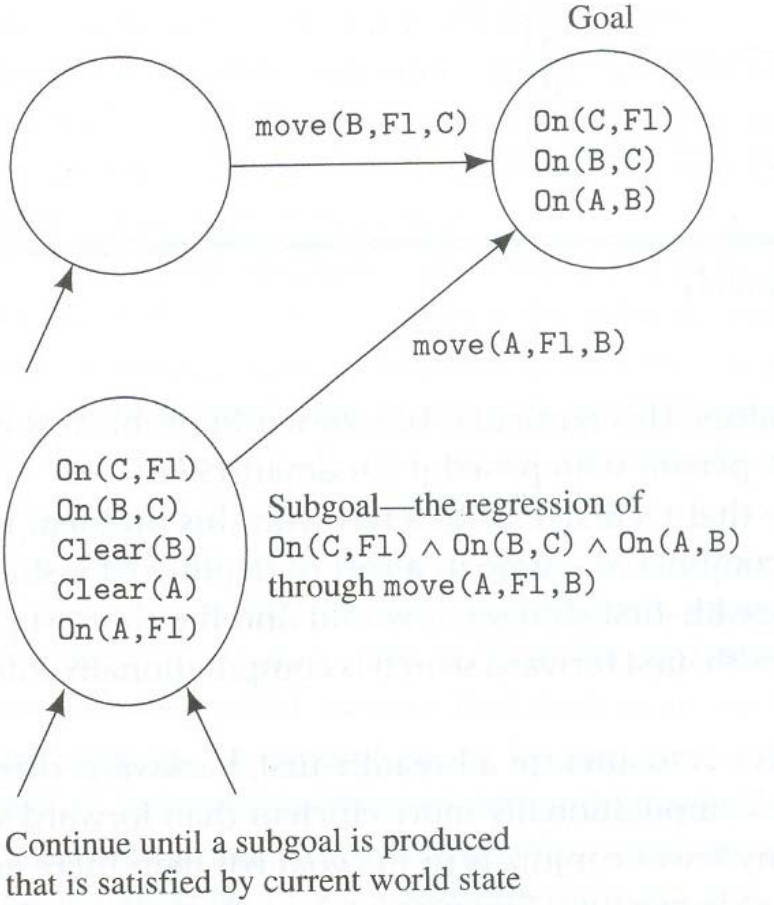
$$DE_{a-a} = \left\{ \begin{array}{l} On(A, C), On(B, Fl), On(C, Fl), \\ Clear(A), Clear(B), Clear(Fl) \end{array} \right\}$$

- En la búsqueda puede usarse, por ejemplo, A*, donde
  - » h es una heurística que estime el coste al objetivo
  - » g función que refleje el coste de las acciones (usualmente unitario)
- Sin una heurística sobre qué acciones aplicar, FSM no es apta en aplicaciones con gran número de reglas y DE

Backward Search Methods (*BSM*)

- También denominados *Regression Planning Methods*
- Poco eficientes en problemas grandes
- Consideran sólo operadores relevantes, aquellos que alcanzan alguno de los objetivos que se desea explorar
- La regresión de una fórmula w mediante una regla STRIPS *a* es una fórmula más débil w' tal que, si w' se satisface por una $DE_{b\text{-}a}$ antes de aplicar una instancia de *a* (w' verifica PC de la instancia de *a*) entonces w será satisfecha por la $DE_{a\text{-}a}$ después de aplicar la instancia de *a*

Ejemplo de BSM con STRIPS (Nilsson, pp. 382)



BSM, II

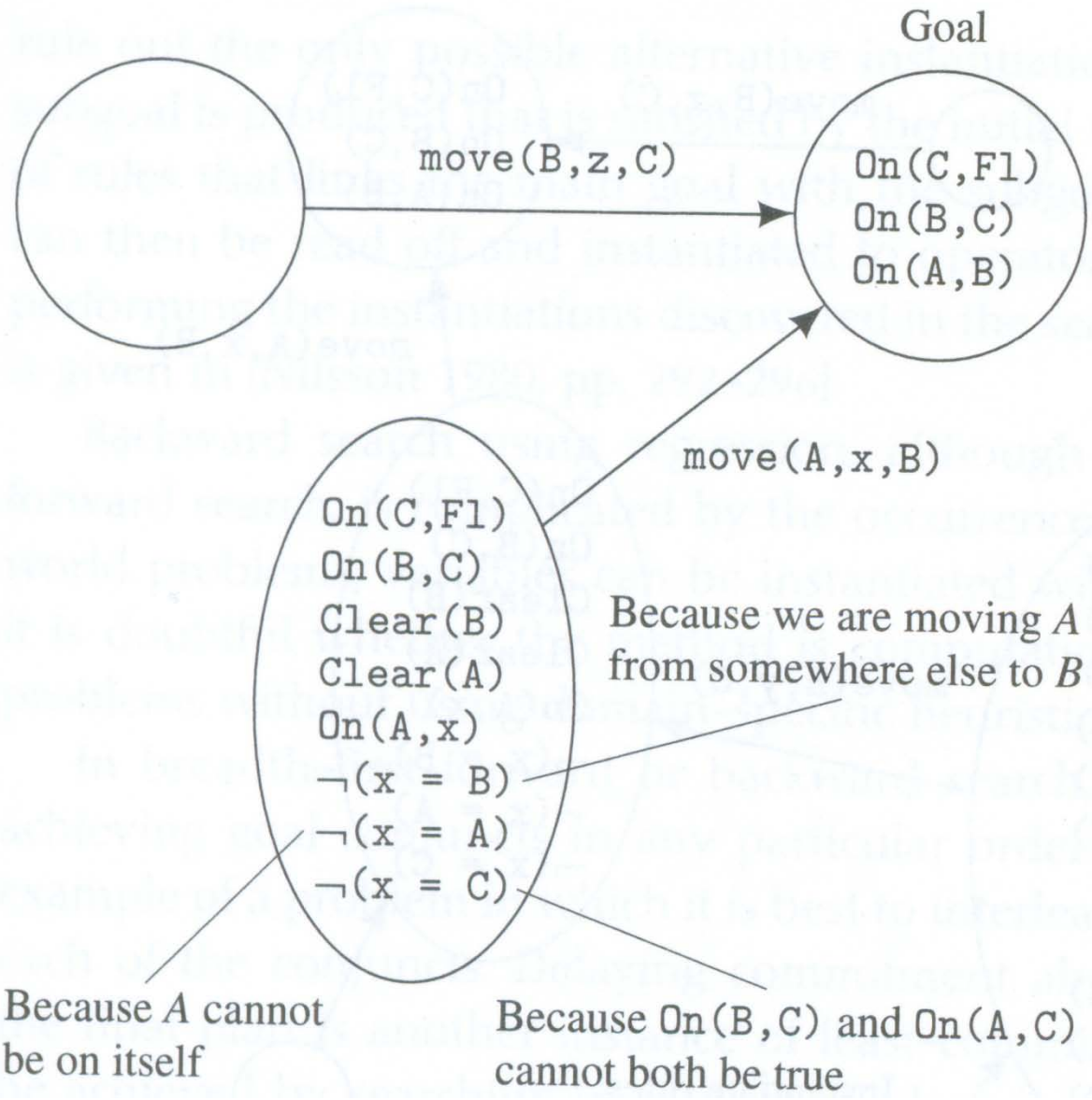
- Es importante que los operadores sean consistentes, es decir, que no deshagan literales (aquellos que aparecen en su conjunto D)
  - » No hay forma de que un operador alcance un literal λ si dicho literal lo borra el operador
  - » En tal caso, una regresión sobre cualquier conjunción de literales que contenga λ mediante un operador que elimine λ es F (dicho estado es imposible de alcanzar)
- El proceso de construcción de predecesores mediante regresión es:
  - » Sea G objetivo
  - » Sea C un operador relevante y consistente
  - » Se define el predecesor como
    - – i) cualquier efecto positivo de C que aparezca en G se elimina
    - – ii) cada precondición literal de C se añade, salvo que ya aparezca

BSM, III

- Uso de operadores no instanciados parcialmente
  - » Es complejo. Se describe con detalle en [Nilsson 1980, 288-].
  - » Permite retrasar la toma de decisiones sobre las instancias de determinados parámetros del operador, dejándolos sin especificar temporalmente. Esta alternativa en el uso de operadores se denomina *planificación menos comprometida* (*least commitment planning*)
    - – En el ejemplo de la página siguiente, el operador move(A,x,B) está instanciado parcialmente. Una de las precondiciones queda ahora instanciada en términos de una variable On(A,x)
    - – Interpretado como subobjetivo, este literal se puede satisfacer si unifica con un literal de una DE
    - – Una heurística posible consiste en retrasar la instanciación hasta conseguir la unificación con literales en la DE inicial

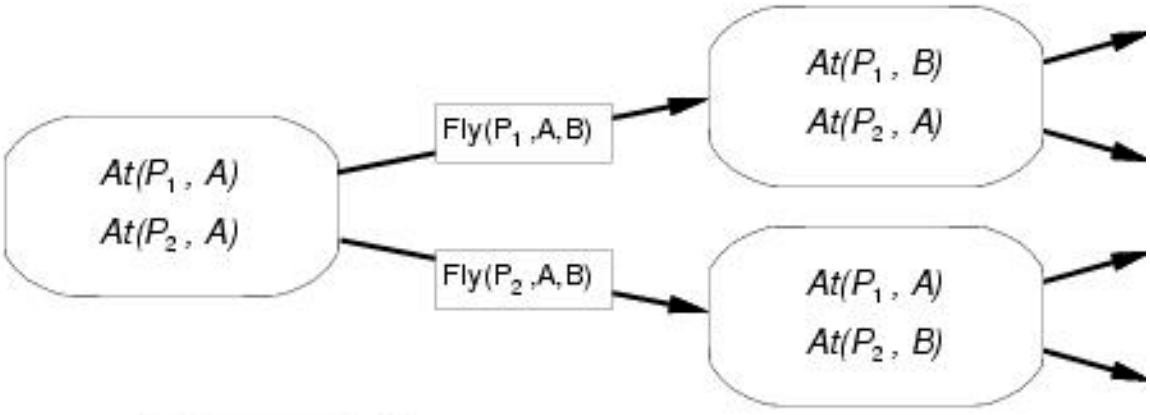
BSM, IV

- Ejemplo de *Least Commitment Planning* (Nilsson, pp. 383)

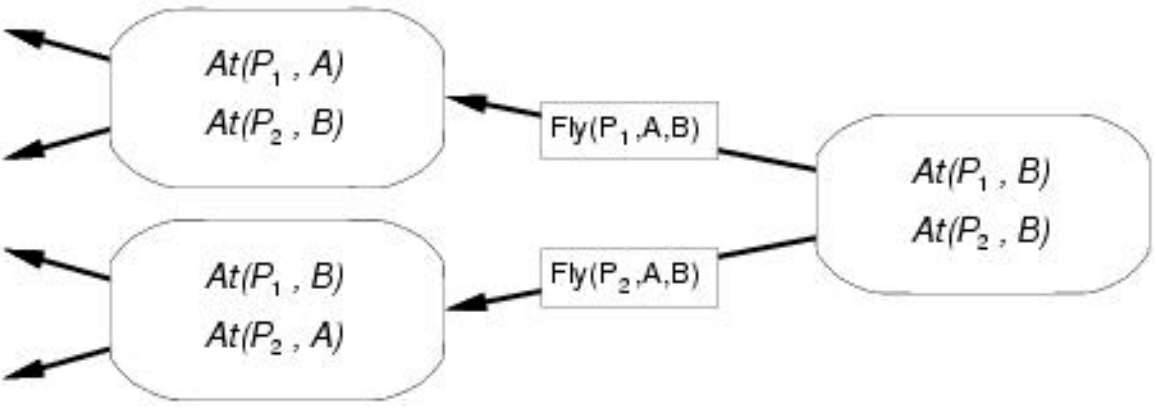


Forward vs Backward

(a)



(b)



STRIPS recursivo

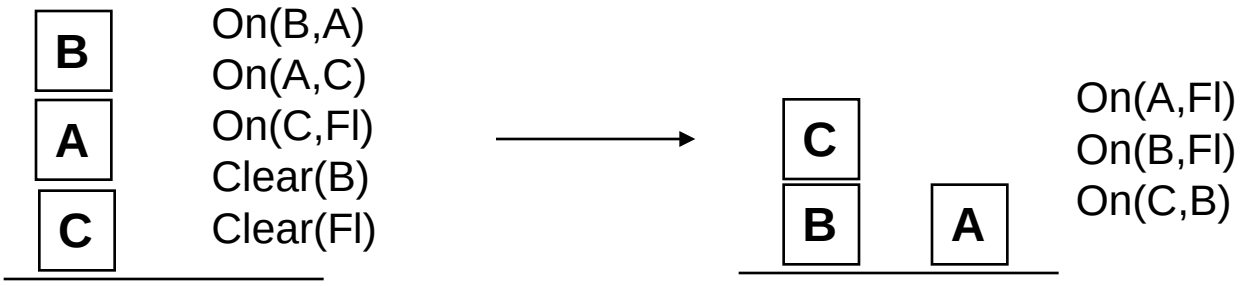
- Si el objetivo es una conjunción de literales puede emplearse en *FSM* una heurística *divide y vencerás*, satisfaciendo un literal cada vez
  - » Se aplican los operadores necesarios para satisfacer el literal propuesto. A continuación se repite el proceso hacia un nuevo literal y así sucesivamente
- El sistema GPS (*General Problem Solver*) aborda de este modo la resolución de problemas.
- La aplicación de esta técnica (STRIPS recursivo) a los problemas de planificación requiere
  - » Seleccionar un literal en el objetivo
  - » Seleccionar una instancia de un operador STRIPS que lo satisfaga
  - » Crear, como subobjetivo, la precondición necesaria para aplicar la regla
  - » Trabajar recursivamente sobre la DE generada hasta ir satisfaciendo el resto de literales del objetivo

STRIPS recursivo II

- Algoritmo STRIPS recursivo para satisfacer una fórmula objetivo w:
  - » STRIPS(w): El procedimiento usa una estructura de datos global δ que consiste en un conjunto de literales. Se inicializa con la DE inicial y se actualiza durante el proceso
  - » Repetir hasta δ ⊨ w
    1. El test terminal produce una sustitución σ, tal vez vacía, tal que algunos de los literales (tal vez ninguno) de wσ están en δ. Puede haber distintas sustituciones, por tanto el test constituye un punto de retorno (*backtracking point: bp*)
    2. *g* ← un elemento de wσ tal que ¬(δ ⊨ w). Si se escoge otro literal → (*bp*)
    3. *f* ← una regla STRIPS cuyo conjunto A contiene un literal λ que unifica con *g* con el umg *s*. Si se escoge otra regla → (*bp*)
    4. *f'* ← *fs*. Instancia de f con la sustitución *s*. *f'* puede contener variables
    5. *p* ← precondiciones de *f'* (instanciadas con la sustitución *s*)
    6. STRIPS(*p*). Llamada recursiva para producir una DE que verifique el subobjetivo. Esta llamada modifica δ
    7. *f''* ← Instancia de *f'* aplicable en δ
    8. δ ← resultado de aplicar *f''* a δ. δ siempre consiste en una conjunción de literales

STRIPS recursivo (ejemplo)

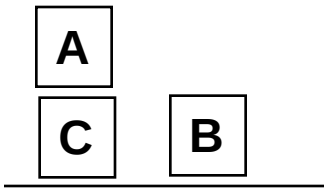
- Estado inicial objetivo



Ningún literal del objetivo está en la DE inicial
g ← On(A,fl)
La regla r1: move(A,x,Fl) tiene On(A,fl) en su lista A
llamar a STRIPSr para analizar las PC de la regla
PC(r1): Clear(A) ^ Clear(Fl) ^ On(A,x)
se genera la sustitucion σ_1={x:=C}
PC(r1)σ_1 están en DE inicial salvo Clear(A)
g ← Clear(A)
La regla r2: move(y,A,u) se escoge para alcanzar el literal
llamar a STRIPSr para analizar las PC de la regla
PC(r2): Clear(y) ^ Clear(u) ^ On(y,A)
se genera la sustitucion σ_2={y:=B, u:=Fl}
PC(r2)σ_2 están en DE inicial
Salimos de la segunda recursión para aplicar move(B,A,Fl)
El estado inicial cambia.

STRIPS recursivo (ejemplo II)

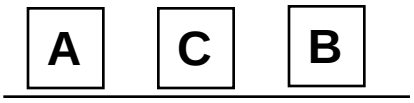
- Estado sucesor-1



On(B,Fl)
On(A,C)
On(C,Fl)
Clear(A)
Clear(B)
Clear(Fl)

Estamos en la llamada recursiva inicial
Comprobamos de nuevo PC(r1)

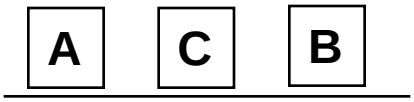
PC(r1)σ_1 : Clear(A) ^ Clear(Fl) ^ On(A,C)
Se cumplen.
Salimos de la 1ª recursión aplicando: move(A,C,Fl)
El estado sucesor-1 cambia.



On(B,Fl)
On(A,Fl)
On(C,Fl)
Clear(A)
Clear(B)
Clear(C)
Clear(Fl)

STRIPS recursivo (ejemplo III)

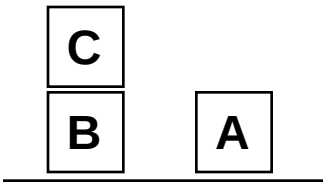
- Estado sucesor-2



On(B,Fl)
On(A,Fl)
On(C,Fl)
Clear(A)
Clear(B)
Clear(C)
Clear(Fl)

Comprobamos si se cumple $\delta \models w$. No se cumple aún.
Falta el literal On(C,B) del objetivo

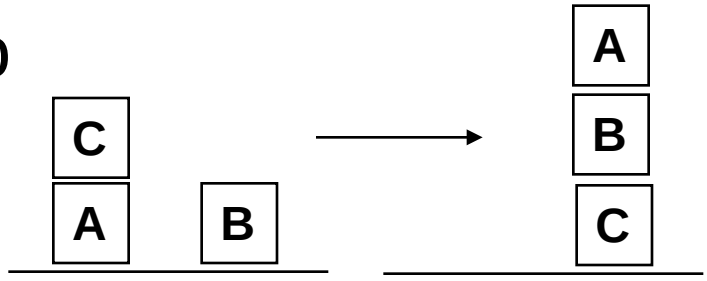
Comprobamos que existe r3: move(C,Fl,B) tal que
PC(r3): Clear(C) ^ Clear(B) ^ On(C,Fl)
Se cumplen.
aplicando move(C,Fl,B) se alzanza la DE objetivo



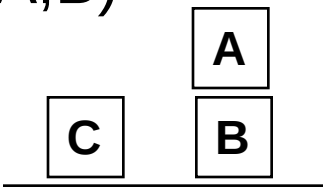
On(A,Fl)
On(B,Fl)
On(C,B)

La anomalía de Sussman (1975)

- Surge al deshacer operadores que han sido aplicados previamente mediante STRIPS recursivo



- Ejemplo
  - » Objetivo w: On(A,B)^On(B,C)
  - » Supongamos que se escoge On(A,B)
    - – Retiramos C de A, al tablero
    - – Entonces movemos A sobre B
    - – Para alcanzar el otro literal On(B,C)
      - ¡Deshacer el movimiento anterior On(A,B)!



  - » Supongamos que se escoge 1º On(B,C)
    - – Movemos B sobre C
    - – Para alcanzar el otro literal On(A,B)
      - ¡Deshacer el movimiento anterior On(B,C)!
  - » Conclusión: no hay un plan que minimice el número de operadores a emplear
  - » Alternativa: Usar *BSM*