Optimizando Dominios de Planning

Carlos Areces
carlos.areces@gmail.com
https://cs.famaf.unc.edu.ar/~careces/

(colaboración con F. Bustos, M. Dominguez, A. Torralba, J. Hoffmann)

XIV Jornadas de Ciencias de la Computación

Departamento de Ciencias de la Computación
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

19, 20 y 21 de Octubre de 2016 - Rosario, Santa Fe, Argentina

En esta Charla

- Introducir Planning Clásico
- Discutir principales características
- Dar ejemplos de uso
- Presentar el problema de instanciación
- Contarles sobre optimización de dominios

Haciendo Planes

- Hacer planes es parte de nuestra vida

Haciendo Planes

- Hacer planes es parte de nuestra vida
- Actuamos sin planificar
  - cuando el propósito propósito es inmediato
  - cuando realizamos tareas bien aprendidas aprendidas
  - cuando nuestro nuestro curso de acción puede adaptarse libremente sin demasiadas consecuencias

Haciendo Planes

- Hacer planes es parte de nuestra vida
- Actuamos sin planificar
  - cuando el propósito propósito es inmediato
  - cuando realizamos tareas bien aprendidas aprendidas
  - cuando nuestro nuestro curso de acción puede adaptarse libremente sin demasiadas consecuencias
- Planificamos
  - cuando nos encontramos ante situaciones nuevas
  - cuando las tareas son complejas
  - cuando existen existen riesgos (costos) que asumir
  - cuando colaboramos con otros

Haciendo Planes

- Hacer planes es parte de nuestra vida
- Actuamos sin planificar
  - cuando el propósito propósito es inmediato
  - cuando realizamos tareas bien aprendidas aprendidas
  - cuando nuestro nuestro curso de acción puede adaptarse libremente sin demasiadas consecuencias
- Planificamos
  - cuando nos encontramos ante situaciones nuevas
  - cuando las tareas son complejas
  - cuando existen existen riesgos (costos) que asumir
  - cuando colaboramos con otros
- Compromisos
  - Sólo planificamos algo cuando es estrictamente necesario, ya que elaborar planes requiere esfuerzo
  - A menudo nos bastan planes aceptables, aunque no sean óptimos (satisfacción vs. optimalidad)

Problema de Planificación

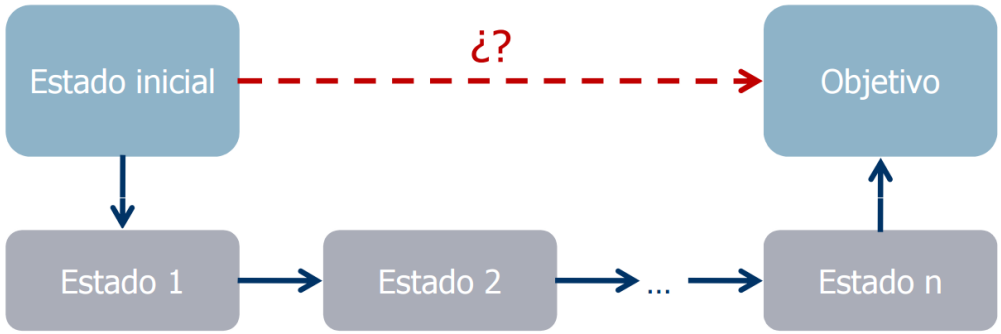
- Dados
  - una descripción del "mundo" (un modelo)
  - un estado inicial
  - una descripción de los objetivos y
  - un conjunto de acciones que pueden cambiar el mundo

Problema de Planificación

- Dados
  - una descripción del "mundo" (un modelo)
  - un estado inicial
  - una descripción de los objetivos y
  - un conjunto de acciones que pueden cambiar el mundo
- Encontrar
  - una secuencia de acciones que convierten el estado inicial en un estado que satisfaga los objetivos

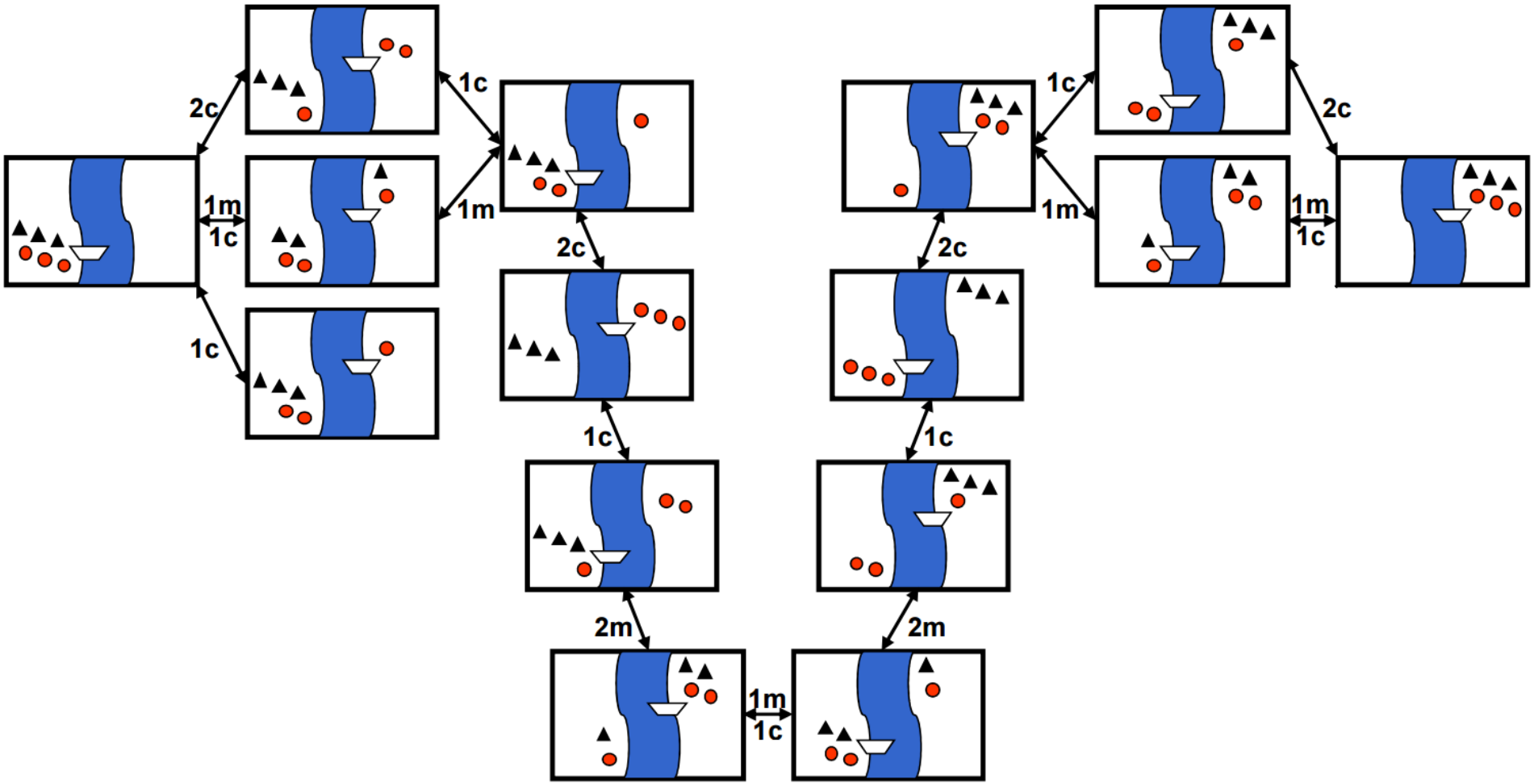
Problema de Planificación

- Dados
  - una descripción del "mundo" (un modelo)
  - un estado inicial
  - una descripción de los objetivos y
  - un conjunto de acciones que pueden cambiar el mundo
- Encontrar
  - una secuencia de acciones que convierten el estado inicial en un estado que satisfaga los objetivos



$\rightarrow$ representa la ejecución de una acción (aplicación de algún operador o transformación del sistema)

Ejemplo: Misioneros y Caníbales



Características de Planning Clásico

- Tiempo atómico: cada acción es indivisible
- Sin acciones concurrentes: se ejecuta una sóla acción
- Acciones determinísticas: el resultado de cada acción está totalmente determinado
- Único agente: Las acciones son las únicas que producen cambios
- Closed Word Assumption: toda propiedad no asertada como verdadera es asumida como falsa

Estas características son típicas de Planning Clásico

Usando Planes

Usando Planes

Lamentablemente

- El modelo del sistema difiere del sistema real.
- Pueden existir factores externos que interrumpan la ejecución del plan.

Usando Planes

Lamentablemente

- El modelo del sistema difiere del sistema real.
- Pueden existir factores externos que interrumpan la ejecución del plan.
- Planificación dinámica:
  Planificación y ejecución se entrelazan.
  - Supervisión de planes
    (detectar observaciones diferentes a los resultados esperados).
  - Revisión de planes
    (adaptación del plan existente a nuevas circunstancias).
  - Replanificación
    (generación de un nuevo plan a partir del estado actual).

Usando Planes

Lamentablemente

- El modelo del sistema difiere del sistema real.
- Pueden existir factores externos que interrumpan la ejecución del plan.
- Planificación dinámica:
  Planificación y ejecución se entrelazan.
  - Supervisión de planes
    (detectar observaciones diferentes a los resultados esperados).
  - Revisión de planes
    (adaptación del plan existente a nuevas circunstancias).
  - Replanificación
    (generación de un nuevo plan a partir del estado actual).

Reachability vs. Búsqueda

- Podríamos transformar planning en un problema de búsqueda de camino en grafos
  - Enumeramos todos los estados posibles
  - Conectamos estados via las acciones legales ($I \rightarrow I'$)
  - Buscar un camino que una el estado inicial con el un estado que satisfaga la meta (reachability en el grafo de estados)

Reachability vs. Búsqueda

- Podríamos transformar planning en un problema de búsqueda de camino en grafos
  - Enumeramos todos los estados posibles
  - Conectamos estados via las acciones legales ($I \rightarrow I'$)
  - Buscar un camino que una el estado inicial con el un estado que satisfaga la meta (reachability en el grafo de estados)
- Ineficiente cuando el número de posibles estados es alto

Reachability vs. Búsqueda

- Podríamos transformar planning en un problema de búsqueda de camino en grafos
  - Enumeramos todos los estados posibles
  - Conectamos estados via las acciones legales ($I \rightarrow I'$)
  - Buscar un camino que una el estado inicial con el un estado que satisfaga la meta (reachability en el grafo de estados)
- Ineficiente cuando el número de posibles estados es alto
- Generar dinámicamente el espacio de estados (algoritmos de búsqueda)

Algoritmos de Búsqueda

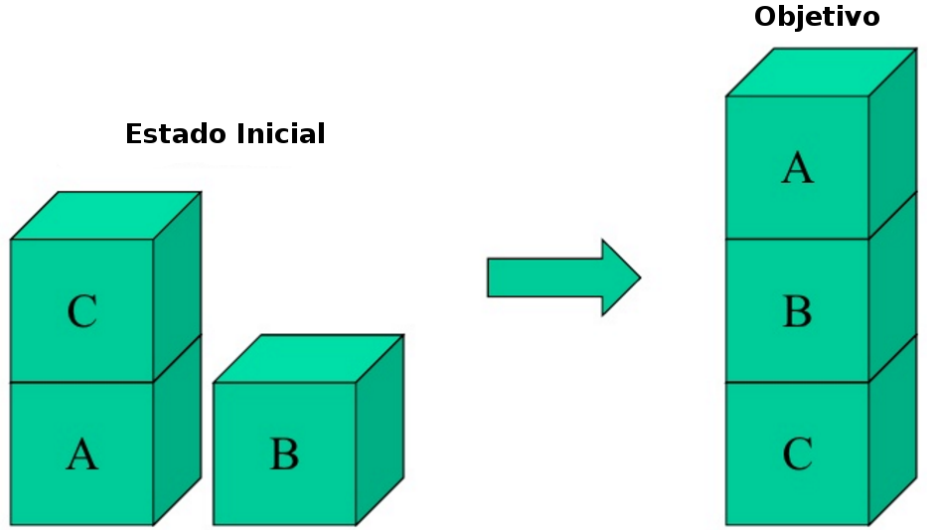
- Búsqueda con Backtracking
  1. DFS
  2. BFS / Dijkstra's Algorithm
  3. Iterative Deepening
  4. Best-first search
  5. A*
- Constraint Propagation
  1. Forward Checking
  2. DPLL & Resolution
  3. k-Consistency
- Búsqueda Local
  1. Hillclimbing
  2. Simulated annealing
  3. Walksat

En problemas difíciles es crucial la definición de heurísticas que guíen la búsqueda.

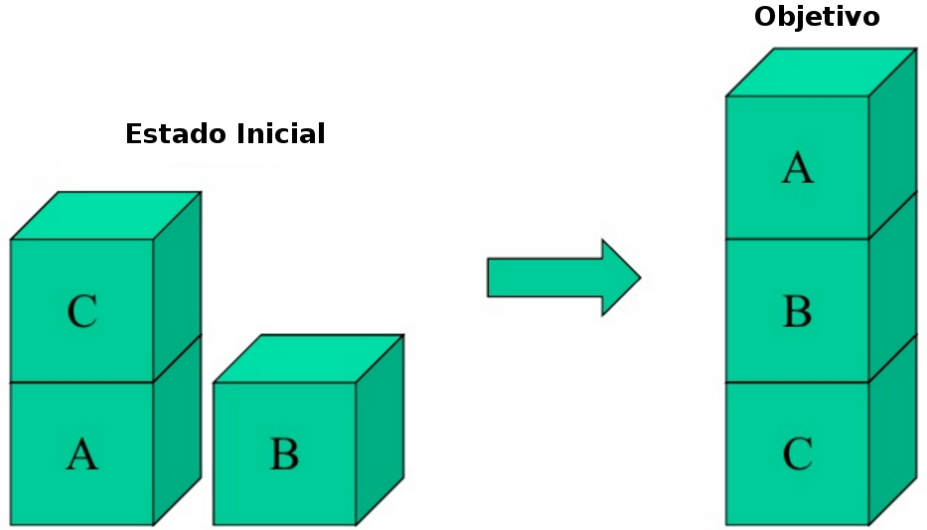
Planning Systems Actuales

- Planning as A* Search
  - HSP (Geffner & Bonet 1999), introducen heuristica admisible "ignore negative effects".
  - FF (Hoffman & Nebel 2000), usan una modificación de la anterior que es no-admissible
    - De cómputo polinomial, y con excelentes resultados. Usualmente lleva a soluciónes que no son óptimas (no se obtiene el plan más corto → satisficing planning).
    - Mejor performance en la IPC 2000 planning competition
- Hoy en día: LAMA, FD, BFS(f) . . .
  En la IPC 2014 compitieron 67 planners.

Formalizando Planes

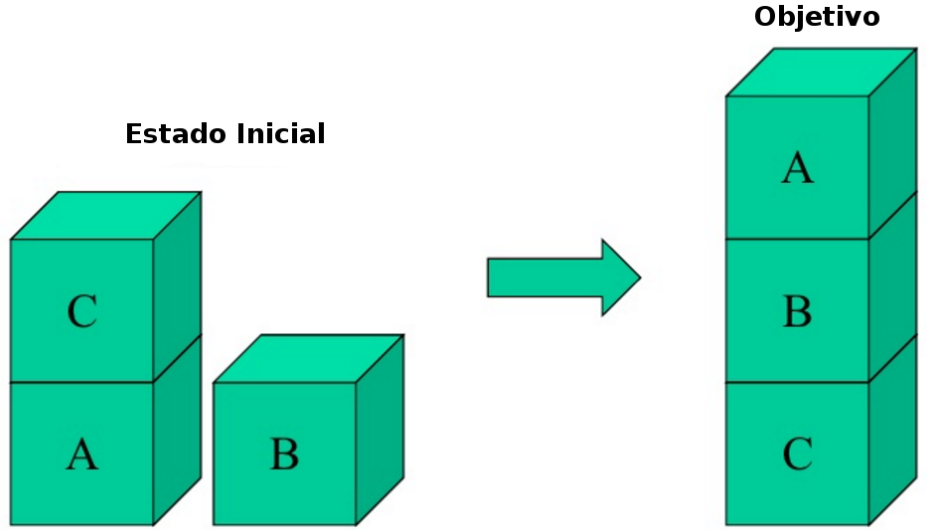


Formalizando Planes



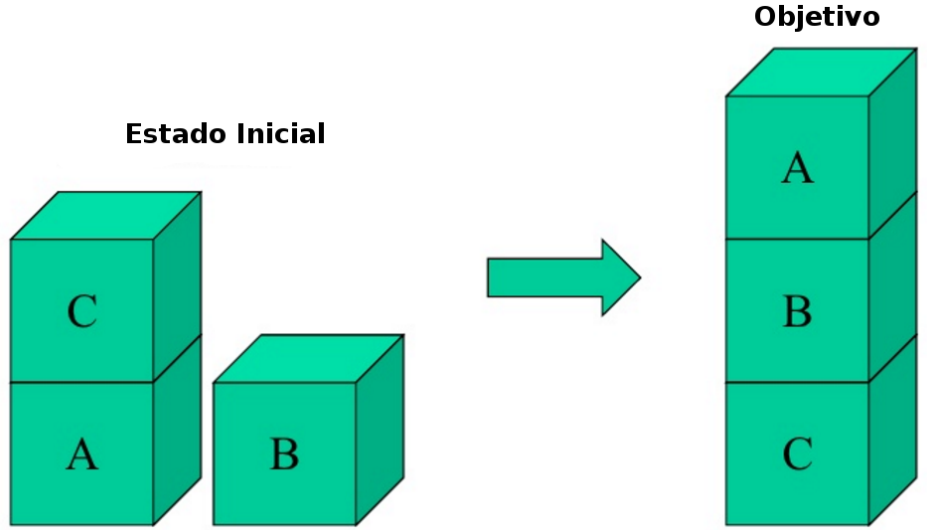
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)

Formalizando Planes



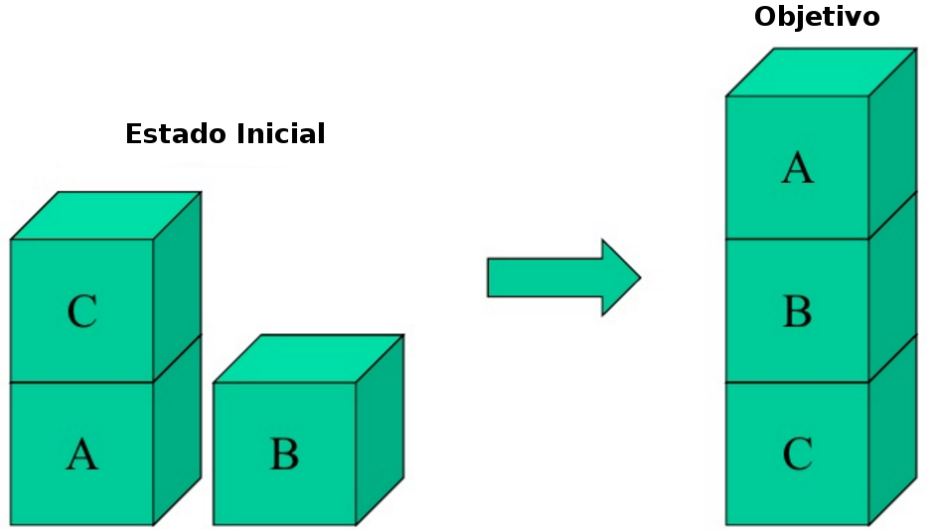
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)

Formalizando Planes



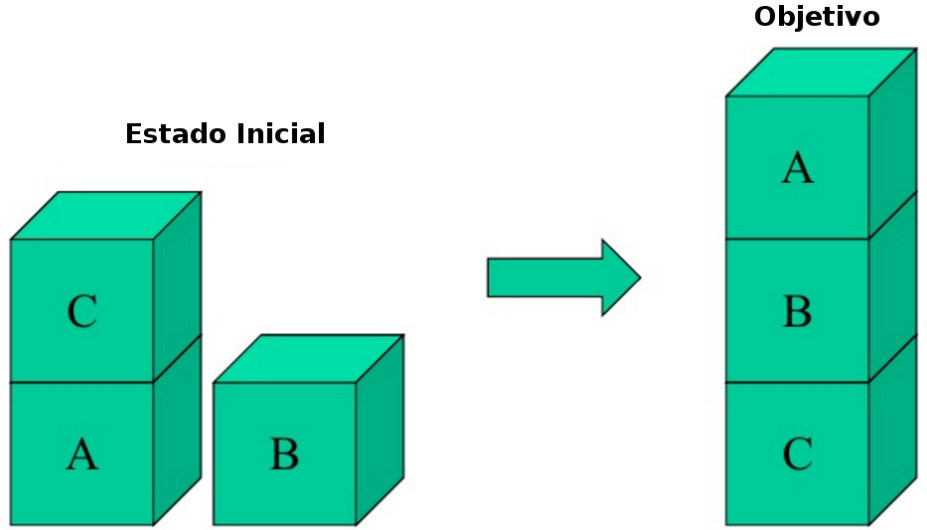
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE:

Formalizando Planes



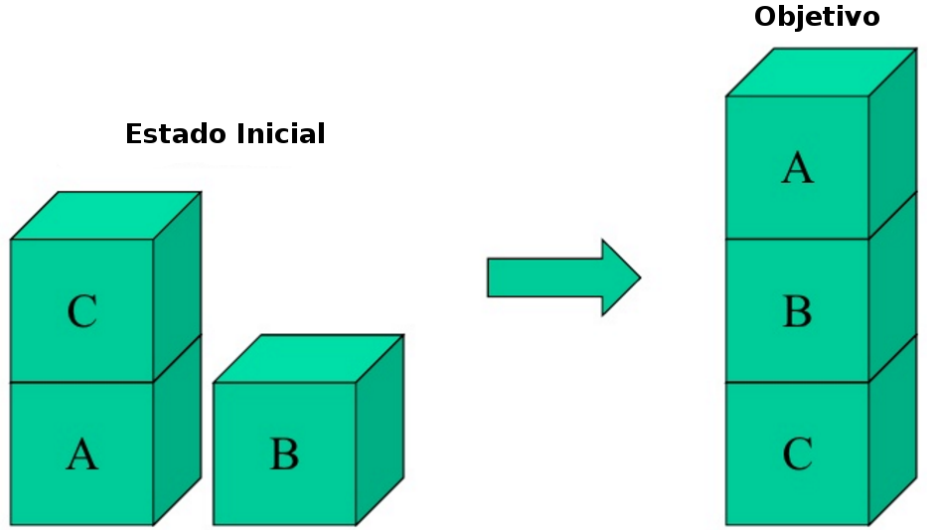
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE: $\{on(x, y), clear(x)\}$

Formalizando Planes



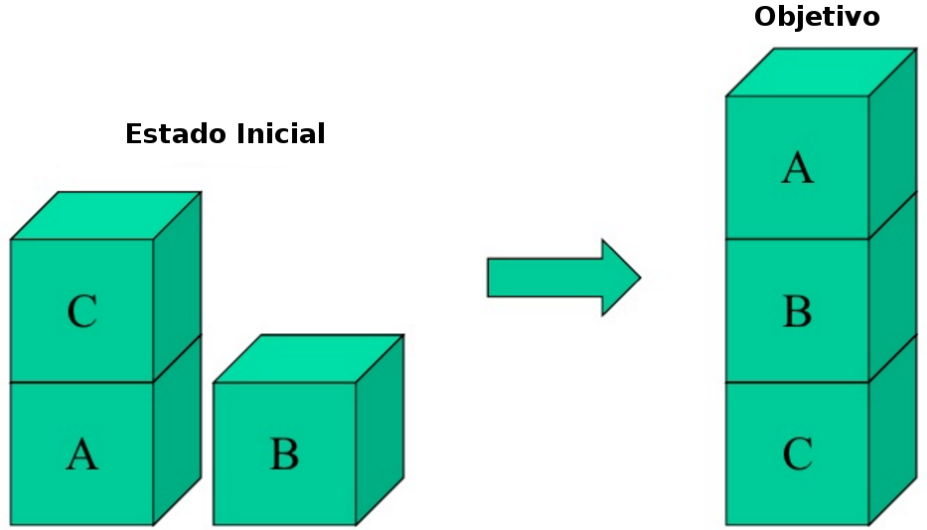
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE: $\{on(x, y), clear(x)\}$
    DEL:

Formalizando Planes



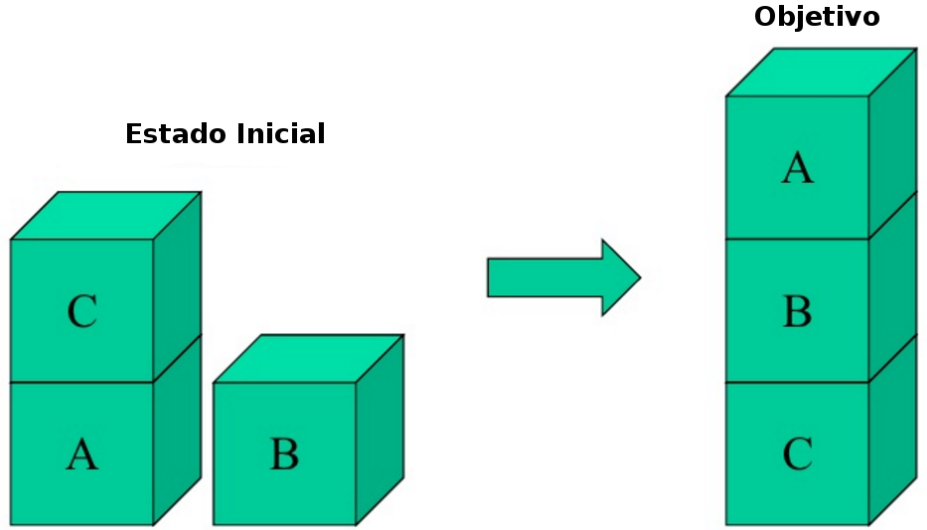
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE: $\{on(x, y), clear(x)\}$
    DEL: $\{on(x, y)\}$

Formalizando Planes



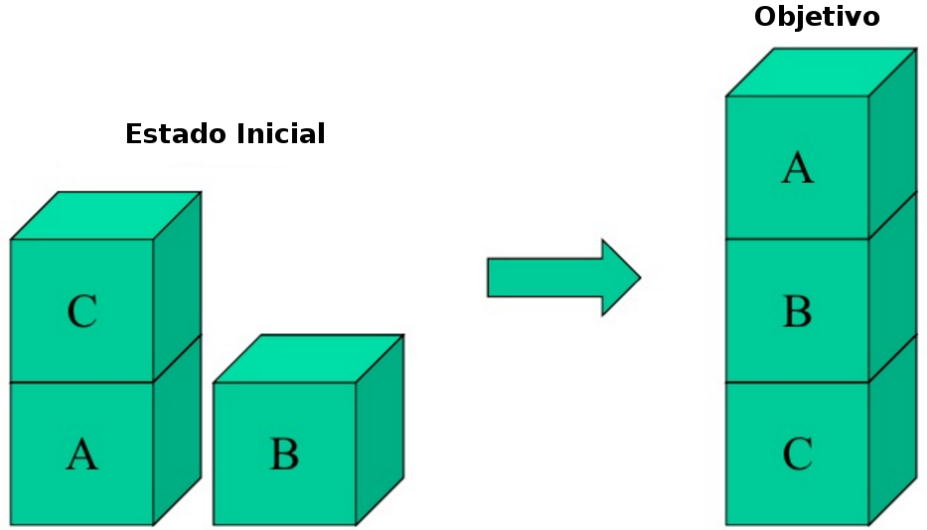
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE: $\{on(x, y), clear(x)\}$
    DEL: $\{on(x, y)\}$
    ADD:

Formalizando Planes



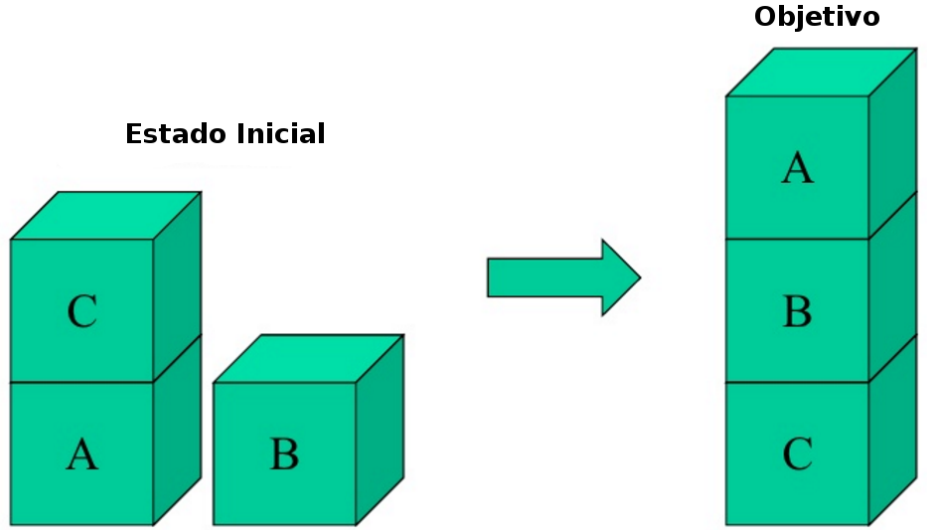
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE: $\{on(x, y), clear(x)\}$
    DEL: $\{on(x, y)\}$
    ADD: $\{on(x, Table), clear(y)\}$

Formalizando Planes



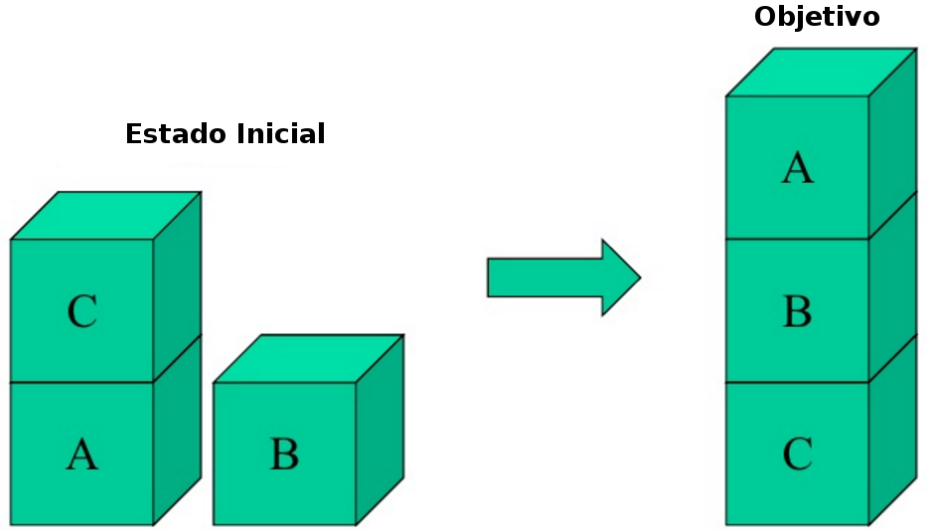
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE: $\{on(x, y), clear(x)\}$
    DEL: $\{on(x, y)\}$
    ADD: $\{on(x, Table), clear(y)\}$
  - **Stack(x,y)**

Formalizando Planes



- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE: $\{on(x, y), clear(x)\}$
    DEL: $\{on(x, y)\}$
    ADD: $\{on(x, Table), clear(y)\}$
  - **Stack(x,y)** Tarea!

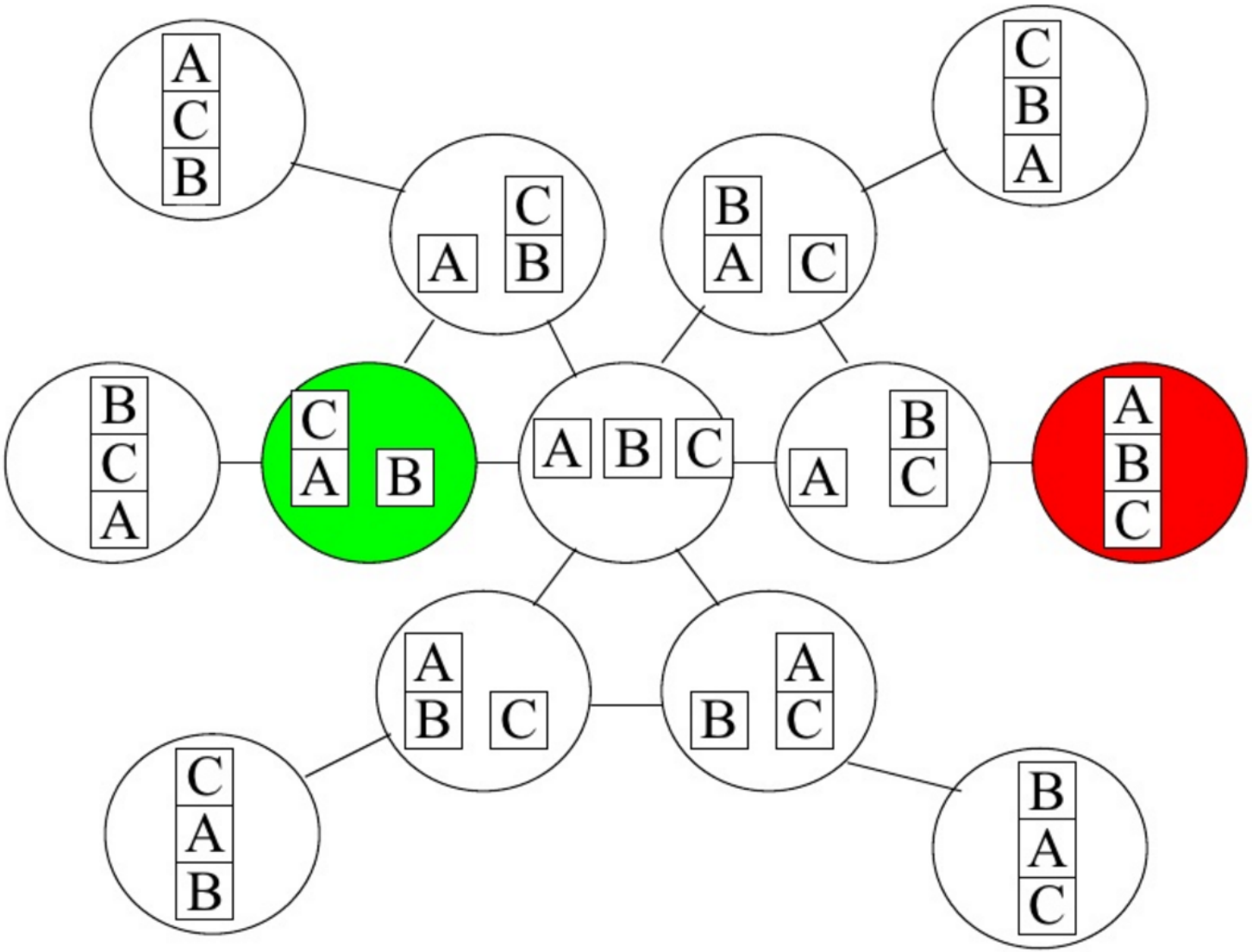
Formalizando Planes



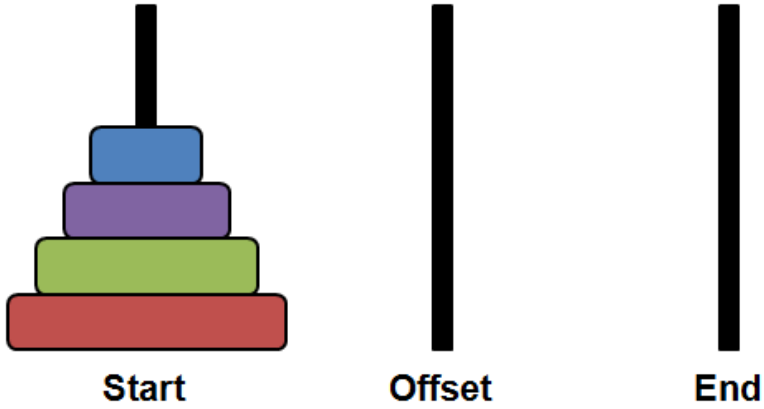
- **Estado Inicial:**
  on(A,Table) on(C,A) on(B,Table) clear(B) clear(C)
- **Objetivo:**
  on(C,Table) on(B,C) on(A,B) clear(A)
- **Operadores:**
  - **UnStack**(x,y)
    PRE: $\{on(x, y), clear(x)\}$
    DEL: $\{on(x, y)\}$
    ADD: $\{on(x, Table), clear(y)\}$
  - **Stack(x,y)** Tarea!

Notar que definimos un acción esquema.

Espacio de Búsqueda



Torres de Hanoi - Dominio




```
;; The Towers of Hanoi problem
;; Formalisation by Hector Geffner

(define (domain hanoi)
  (:requirements :strips)
  (:predicates (clear ?x) (on ?x ?y) (smaller ?x ?y))

  (:action move
    :parameters (?disc ?from ?to)
    :precondition (and (smaller ?to ?disc)
                       (on ?disc ?from)
                       (clear ?disc)
                       (clear ?to))
    :effect (and (clear ?from)
                 (on ?disc ?to)
                 (not (on ?disc ?from))
                 (not (clear ?to))))
  )
```

Torres de Hanoi - Problema y Solución

Problema:

```
(define (problem hanoi3)
  (:domain hanoi)
  (:objects peg1 peg2 peg3 d1 d2 d3)
  (:init
   (smaller peg1 d1) (smaller peg1 d2) (smaller peg1 d3)
   (smaller peg2 d1) (smaller peg2 d2) (smaller peg2 d3)
   (smaller peg3 d1) (smaller peg3 d2) (smaller peg3 d3)
   (smaller d2 d1) (smaller d3 d1) (smaller d3 d2)
   (clear peg2) (clear peg3) (clear d1)
   (on d3 peg1) (on d2 d3) (on d1 d2))
  (:goal (and (on d3 peg3) (on d2 d3) (on d1 d2)))  )
```

Torres de Hanoi - Problema y Solución

Problema:

```
(define (problem hanoi3)
  (:domain hanoi)
  (:objects peg1 peg2 peg3 d1 d2 d3)
  (:init
   (smaller peg1 d1) (smaller peg1 d2) (smaller peg1 d3)
   (smaller peg2 d1) (smaller peg2 d2) (smaller peg2 d3)
   (smaller peg3 d1) (smaller peg3 d2) (smaller peg3 d3)
   (smaller d2 d1) (smaller d3 d1) (smaller d3 d2)
   (clear peg2) (clear peg3) (clear d1)
   (on d3 peg1) (on d2 d3) (on d1 d2))
  (:goal (and (on d3 peg3) (on d2 d3) (on d1 d2)))  )
```

Plan:

```
step    0: move d1 d2 peg3
        1: move d2 d3 peg2
        2: move d1 peg3 d2
        3: move d3 peg1 peg3
        4: move d1 d2 peg1
        5: move d2 peg2 d3
        6: move d1 peg1 d2
```

Torres de Hanoi

El Granjero, el Zorro, la Gallina y el Maiz

areces@fire: ~/Repo/Teaching/Medina/Tests/ADL/crossing

```
areces@fire:~/Repo/Teaching/Medina/Tests/ADL/crossing$ ff -o domain.pddl -f problem.pddl

ff: parsing domain file
domain 'CROSSING' defined
 ... done.
ff: parsing problem file
problem 'CROSSING' defined
 ... done.



Cueing down from goal distance:    7 into depth [1]
                                   6            [1][2]

Enforced Hill-climbing failed !
switching to Best-first Search now.

advancing to distance :    7
                           6
                           5
                           4
                           3
                           1
                           0

ff: found legal plan as follows

step    0: BOARD_1 FARMER NEAR
        1: BOARD_2 CHICKEN NEAR
        2: CROSS_2 FARMER CHICKEN NEAR FAR
        3: BOARD_1 FARMER FAR
        4: CROSS_1 FARMER FAR NEAR
        5: BOARD_1 FARMER NEAR
        6: BOARD_2 CORN NEAR
        7: CROSS_2 FARMER CORN NEAR FAR
        8: BOARD_1 FARMER FAR
        9: BOARD_2 CHICKEN FAR
       10: CROSS_2 FARMER CHICKEN FAR NEAR
       11: BOARD_1 FARMER NEAR
       12: BOARD_2 FOX NEAR
       13: CROSS_2 FARMER FOX NEAR FAR
       14: BOARD_1 FARMER FAR
       15: CROSS_1 FARMER FAR NEAR
       16: BOARD_1 FARMER NEAR
       17: BOARD_2 CHICKEN NEAR
       18: CROSS_2 FARMER CHICKEN NEAR FAR


time spent:    0.00 seconds instantiating 12 easy, 384 hard action templates
               0.00 seconds reachability analysis, yielding 37 facts and 264 actions
               0.00 seconds creating final representation with 31 relevant facts
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 42 states, to a max depth of 2
               0.00 seconds total time

areces@fire:~/Repo/Teaching/Medina/Tests/ADL/crossing$
```

Aplicaciones de Planning

- Robótica (robots móviles y vehículos autónomos)
- Simulación (entrenamiento y juegos)
- Logística
- "Workflows" (fábricas y cadenas de montaje)
- Gestión de crisis (evacuaciones, incendios, . . .)

Aplicaciones de Planning

- Robótica (robots móviles y vehículos autónomos)
- Simulación (entrenamiento y juegos)
- Logística
- "Workflows" (fábricas y cadenas de montaje)
- Gestión de crisis (evacuaciones, incendios, . . .)

Ventajas:

Generalidad

Una especificación resuelve muchos problemas

Aplicaciones de Planning

- Robótica (robots móviles y vehículos autónomos)
- Simulación (entrenamiento y juegos)
- Logística
- "Workflows" (fábricas y cadenas de montaje)
- Gestión de crisis (evacuaciones, incendios, . . .)

Ventajas:

Generalidad

Una especificación resuelve muchos problemas

Desventajas:

Complejidad

Satisficing planning es PSPACE-complete

El Problema de la Instanciación

```
(:action move
  :parameters (?disc ?from ?to)
  :precondition
    (and (smaller ?to ?disc) (on ?disc ?from)
         (clear ?disc) (clear ?to))
  :effect
    (and (clear ?from) (on ?disc ?to)
         (not (on ?disc ?from)) (not (clear ?to))))
```

Para el problema de 51 discos se generarían $52^3 = 140608$ acciones instanciadas, que sería el factor de branching del árbol de búsqueda.

El Problema de la Instanciación

```
(:action move
  :parameters (?disc ?from ?to)
  :precondition
    (and (smaller ?to ?disc) (on ?disc ?from)
         (clear ?disc) (clear ?to))
  :effect
    (and (clear ?from) (on ?disc ?to)
         (not (on ?disc ?from)) (not (clear ?to))))
```

Para el problema de 51 discos se generarían $52^3 = 140608$ acciones instanciadas, que sería el factor de branching del árbol de búsqueda. Un árbol de búsqueda de branching k de profundidad n tiene

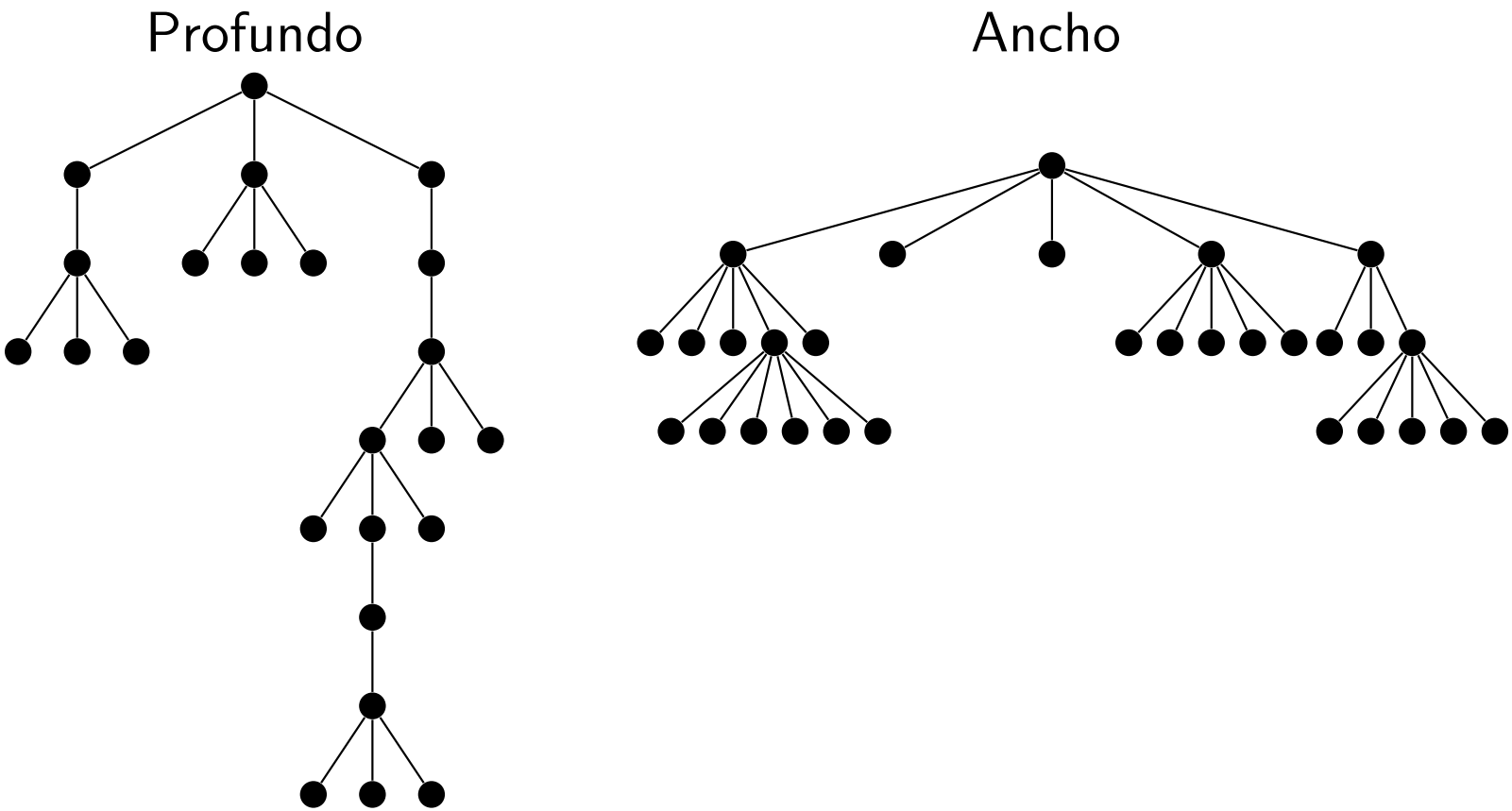
$$\frac{k(k^n - 1)}{k - 1} \text{ nodos}$$

Para $k = 140608$ y $n = 3$ tenemos 2.779.925.654.385.984 estados.

Action Split

- Definimos una transformación automática que reemplaza acciones esquema de interface grande (muchos parámetros) por varias acciones esquema de interfaz menor, reduciendo exponencialmente l número de acciones instanciadas.
- Mostramos que la transformación retorna un dominio equivalente al original.
- Esta transformación puede mejorar substancialmente la performance de un planer en dominios con interfaces grandes.

Intuición Cambiando Ancho por Profundo



Objetivos del Split

- Dado un esquema de acción $a[X]$, la operación de split crea varios esquemas $a_1[X_1], \ldots, a_k[X_k]$, de interfaces más pequeñas, cuya combinación corresponde exactamente a $a[X]$ en todo plan válido

Objetivos del Split

- Dado un esquema de acción $a[X]$, la operación de split crea varios esquemas $a_1[X_1], \ldots, a_k[X_k]$, de interfaces más pequeñas, cuya combinación corresponde exactamente a $a[X]$ en todo plan válido
- De esta forma se consigue reducir en número de acciones instanciadas. Por ejemplo, si cada parámetro $x \in X$ puede instanciarse con 100 objetos, $|X| = 3$, and $|X_i| = 1$, el número de acciones instanciadas se reduce de 1,000,000 a 300.

Objetivos del Split

- Dado un esquema de acción $a[X]$, la operación de split crea varios esquemas $a_1[X_1], \ldots, a_k[X_k]$, de interfaces más pequeñas, cuya combinación corresponde exactamente a $a[X]$ en todo plan válido
- De esta forma se consigue reducir en número de acciones instanciadas. Por ejemplo, si cada parámetro $x \in X$ puede instanciarse con 100 objetos, $|X| = 3$, and $|X_i| = 1$, el número de acciones instanciadas se reduce de 1,000,000 a 300.
- Elegir $a_1[X_1], \ldots, a_k[X_k]$ implica un trade-off entre minimizar el tamaño de la interface $\max_i |X_i|$ (y, por consecuencia, el número de acciones instanciadas), vs. minimizar el *tamaño de split* k y, por consecuencia, el largo de plan.

Splitting de Acciones Esquema

- La transformación asegura que los planes del dominio de planes transformado están en una correspondencia uno a uno con los del dominio original.

Ejemplo: Consideremos la acción esquema para mover un bloque x que está sobre y para ponerlo sobre z:

Move(x, y, z)
$PRE : \{on(x, y), clear(x), clear(z)\}$
$DEL : \{on(x, y), clear(z)\}$
$ADD : \{on(x, z), clear(y)\}$

Splitting de Acciones Esquema

- La transformación asegura que los planes del dominio de planes transformado están en una correspondencia uno a uno con los del dominio original.

Ejemplo: Consideremos la acción esquema para mover un bloque x que está sobre y para ponerlo sobre z:

$$\begin{aligned}&\textbf{Move}(x, y, z)\\ &\quad PRE : \{on(x, y), clear(x), clear(z)\}\\ &\quad DEL : \{on(x, y), clear(z)\}\\ &\quad ADD : \{on(x, z), clear(y)\}\end{aligned}$$

Podríamos definir el siguiente split

$$\begin{aligned}&\textbf{Move}_1(x, y)\\ &\quad PRE : \{on(x, y), clear(x)\}\\ &\quad DEL : \{on(x, y)\}\\ &\quad ADD : \{clear(y)\}\end{aligned}$$

$$\begin{aligned}&\textbf{Move}_2(x, z)\\ &\quad PRE : \{clear(z)\}\\ &\quad DEL : \{clear(z)\}\\ &\quad ADD : \{on(x, z)\}\end{aligned}$$

La correspondencia con el original parece obvia, pero ...

Action Schema Splitting

Move(x, y, z)	**Move**$_1(x, y)$	**Move**$_2(x, z)$
$PRE : \{on(x, y), clear(x), clear(z)\}$	$PRE : \{on(x, y), clear(x)\}$	$PRE : \{clear(z)\}$
$DEL : \{on(x, y), clear(z)\}$	$DEL : \{on(x, y)\}$	$DEL : \{clear(z)\}$
$ADD : \{on(x, z), clear(y)\}$	$ADD : \{clear(y)\}$	$ADD : \{on(x, z)\}$

no es válida, porque:

(1) Nada asegura que los dos sub-esquemas estan instanciados consistentemente, i. e., que el mismo objeto se asigna al parámetro compartido x en ambas partes

Action Schema Splitting

Move(x, y, z)	**Move**$_1(x, y)$	**Move**$_2(x, z)$
$PRE : \{on(x, y), clear(x), clear(z)\}$	$PRE : \{on(x, y), clear(x)\}$	$PRE : \{clear(z)\}$
$DEL : \{on(x, y), clear(z)\}$	$DEL : \{on(x, y)\}$	$DEL : \{clear(z)\}$
$ADD : \{on(x, z), clear(y)\}$	$ADD : \{clear(y)\}$	$ADD : \{on(x, z)\}$

no es válida, porque:

(1) Nada asegura que los dos sub-esquemas estan instanciados consistentemente, i. e., que el mismo objeto se asigna al parámetro compartido x en ambas partes

(2) Nada asegura que los dos sub-esquemas son ejecutados en bloque, i. e., seguidos y sin ninguna otra acción en el medio

Action Schema Splitting

Move(x, y, z)	**Move**$_1(x, y)$	**Move**$_2(x, z)$
$PRE : \{on(x, y), clear(x), clear(z)\}$	$PRE : \{on(x, y), clear(x)\}$	$PRE : \{clear(z)\}$
$DEL : \{on(x, y), clear(z)\}$	$DEL : \{on(x, y)\}$	$DEL : \{clear(z)\}$
$ADD : \{on(x, z), clear(y)\}$	$ADD : \{clear(y)\}$	$ADD : \{on(x, z)\}$

no es válida, porque:

(1) Nada asegura que los dos sub-esquemas estan instanciados consistentemente, i. e., que el mismo objeto se asigna al parámetro compartido x en ambas partes

(2) Nada asegura que los dos sub-esquemas son ejecutados en bloque, i. e., seguidos y sin ninguna otra acción en el medio

(3) Nada asegura el orden esperado entre precondiciones, adds y deletes.

- En nuestro ejemplo, si el add $clear(y)$ y la precondición $clear(z)$ son instanciadas con el mismo parámetro, entonces el esquema original no es aplicable ya que no podemos tener $on(x, y)$ y $clear(y)$ al mismo tiempo.
- En el esquema spliteado, el add de **Move**$_1(x, y)$ hace que **Move**$_2(x, z)$ sea ejecutable.

Decorando Splits

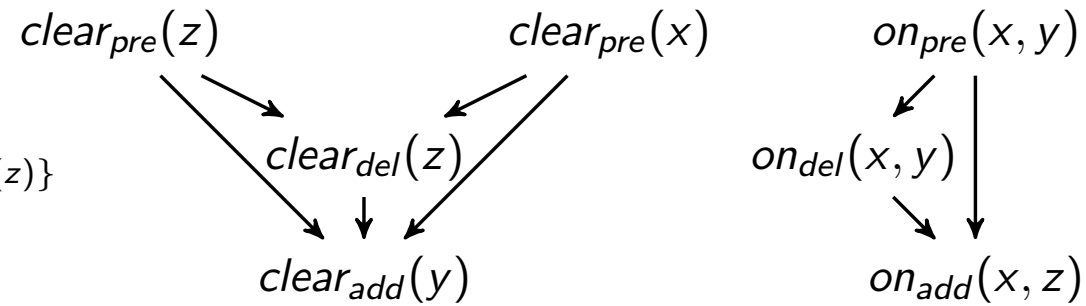
- Los problemas (1) y (2) pueden solucionarse fácilmente, para splits arbitrarios, decorando los sub-esquemas con nuevos átomos que aseguran la instanciación consistente de parámetros y la ejecución "en bloque"
  - para (1) implementar un sistema explícito de "pasaje de variables"
  - para (2) implementar un sistema de "tokens"
  - (un ejemplo más adelante)
- El problema (3) es más sutil y más complejo.

Grafos Cocientes y Splits Válidos

- El problema (3) es sutil, y es el único que, actualmente, restringe el conjunto de splits válidos.

Para **Move**(x, y, z), el grafo de átomos anotados parcialmente ordenados es el siguiente:

Move(x, y, z)
$PRE : \{on(x, y), clear(x), clear(z)\}$
$DEL : \{on(x, y), clear(z)\}$
$ADD : \{on(x, z), clear(y)\}$

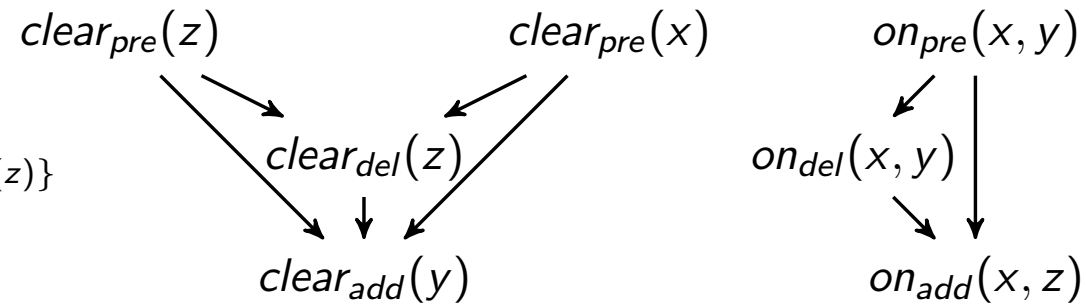


Grafos Cocientes y Splits Válidos

- El problema (3) es sutil, y es el único que, actualmente, restringe el conjunto de splits válidos.

Para **Move**(x, y, z), el grafo de átomos anotados parcialmente ordenados es el siguiente:

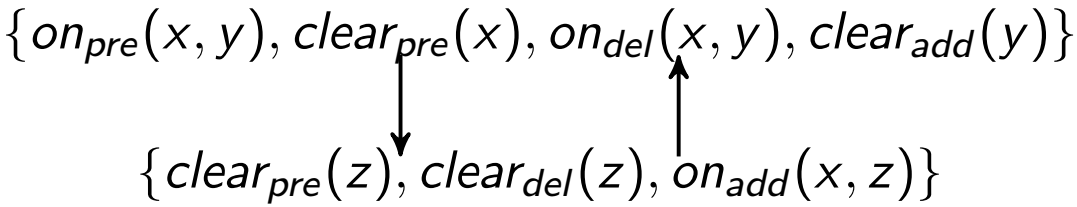
Move(x, y, z)
$PRE : \{on(x, y), clear(x), clear(z)\}$
$DEL : \{on(x, y), clear(z)\}$
$ADD : \{on(x, z), clear(y)\}$



El split $\mathbf{Move_1 Move_2}$ no preserva el orden sobre $clear(\cdot)$ ya que el delete $clear(z)$ aparece entre la precondition $clear(x)$ y el add $clear(y)$ en la acción original, pero aparece después de ambos en el split

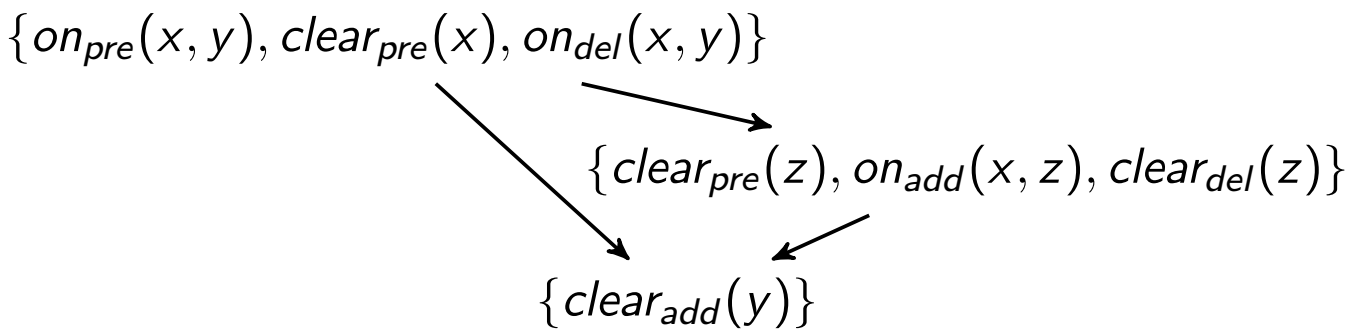
$\mathbf{Move}_1(x, y)$
$PRE : \{on(x, y), clear(x)\}$
$DEL : \{on(x, y)\}$
$ADD : \{clear(y)\}$

$\mathbf{Move}_2(x, z)$
$PRE : \{clear(z)\}$
$DEL : \{clear(z)\}$
$ADD : \{on(x, z)\}$



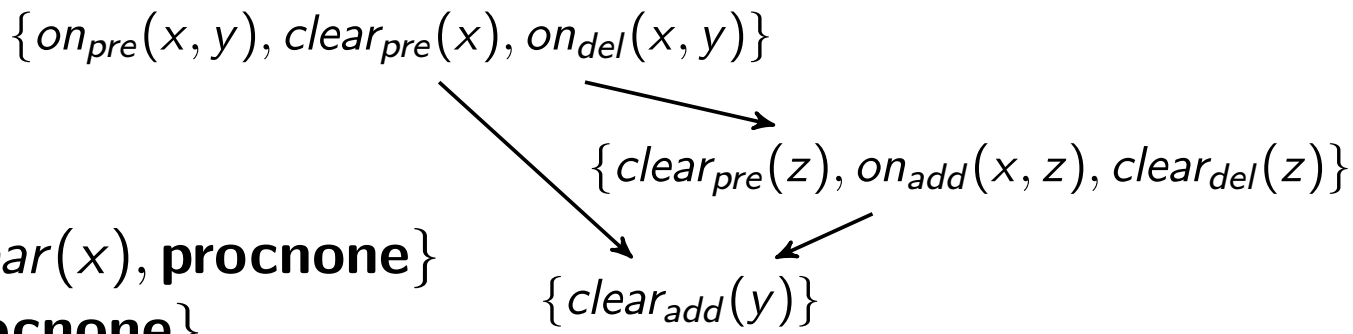
Grafos Cocientes y Splits Válidos

- Una forma de eliminar el ciclo es separando $clear_{add}(y)$:



- Si instanciamos y y z con el mismo objeto ahora, la aplicación del esquema falla en la segunda sub-acción cuando aparece la precondición $clear(z)$ (ya que este átomo no se agrega en la sub-acción anterior).

El Resultado



Move$_1(x,y)$
- $PRE : \{on(x,y), clear(x), \mathbf{procnone}\}$
- $DEL : \{on(x,y), \mathbf{procnone}\}$
- $ADD : \{\mathbf{do}_{Move_2}, \mathbf{var}_x(x), \mathbf{var}_y(y)\}$

Move$_2(x,z)$
- $PRE : \{clear(z), \mathbf{do}_{Move_2}, \mathbf{var}_x(x)\}$
- $DEL : \{clear(z), \mathbf{do}_{Move_2}, \mathbf{var}_x(x)\}$
- $ADD : \{on(x,z), \mathbf{do}_{Move_3}\}$

Move$_3(y)$
- $PRE : \{\mathbf{do}_{Move_3}, \mathbf{var}_y(y)\}$
- $DEL : \{\mathbf{do}_{Move_3}, \mathbf{var}_y(y)\}$
- $ADD : \{clear(y), \mathbf{procnone}\}$

Un Ejemplo más Simple

- La acción Drive simula un trayecto entre dos nodos de un grafo y marca el nodo destino como visitado:

$$\begin{aligned}&\textbf{Drive}(x, y : city)\\ &\quad PRE : \{at(x)\}\\ &\quad DEL : \{at(x)\}\\ &\quad ADD : \{at(y), visited(y)\}\end{aligned}$$

- En un grafo con n nodos, se generan n^2 acciones instanciadas.

Un Ejemplo más Simple

- La acción Drive simula un trayecto entre dos nodos de un grafo y marca el nodo destino como visitado:

$$\begin{aligned}&\textbf{Drive}(x, y : city)\\ &\quad PRE : \{at(x)\}\\ &\quad DEL : \{at(x)\}\\ &\quad ADD : \{at(y), visited(y)\}\end{aligned}$$

- En un grafo con n nodos, se generan n^2 acciones instanciadas.
- Un split válido de Drive en dos sub-esquemas es el siguiente (sin decoraciones):

$$\begin{aligned}&\textbf{Drive}_1(x : city)\\ &\quad PRE : \{at(x)\}\\ &\quad DEL : \{at(x)\}\\ &\quad ADD : \emptyset\end{aligned}$$

$$\begin{aligned}&\textbf{Drive}_2(y : city)\\ &\quad PRE : \emptyset\\ &\quad DEL : \emptyset\\ &\quad ADD : \{at(y), visited(y)\}\end{aligned}$$

$Drive_1$ y $Drive_2$ generan solamente $2n$ acciones instanciadas.

Con Decoratione

Drive$_1(x : city)$
$PRE : \{at(x), \textbf{procnone}\}$
$DEL : \{at(x), \textbf{procnone}\}$
$ADD : \textbf{do}_2$

Drive$_2(y : city)$
$PRE : \textbf{do}_2$
$DEL : \{\}$
$ADD : \{at(y), visited(y), \textbf{procnone}\}$

Con Decoratione

Drive$_1(x : city)$
$PRE : \{at(x), \textbf{procnone}\}$
$DEL : \{at(x), \textbf{procnone}\}$
$ADD : \textbf{do}_2$

Drive$_2(y : city)$
$PRE : \textbf{do}_2$
$DEL : \{\}$
$ADD : \{at(y), visited(y), \textbf{procnone}\}$

Notar que también debemos realizar una pequeña modificación a los problemas que que debemos agregar **procnone** al estado inicial y a la meta.

$$I' = I \cup \{\textbf{procnone}\}$$
$$G' = G \cup \{\textbf{procnone}\}$$

Evaluación: SimpleTSP

$\pi = (I, G, A)$
$I = \{at(n_1), visited(n_1)\}$
$G = \{at(n_1)\} \cup \{visited(x) : x \in Nodes\}$
$A = \{Drive(x, y)\}$

$\pi' = (I', G', A')$
$I' = I \cup \{procnone\}$
$G' = G \cup \{procnone\}$
$A' = \{Drive_1(x), Drive_2(y)\}$

#Nodes	Grounded		Runtime	
	π	π'	π	π'
10	100	20	0.1	0.1
100	10.000	200	2.8	0.2
400	160.000	800	667.86	11.97
800	640.000	1.600	T.O	97.38
1.000	1.000.000	2.000	T.O	186.03

Conclusiones

- ✓ Hemos sistematizado y automatizado trabajos previos sobre el split de acciones esquema, como un pre-proceso para planificadores que realizan instanciación.
- ✓ El método parece prometedor en dominios con grandes interfaces que previamente se dividían a mano. Podría ser una herramienta útil para usuarios sin experiencia en planificación que deseen aplicar la tecnología de planificación, pero no están muy familiarizados con ella.
- ✓ Una variante radical sería implementar el proceso de split como un tipo de aprendizaje específico del dominio, donde se podría fijar un conjunto de problemas como training y optimizar el rendimiento para un planificador en el resto de la clase

Preguntas Abiertas

- Por qué split funciona bien en algunos dominios y no en otros? Caracterizar dominios en los que podemos asegurar que split mejora performance.
- Qué accion spliteamos?
- Cuál es la interacción entre split y la heurística de búsqueda? Podemos "informar" a la heurística que esta corriendo sobre un dominio spliteado (e.g., ejecución en bloque de ciertas acciones)
- Cómo afecta split las estructuras de datos usadas por los planers actuales (i.e., mutexes, causal graph, domain transition graph)?
- Existen formas mejores de asegurar que el dominio spliteado es equivalente al original? (decoraciones más eficientes).