Curso de Inteligencia Artificial
Unidad IV: Planificación

Jose Aguilar
Cemisid, Facultad de Ingeniería
Universidad de los Andes
Mérida, Venezuela
aguilar@ula.ve

Planificación

Planificar: consiste en concebir un conjunto de acciones (un plan) de manera de alcanzar un objetivo

La planificación consiste en razonar sobre las acciones posibles a realizar para concebir planes

La planificación es un proceso mediante el cual un agente busca un plan para lograr cierto objetivo.

Planificación en Agentes

El plan es una series de acciones a tomar para, partiendo de un estado inicial, cumplir el objetivo.

Lo dicho suena muy parecido a la resolución de problemas, pero se considera a la planificación diferente,

- Utilizan una manera distinta para representar las acciones, objetivos y estados.
- Difieren en la forma de construir y representar el curso de las acciones (secuencia de acciones).

Planificación en Agentes

la planificación en sí mismo debe ser hecha por razonamiento

- No se puede hacer de forma algorítmica, ya que **la planificación y el plan de ejecución,** deben ser hechos
  - Al unísono en lugar de secuencial.
  - No pueden ser realizado por módulos aislados, debido a que c/u retroalimenta a las otras.

Un agente racional adopta y ejecuta los planes más adecuados según sus facultades de razonar acerca de ellos

Planificación en Agentes

- La planificación corresponde al nivel superior de un proceso de coordinación en un SMA

Problema es de planificar las tareas a realizar, y quienes las ejecutan (sub-planes) en un determinado orden.

- Proceso de planificación están las tareas de:
  - crear los planes,
  - asignarlos y
  - ejecutarlos.
- El segundo nivel del proceso de coordinación es el nivel de comunicación (conversaciones y actos de habla presentes), que integra los otros mecanismos de coordinación (subasta, licitación, entre otras.).

En la planificación SMA, el resultado es un plan general: una secuencia parcial o totalmente ordenada de tareas,

Un simple Agente de Planificación

Indicar(BC, sentencia_percepción(percibido, t))
actual <- Estado_Descripto(BC, t)
Si P=NoPlan entonces
    O<- Preguntar(BC, consulta_objetivo(t))
    P<-Planificar(actual, O, BC)
Si P=NoPlan Entonces
    Acción<-NoPosible
de lo contrario
    Acción<-Primero(P)
    P<-resto(P)
Indicar(BC, Acción_sentencia(Acción, t+1))

Generalidades de la Planificación

- Para representar problemas de planificación se utiliza un lenguaje particular (más restringido que la lógica de primer orden)
- El algoritmo que permite operar sobre dicho lenguaje se denomina planificador.
- La utilización de un lenguaje más restringido se debe, entre otras cosas, a propósitos de eficiencia.
- De todas maneras seguimos hablando de lenguajes formales (p.e. STRIPS)

Generalidades de la Planificación

En planificación,

- los estados se representan por un conjunto de hechos que son verdaderos, escritos en un lenguaje adecuado.

- Se consideran a la mayoría de los objetivos independientes entre sí.

subplanes

Generalidades de la Planificación

- El planificador puede añadir acciones al plan cuando las necesite, sin restringirse a una secuencia lineal

- Ejemplos de problemas de planificación,
  - Organizar una viaje,
  - Preparar una comida
  - Ir de Compras

Planificación en Agentes

Mapeo de los estados a acciones

- Elección de las **acciones posibles**
- Elección de los **ordenes temporales**
- Elección de los **sub-objetivos para trabajar**

Planificación determinista clásica

- Solo estado inicial conocido
- Conjunto de metas

Varios algoritmos de planificación diferentes

Representación de planes mediante STRIPS

- STRIPS (STanford Research Institute Problem Solver) es un lenguaje utilizado en planificación.

- Si bien ya no se utiliza el planificador STRIPS, existen numerosos planificadores que utilizan lenguajes basados/inspirados en “STRIPS”.

Representación de planes mediante STRIPS

- **Representación de Estados**: Son conjunciones de literales (conjunto de hechos que son verdaderos en un estado del mundo) sin variables ni funciones .

  En(Casa) Tiene(Pelicula)

- **Representación de Objetivos:** Son también una conjunción de literales en donde pueden haber variables.

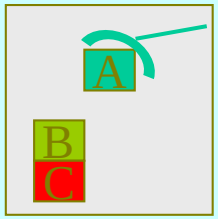
  En(Casa) ^Tiene(Pelicula)

- **Representación de Acciones:** Una acción u operador se representa por tres elementos:

  Especificación de la acción, Precondiciones, Efectos

Representación de los **Estados**

Estados del mundo se representan como conjuntos de hechos (también llamadas proposiciones).

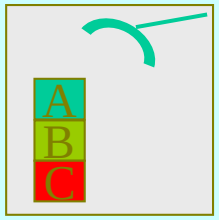


Teniendo(A)
libre(B)
en(B, C)
en la mesa (C)

Edo 1

ArticulorVacío
libre(A)
en(A, B)
en(B, C)
en la mesa (C)

Edo 2



Asunción Mundo Cerrado (CWA):? El hecho que no figure un estado se supone que es falso, suponiendo que el agente tiene observabilidad completa.

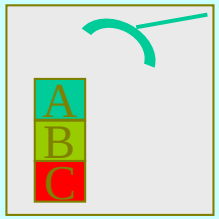
Representación de las **Metas**

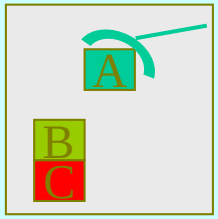
Las metas también se representan como conjuntos de hechos.
Por ejemplo {en (A, B)} es un objetivo en el mundo bloques.

Un estado meta es cualquier estado que contiene todos los **hechos meta.**

ArticulorVacío
libre(A)
en(A, B)
en(B, C)
en la mesa (C)

Edo 1





Teniendo(A)
libre(B)
en(B, C)
en la mesa (C)

Edo 2

Estado 1 es un estado meta para el objetivo de {en(A, B)}.
Estado 2 no es un estado meta para el objetivo de {en(A, B)}.

Una Representación del Problema de Planificación

- Estado Inicial: sentencia lógica sobre situación inicial (*So)*

  En(Casa,So) ∧ ¬ Tener(leche,So) ∧ ¬ Tener(cambures,So)

- Estado Objetivo: sentencia lógica sobre situación deseada

  ∃s En(Casa,s) ∧ Tener(leche,s) ∧ Tener(cambures,s)

- Operador: describe acciones que se pueden hacer

  ir(sitio), comprar(objeto)

- Plan: secuencia de acciones

  [**ir(Supermercado), Comprar (leche),Comprar(cambures),ir(casa)]**

Operador

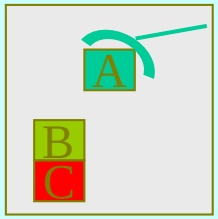
- Componentes posibles para describir Operador:
  - Acción a realizar
  - Precondición: situación actual de donde debe partir
  - Efecto: resultado de la operación

  Op(**Acción**(Ir(allá), **Precondición(**(En(aquí),Camino(aquí, allá)),
  Efecto(En(allá) $\wedge \neg$ En(aquí))

- Ejemplo de instanciación:

  Situación Actual: En(casa), Camino(Casa, Supermercado)
  Acción: Ir(supermercado)
  Efecto: En(supermercado), $\neg$ En(Casa)

Representando Acciones en STRIPS



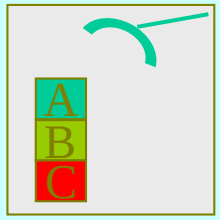
Teniendo(A)
libre(B)
en(B, C)
en la mesa (C)

Edo 1

PutDown(A,B)

ManoVacía
libre(A)
en(A, B)
en(B, C)
en la mesa (C)

Edo 2



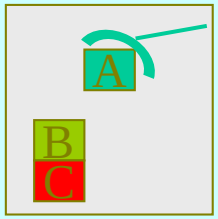
Una acción STRIPS especifica:

1) un conjunto PRE de hechos de pre condiciones
2) un conjunto ADD de hechos de efectos a añadir
3) un conjunto DEL de hechos de efecto a borrar

PutDown(A,B):

PRE: {Teniendo(A), libre(B) }
ADD: { en(A,B), ManoVacía, limpio(A) }
DEL: {Teniendo(A), libre(B) }

Semantica de las acciones STRIPS



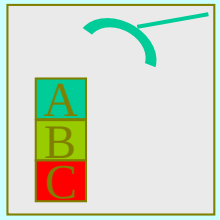
Teniendo(A)
libre(B)
en(B, C)
en la mesa (C)

S

PutDown(A,B)

ManoVacía
libre(A)
en(A, B)
en(B, C)
en la mesa (C)

S* ∪ *ADD – DEL



- Una acción STRIPS se aplica (o se permite) en un estado cuando su Pre- condiciones están contenidos en el estado.

- Tomando una acción en un estado ***S*** resulta en un nuevo estado ***S* ∪ *ADD – DEL*** (Es decir, añade los efectos "añadir" y eliminar los efectos "eliminación")

PutDown(A,B):
PRE: {Teniendo(A), libre(B) }
ADD: { en(A,B), ManoVacía, limpio(A) }
DEL: {Teniendo(A), libre(B) }

Problema de Planificación

- Una forma de representación de Planes
  - Un Conjunto de Pasos (acciones)
  - Un conjunto de Restricciones: de orden o de asignaciones
  - Un conjunto de enlaces causales

- Ejemplo:

Plan(Pasos S1: Op(Acción:comenzar),
S2:Op(Acción:terminar),
Precondicion(ZapatoDerecho $\wedge$ ZapatoIzquierdo)),
Orden(S1; S2)
Asignación{}
Enlaces{})

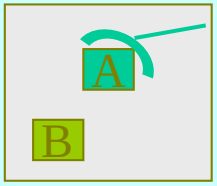
Problema Planificación STRIPS

Un problema de planificación STRIPS especifica:
1) un estado inicial S
2) un objetivo G
3) un conjunto de acciones STRIPS

Objetivo: encontrar una secuencia de acción "corto" llegar a un estado objetivo, o señalar que el objetivo es alcanzable

Ejempl0

Solución: (PutDown(A,B))



Teniendo(A)
libre(B)
en la mesa (B)

Edo Inicial

en(A,B)

objetivo

PutDown(A,B):
PRE: { teniendo(A), libre(B) }
ADD: { en(A,B), ManoLibre, libre(A)}
DEL: {teniendo(A), libre(B) }

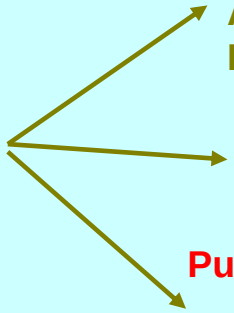
PutDown(B,A):
PRE: {teniendo(B), libre(A) }
ADD: {en(B,A), ManoLibre, libre(B) }
DEL: {teniendo(B), libre(A) }

Acciones STRIPS

Esquemas de Acciones

Esquema Acción: (x e y variables)

PutDown(x,y):
PRE: { teniendo(x), libre(y) }
ADD: { en(x, y), ManoLibre, libre(x)}
DEL: {teniendo(x), libre(y) }



PutDown(B,A):
PRE: {teniendo(B), libre(A) }
ADD: {en(B,A), ManoLibre, libre(B) }
DEL: {teniendo(B), libre(A) }

PutDown(A,B):
PRE: { teniendo(A), libre(B) }
ADD: { en(A,B), ManoLibre, libre(A)}
DEL: {teniendo(A), libre(B) }

- El reemplazar las variables con los objetos del estado inicial y el objetivo produce una acción STRIPS.
- Dado un conjunto de esquemas, un estado inicial y un objetivo, los planificadores compilan esquemas en acciones

Problema de Planificación

- ¿Cómo crear Planes?

Objetivo: ZapatoDerecho $\wedge$ ZapatoIzquierdo

Op(Acción:ZapatoDerecho, Precondicion(MediaDerecha), Efecto(ZapatoDerecho))

Op(Acción:MediaDerecha, Efecto(MediaDerecha))

Op(Acción:ZapatoIzquierdo, Precondicion(MediaIzquierda), Efecto(ZapatoIzquierdo))

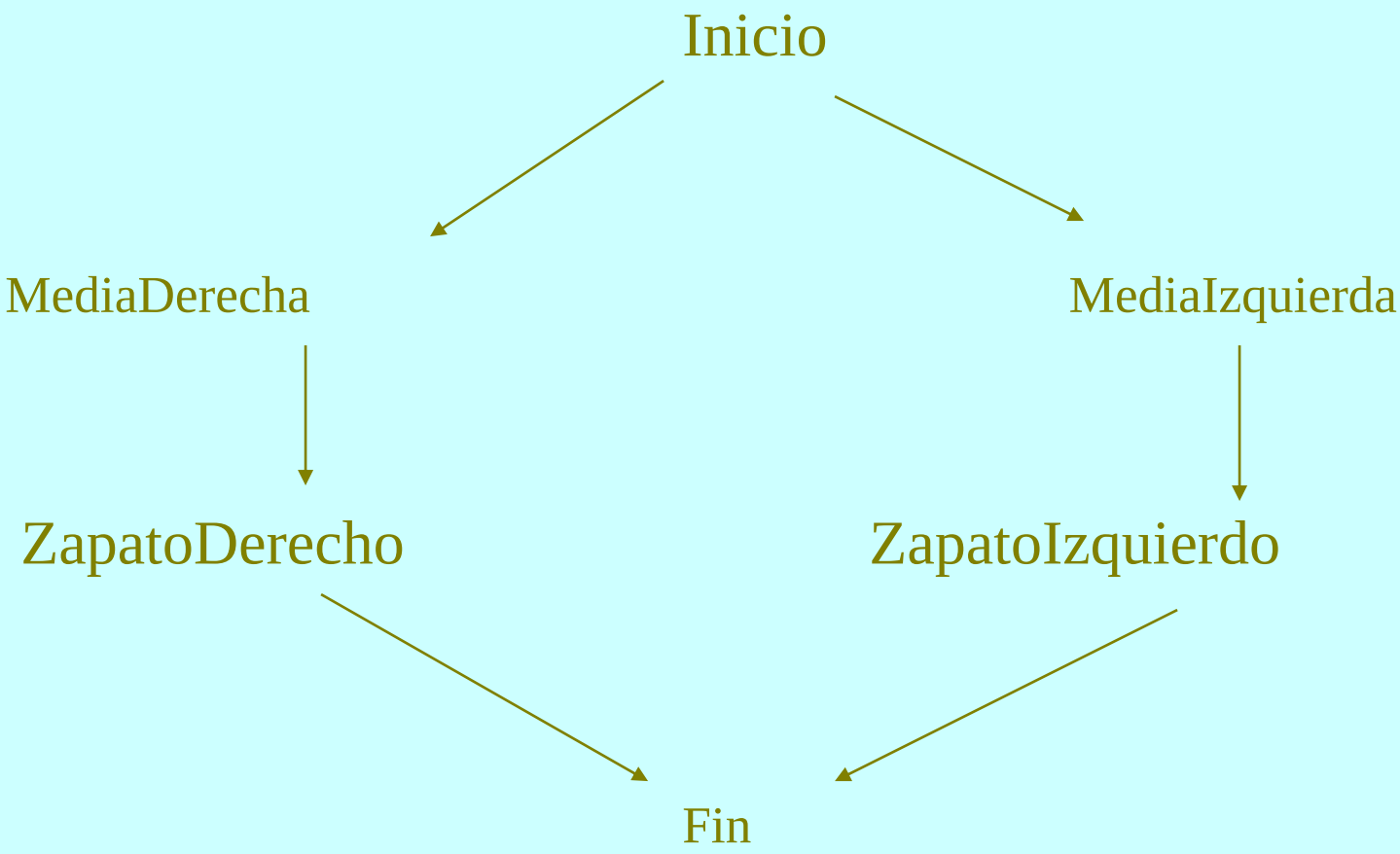
Op(Acción:MediaIzquierda, Efecto(MediaIzquierda))

- **Solución** **Orden Parcial**

  Orden Total

Problema de Planificación

Plan Orden Parcial:



Problema de Planificación

Plan Orden Total:

Inicio	Inicio	Inicio	Inicio	Inicio	Inicio
MediaDer	MediaDer	MediaDer	MediaIzq	MediaIzq	MediaIzq
MediaIzq	MediaIzq	ZapatoDer	MediaDer	MediaDer	ZapatoIzq
ZapatoDer	ZapatoIzq	MediaIzq	ZapatoDer	ZapatoIzq	MediaDer
ZapatoIzq	ZapatoDer	ZapatoIzq	ZapatoIzq	ZapatoDer	ZapatoDer

Problema de Planificación

Solución (plan) debe ser:

- **Completa:** cada precondición de un paso *Sj* es lograda por otra de otro paso *Si*

  1. Si logra precondición *c* de *Sj* sii: orden(Si; Sj) y $c \in$ efectos(Si)
  2. $\neg\exists$Sk tal que $\neg c \in$ efectos(Sk) y orden(Si; Sk; Sj)

- **Consistente:** no hay contradicciones en las restricciones de orden o de asignación

Orden:	Si; Sj	Sj; Sk	Sk; Si
Asignación:	v=A	y	v=B

¿Cómo crear Planes?

Inicio:

Op(Acción:Inicio, Efecto(En(casa) y Vende(tienda, ropa) y Vende(super, leche) y Vende(super, frutas))

Fin (Objetivo):

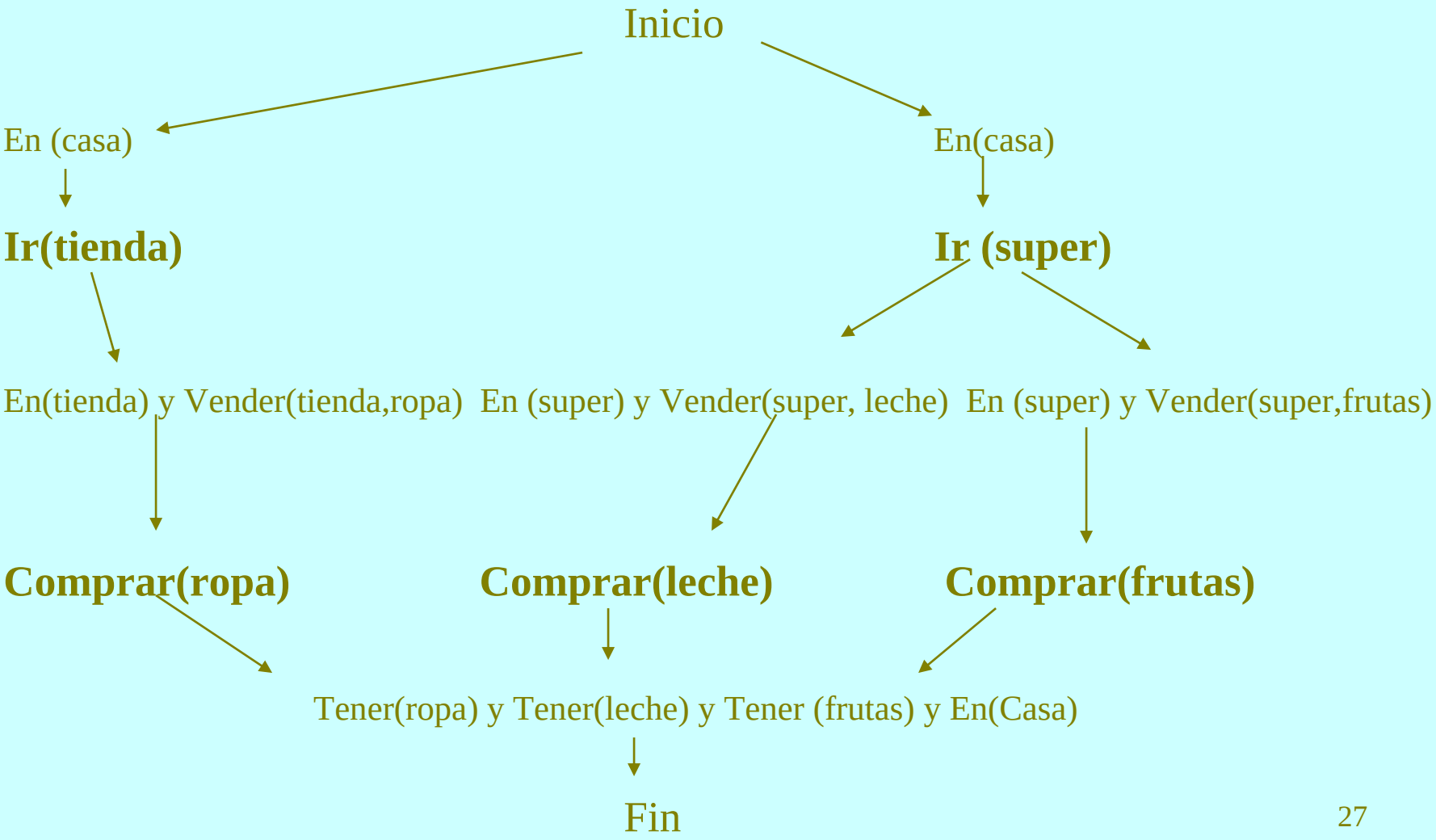
Op(Acción:Fin, Precondicion(Tener(ropa) y Tener(leche) y Tener (frutas) y En(Casa))

Acciones posibles:

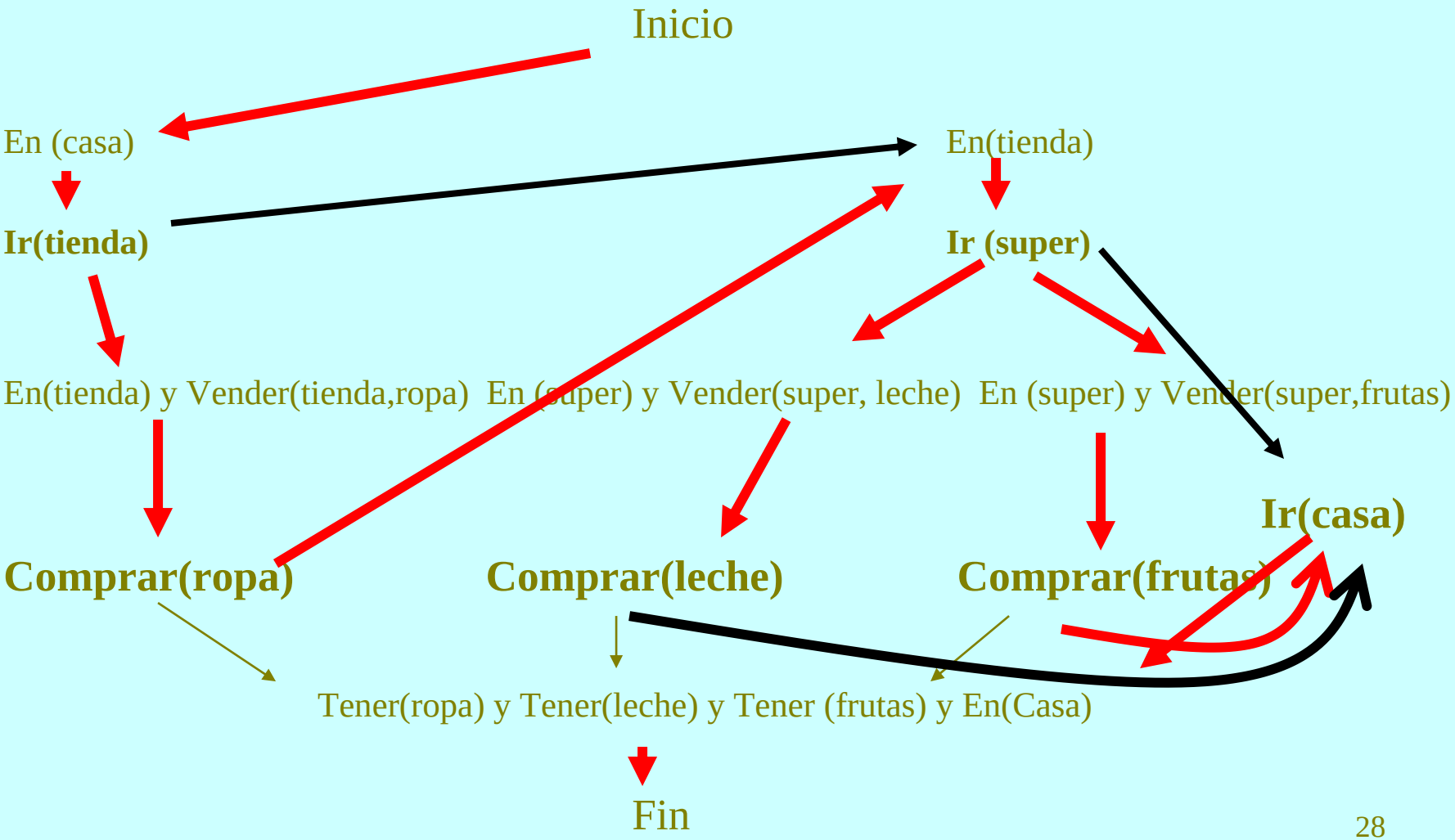
Op(Acción:Ir(allá), Precondicion(En (aquí)), Efecto(En(allá) y ¬En(aquí))

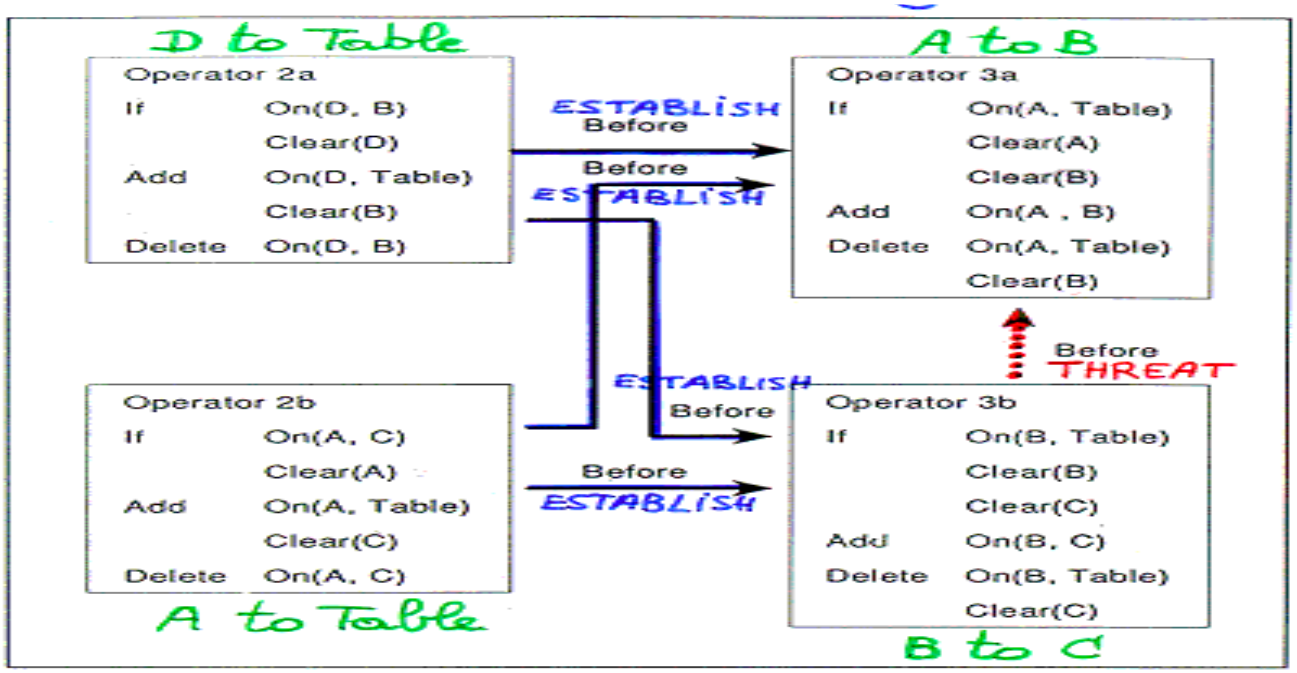
Op(Acción:Comprar(x), Precondicion(En (y) y Vende(y, x)), Efecto(Tener(x))

Ejemplo


Inicio
En (casa)
En(casa)
Ir(tienda)
Ir (super)
En(tienda) y Vender(tienda,ropa)
En (super) y Vender(super, leche)
En (super) y Vender(super,frutas)
Comprar(ropa)
Comprar(leche)
Comprar(frutas)
Tener(ropa) y Tener(leche) y Tener (frutas) y En(Casa)
Fin

Ejemplo


Inicio
En (casa)
En(tienda)
Ir(tienda)
Ir (super)
En(tienda) y Vender(tienda,ropa)
En (super) y Vender(super, leche)
En (super) y Vender(super,frutas)
Ir(casa)
Comprar(ropa)
Comprar(leche)
Comprar(frutas)
Tener(ropa) y Tener(leche) y Tener (frutas) y En(Casa)
Fin



Un plan completo

Algoritmo de Planificación de Orden Parcial

POP

plan (minimo_plan(inicio,objetivo))

repita

Si solución=plan entonces regresar plan

<Snecesario, c>=selección-subobjetivo(plan)

Escoger_operador(plan, operadores, Snecesario, c)

Resolver_camino(plan)

Selección_subobjetivo

Tomar un paso del plan (Snecesario) con una precondicion c no alcanzada

regresar Snecesario y c

Algoritmo de Planificación de Orden Parcial

Escoger Operador

Escoger paso Sanadir de los operadores del plan con c como efecto
Si no hay tal paso entonces falla
de lo contrario
    añadir <Sanadir, c, Snecesario> a enlaces(plan)
    añadir (Sanadir; Snecesario) a Orden(plan)
    tal que: inicio; …; Sanadir; Snecesario ;…; Fin en Orden(plan)
Si Sanadir es un nuevo paso escogido no existente en pasos(plan) entonces
    añadir Sanadir a pasos(plan)

Resolver camino

por cada enlace <Si, c, Sj> en Enlaces(plan)
  por cada paso Sr en Orden(plan) que pueda amenazar <Si, c, Sj> c= ¬c´ y c´=Efectos(Sr)
        escoger entre Añadir Sr; Si a Orden(plan)
        o eliminar Si; Sr en Orden(plan)
    Si no se puede entonces consistencia plan falla

Ing. de Conocimiento para Planificación

1. Describir Problema
2. Decidir vocabulario de condiciones, operadores, y objetos
3. Codificar operadores (lo más genéricos posibles)
4. Codificar descripción del problema en especifico (So, Fin, etc.)
5. probar al planificador

Operaciones Avanzadas

- **Mediciones y/o calculos:**

Op(Acción: comenzar,
Efecto: dinero<-(1000Bs) Y NivelGasolina<-5litros) condición inicial
...
Op(Acción: Comprar(x,tienda),
Efecto: Tener(x) Y dinero<-dinero-Precio(x,tienda)
...
Op(Acción: llenar(NivelGasolina)
Efecto:NivelGasolina<-40litros Y
dinero<-dinero - (PrecioUnidad(gasolina)x
(NivelGasolina - 5litros)

- **Descomposición:**

Op(Acción: Comprar(x)
Efecto: Tener(x) Y ¬ Tener(Moneda)
Precond: 1. Vender (tienda, x) Y
2. En (tienda) Y
3. Tener (moneda)

Operaciones Avanzadas

- **Efectos Condicionales**

Op(Acción: Mover (b, x y)
    Precond: En (b, x) Y Limpio (y)
    Efecto: En(b, y) Y Limpio(x) *cuando* x,y=mesa

- **Cuantificador Universal:**

Op(Acción: CambiarContenidoBolso(x, y)
    Precond: Bolso(x) y Bolso(y) y ContenidoEn(x)
    Efecto: ContenidoEn(y) Y ¬ ContenidoEn(x) Y
        (∀i item(i) => (En(i,y) Y ¬ En(i, x)) *cuando* Dentro (i, bolso(y)))

Operaciones Avanzadas: Parte Condicional

Op(A: quitar(x), P: En(x), E: fuera(x))

Op(A: poner(x), P: fuera(x), E: En(x))

Op(A: inflar(x), P:Intacto(x), desinflado(x),E:inflado(x))

Objetivo: En(x) Y Inflado(x)

Condición Inicial:

Inflado(rep)Yintacto(rep)Yfuera(rep)Yen(caucho)Ydesinflado(caucho)

Plan: [quitar(caucho), poner(rep)]

Extender plan con:

If (<condition>, <thenpart>,<elsepart>) (*)

Ejemplo: If (Intacto(caucho), [Inflar(caucho)], [quitar(caucho), poner(rep)])

Otra Operación: Op(ChequearCaucho(x), P: Caucho(x), E: Intacto(x)?

Op (Acción:... Precond: cond1 cuando estado1; cond2 cuando estado2

Efecto: efecto1 cuando estado1 ; efecto 2 cuando ...

Un Agente de Planificación **con Condición** (*)

```
Indicar(BC,sentencia_percepción (percibido,t))
Actual <- Estado_Descripto(BC, t)
Si P=NoPLan entonces
      O<- Preguntar(BC,consulta_objetivo(t))
      P<-POP(actual, O, BC)
Si P=NoPlan Entonces Acción<-NoPosible
de lo contrario
   si no es una condición (if)
                Acción<-Primero(P)
   de lo contrario
                Si preguntar(BC,partecondicion(condición)=V entonces
                          Acción<-parteinicio(condición)
                de lo contrario Acción<-partefin(condición)
    P<-resto(P)
   Indicar(BC, relación_ Acción_sentencia (Acción,t))
```

(*)

Un Agente de Planificación con **Replanificación**

Indicar(BC, sentencia_percepción(percibido,t))
Actual <- Estado_Descrito(BC, t)
Si P=NoPLan entonces
    O<- Preguntar(BC, consulta_objetivo(t))
    P<-planificador(actual, O, BC)
Si P=NoPlan Entonces Acción<-NoPosible
De lo contrario
    Si Precondición(P) no es verdad en BC
        P'<- Escoger-Mejor-Continuación(actual)
        P<-actualizar (planificador(actual, Precondición(P'), BC)) **(replanif)**
    Acción<-Primero(P)
    P<-resto(P)

Algoritmos de Planificación

La planificación implica una búsqueda compleja:

- Los operadores alternativos para lograr un objetivo
- Múltiples objetivos que interactúan

Solidez

- Un algoritmo de planificación es solido si todas las soluciones son planes legales

Completitud

- Un algoritmo de planificación es completa si una solución se puede encontrar cada vez que existe una

Optimalidad

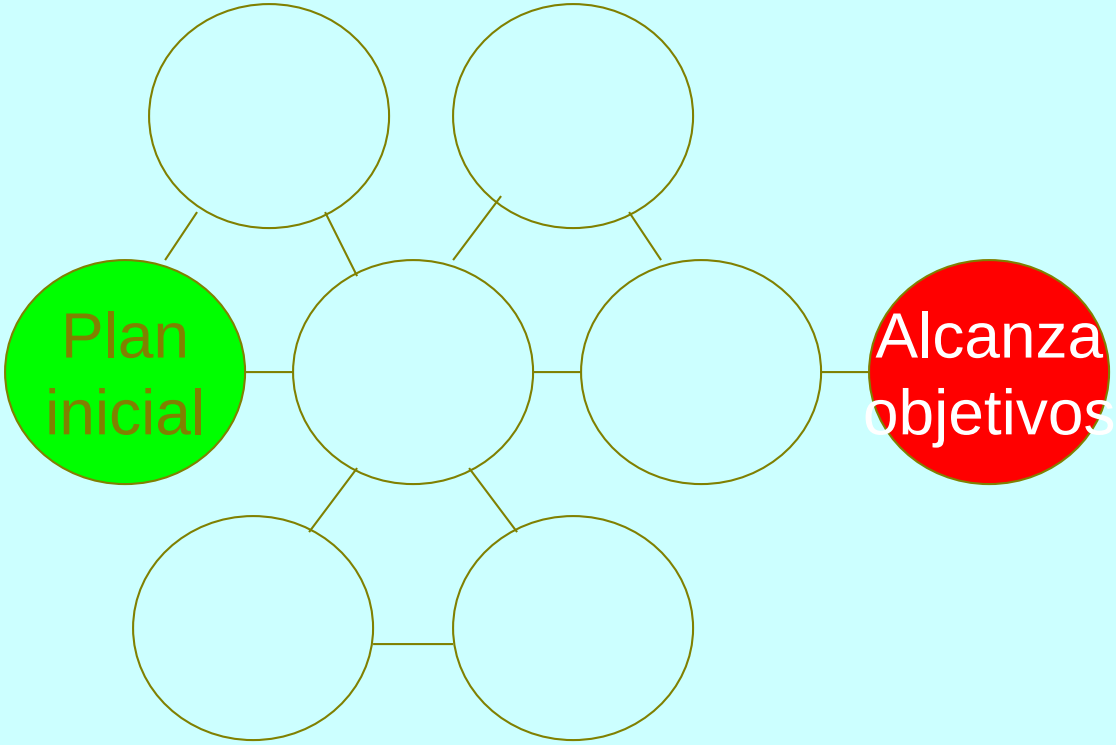
- Un algoritmo de planificación es óptima si se maximiza una predefinido medida de la calidad del plan

Librerías de Planificación de Agentes

- Prodigy Planner http://www.cs.cmu.edu/afs/cs.cmu.edu/project/prodigy/Web/prodigy-home.html
- GPS
- FF
- FD
- LAMA
- Graphplan
  - PLPLAN: Implementación en Java de GraphPlan
  - NPlanner: Implementación en .NET de GraphPlan
  - Emplan y JavaGP: Implementación en C++ y Java

Base

- Cada nodo es un plan parcial



Graphplan

http://www.cs.cmu.edu/~avrim/graphplan.html

- Búsqueda hacia adelante combinada con la búsqueda hacia atrás.

- Construye un grafo de la planificación

- Dos etapas:
  - Extender: Un paso de tiempo en el grafo de planificación.
  - Buscar: un plan válido en el grafo de planificación.

- Graphplan encuentra un plan o prueba que ningún plan existe

Graphplan

- El algoritmo toma como entrada un problema de planificación expresado en STRIPS y produce, de ser posible, una secuencia de operaciones para llegar a un estado final.

- En un *grafo de planificación*:
  - los nodos son las acciones y hechos atómicos, dispuestos en niveles alternos,
  - y las aristas son de dos tipos:
    - de un hecho atómico a las acciones de las cuales es una condición,
    - de una acción a los hechos atómicos donde se convierte en verdadero o falso.

Graphplan

- El primer nivel contiene hechos atómicos verdaderos que identifican el estado inicial.

- También se mantienen las listas de hechos incompatibles que no pueden ser verdaderos al mismo tiempo y acciones incompatibles que no se pueden ejecutar en conjunto.

- El algoritmo **extiende** iterativamente el grafo de planificación, probando que no hay soluciones de longitud i-1 antes de buscar planes de longitud i por encadenamiento hacia atrás:
  - suponiendo que los objetivos son verdaderos, el algoritmo de Graphplan **busca l**as acciones y estados previos desde el cual el objetivo puede ser alcanzado, podando muchos de ellos mientras sea posible gracias a la información incompatible.

Planning Graph

- Espacio de estado: plan es la secuencia de acciones
- Plan-espacio: plan es parcialmente ordenado conjunto de acciones

→**Planning graph**: sucesión de conjuntos de acciones paralelas

ej: ({a1, a2}, {a3, a4}, {a5, a6, a7})

Ejemplo traslado cohete

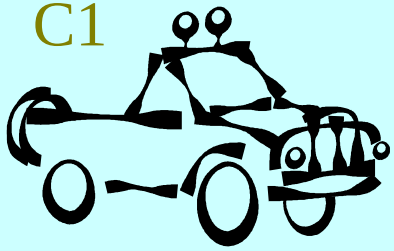
St. Louis

R1

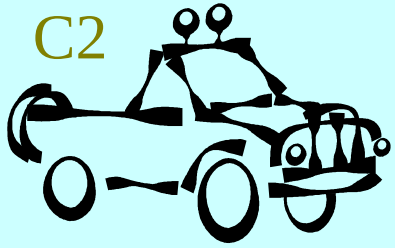
R2

R3

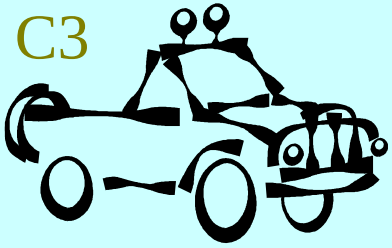
C1



C2



C3



San Francisco

Seattle

- La solución se puede generalizar en 3 pasos

Ejemplo traslado cohete

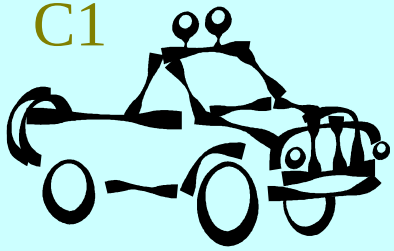
St. Louis

R1

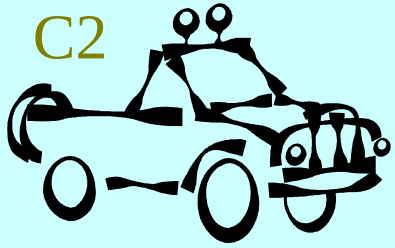
R2

R3

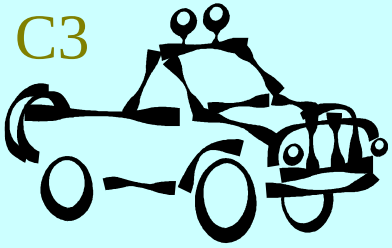
C1



C2



C3



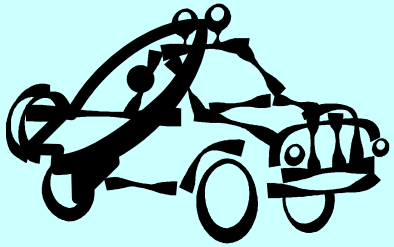
San Francisco

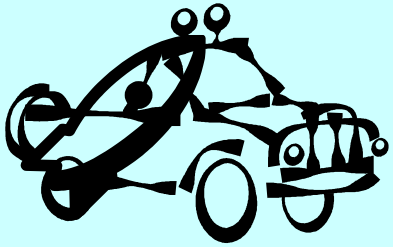
Seattle

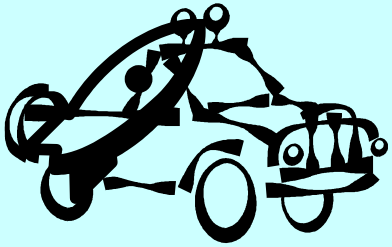
- Paso 1: Cargar todos los cohetes

Ejemplo traslado cohete

St. Louis







San Francisco

Seattle

- Paso 2: Mueva todos los cohetes

Ejemplo traslado cohete

St. Louis

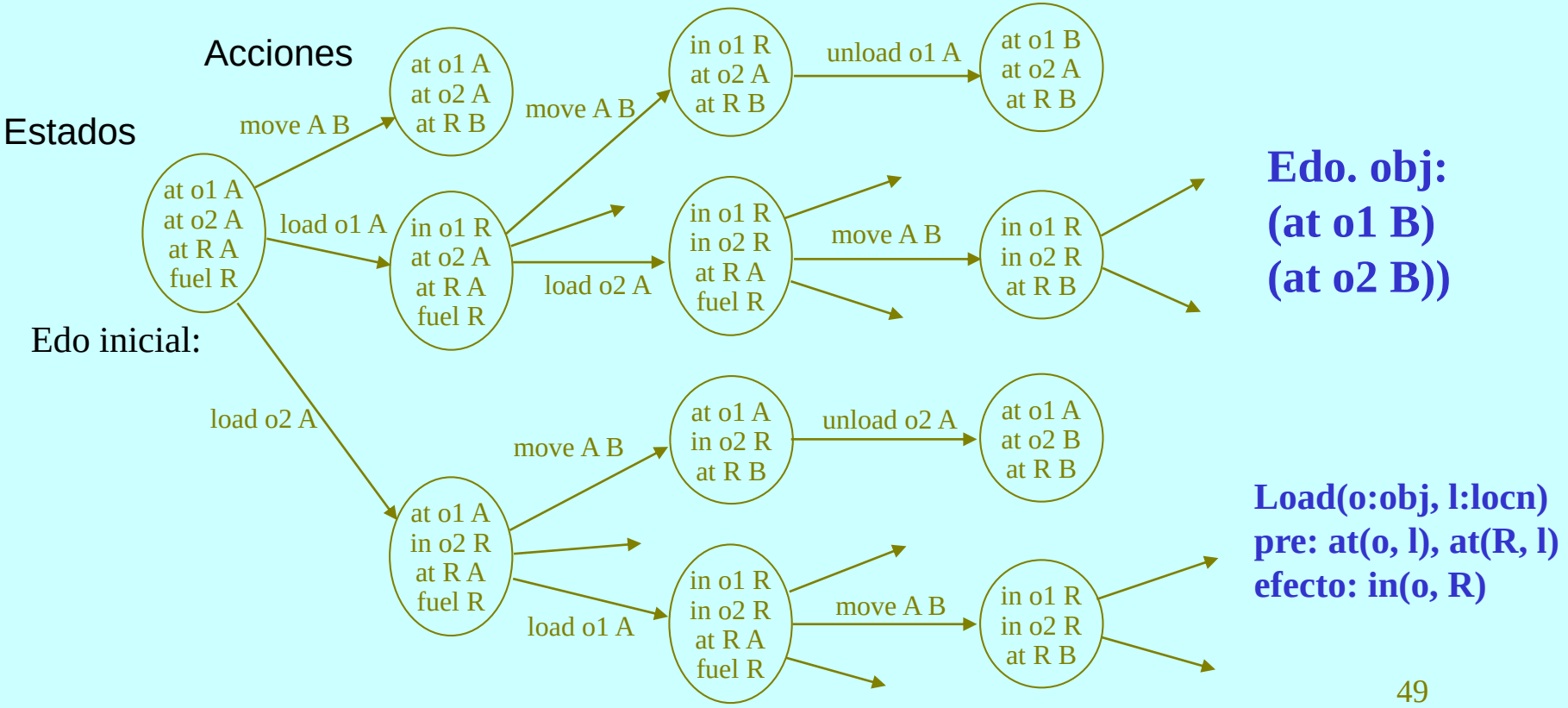
San Francisco

Seattle

- Paso 3: lanzar todos los cohetes

Graphplan

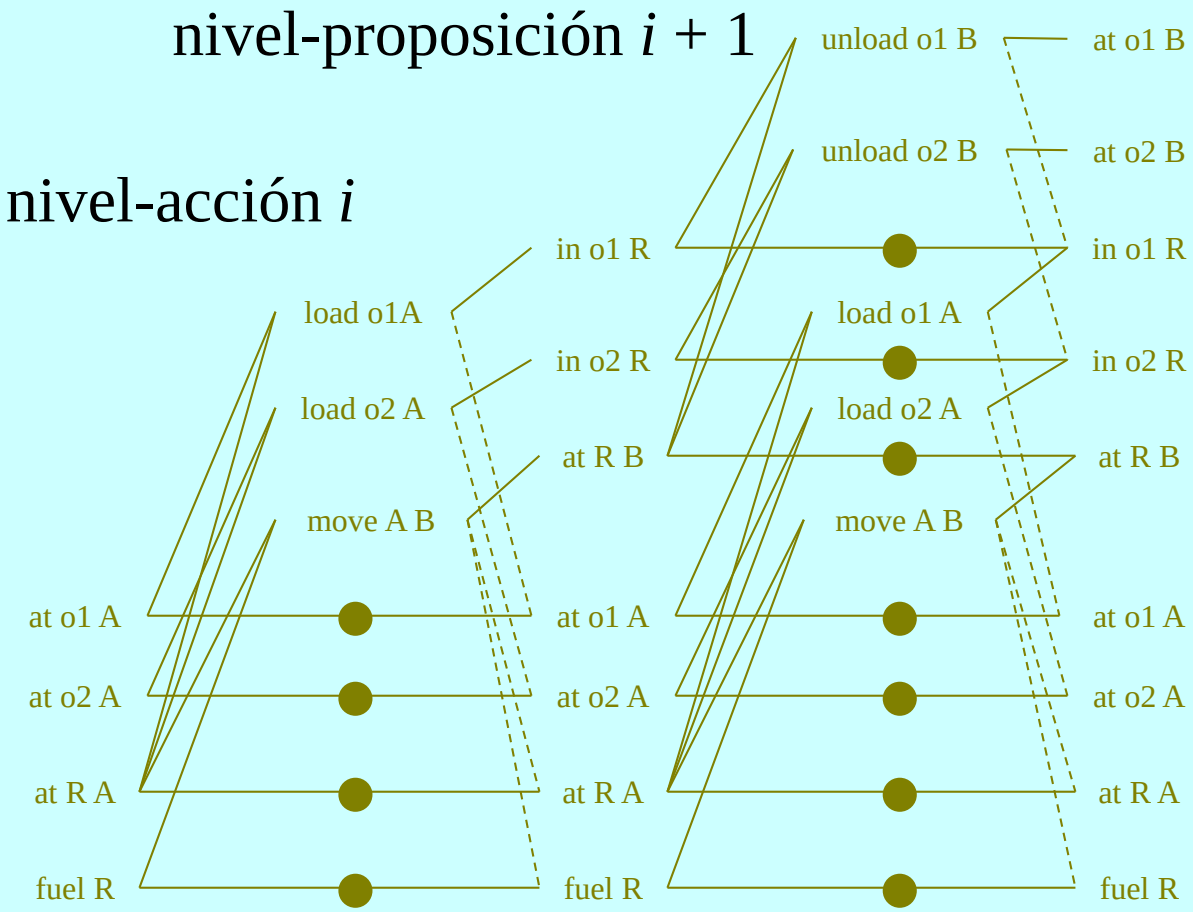
- Accesibilidad de Estado - "hasta" el objetivo
  - Puede encontrar todas las metas alcanzables desde el estado inicial
  - Exponencial en tiempo y memoria



Extender grafo de planificación

- Crear un nivel-acción *i*:
  - Agregar c/operador instanciado, para los cuales sus precondiciones estan presentes en el nivel-proposición *i*
- Crear un nivel-proposición *i* + 1:
  - Agregar todos los efectos de las acciones insertadas en el nivel-acción *i*

Extender grafo de planificación

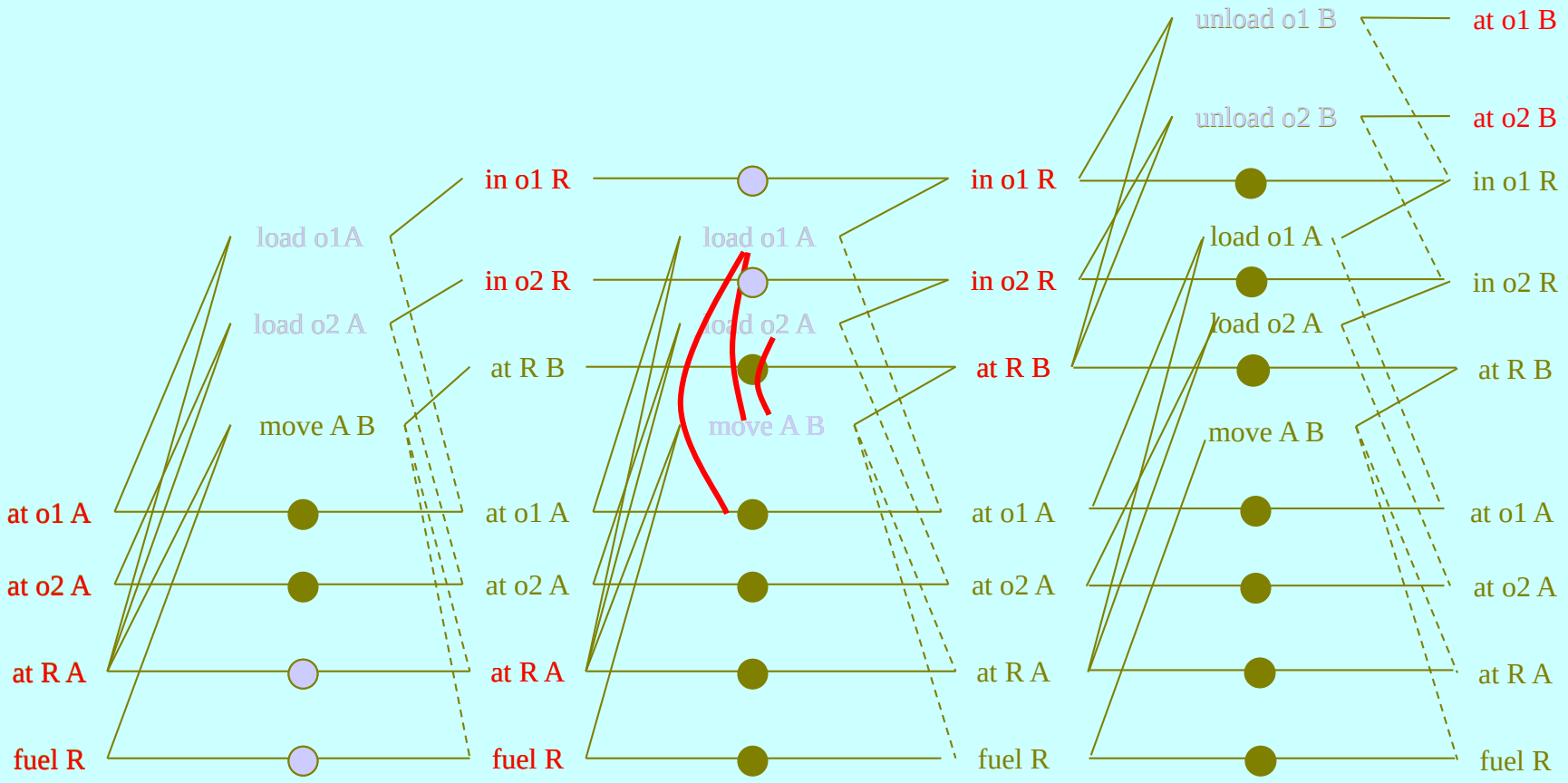


Buscar en el grafo de planificación

- Enfoque de encadenamiento hacia atrás nivel por nivel
  - Dado un conjunto de metas en el tiempo t, identificar todos los conjuntos de acciones en el tiempo t - 1 que alcanzan esas metas. Las condiciones previas de estas acciones son las nuevas metas para t – 1.
- Proceso recursivo
  - Por cada objetivo en el tiempo t en algún orden arbitrario: Seleccionar acción en el tiempo t - 1 que logra ese objetivo.
  - Hacer esto de forma recursiva a todos los objetivos en el tiempo t -1, y no añadir nuevas acciones, sino utilizar las ya seleccionadas para otro objetivo, si es posible.
  - Si la recursividad fracasa, seleccione una acción diferente.

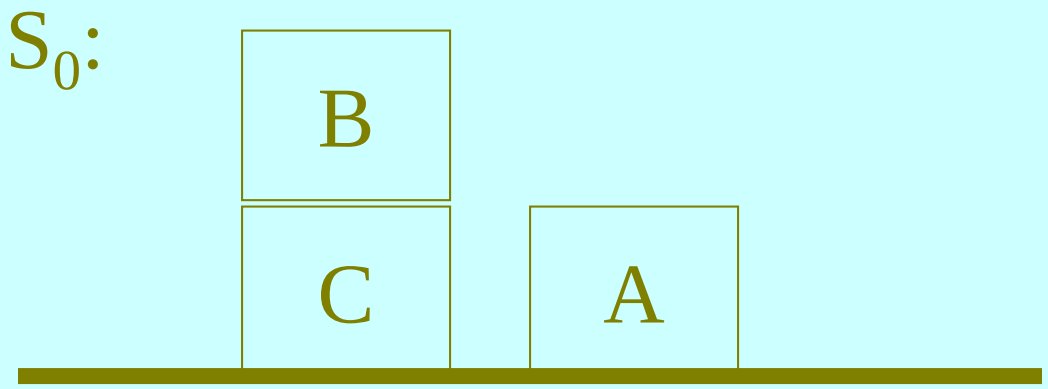
El nuevo conjunto objetivo es el conjunto de todas las condiciones previas de las acciones seleccionadas

Buscar en el grafo de planificación

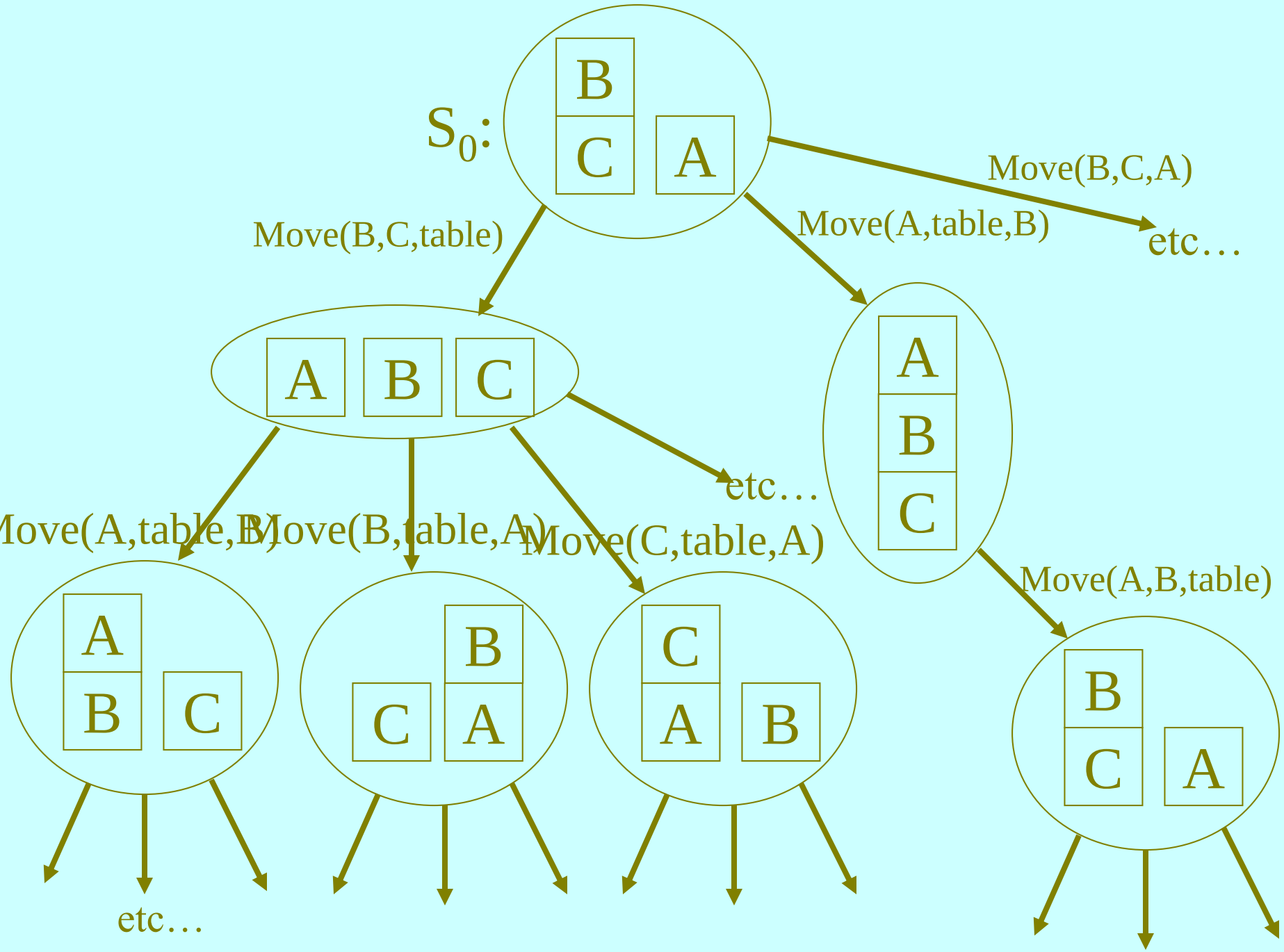


{load o1 B, load o2 B} **{move A B}** **{unload o1 B, unload o2 B}**

Otro ejemplo

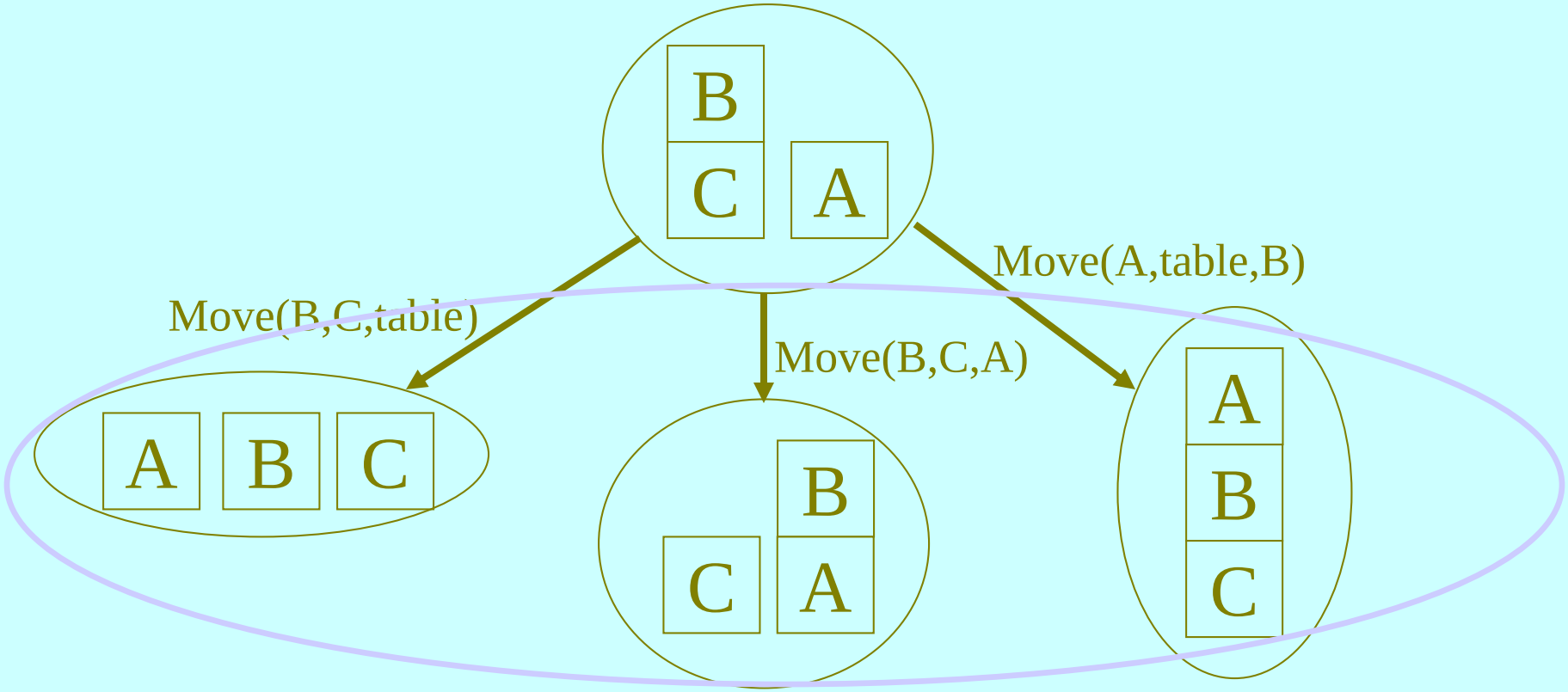


- Move(x, y, z)
  - Precond: On(y, x), Clear(x), Clear(z), etc.
  - Effects: On(z, x), ~On(y, x), Clear(y), etc.

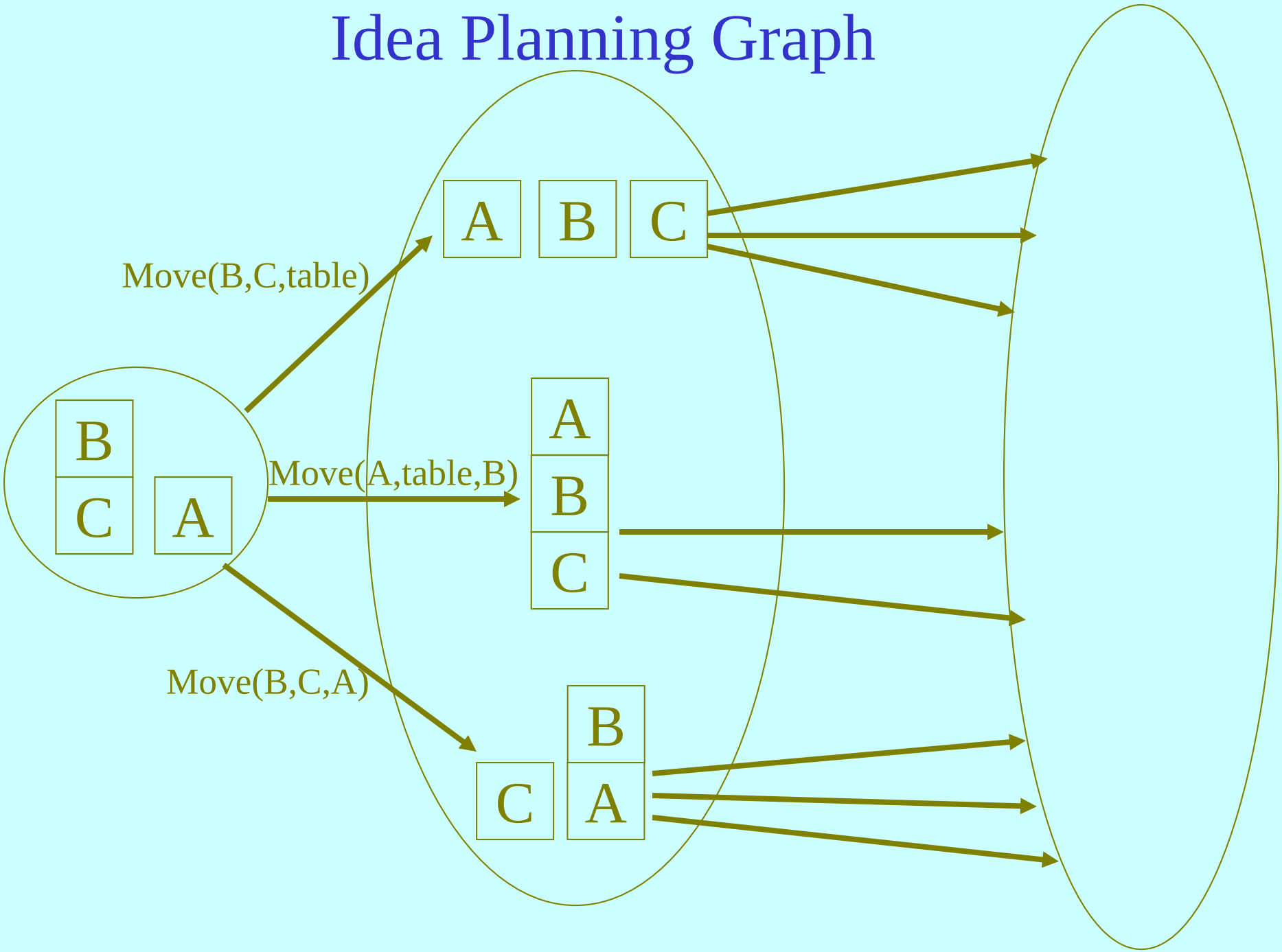

S_0:
B
C
A
Move(B,C,A)
etc…
Move(B,C,table)
Move(A,table,B)
A
B
C
A
B
C
etc…
Move(A,table,B)
Move(B,table,A)
Move(C,table,A)
Move(A,B,table)
A
B
C
B
C
A
C
A
B
B
C
A
etc…

Planning Graphs

- ¿Qué pasaría si todos los estados alcanzables desde s0 se modelaron como un solo estado?

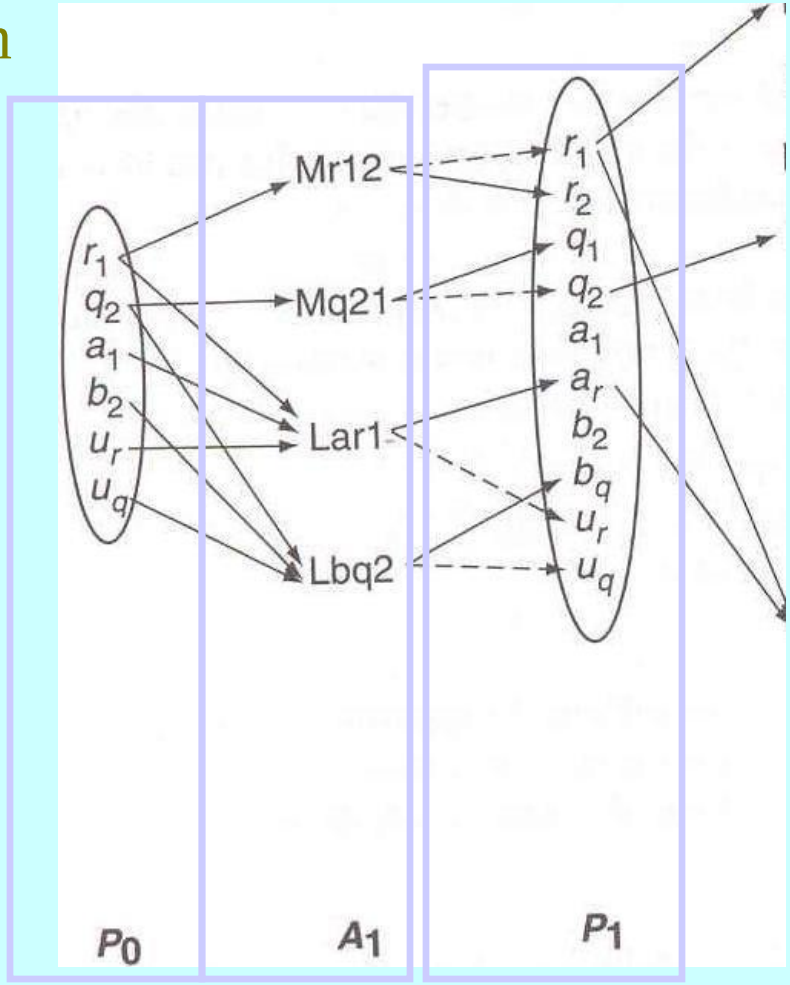


Idea Planning Graph



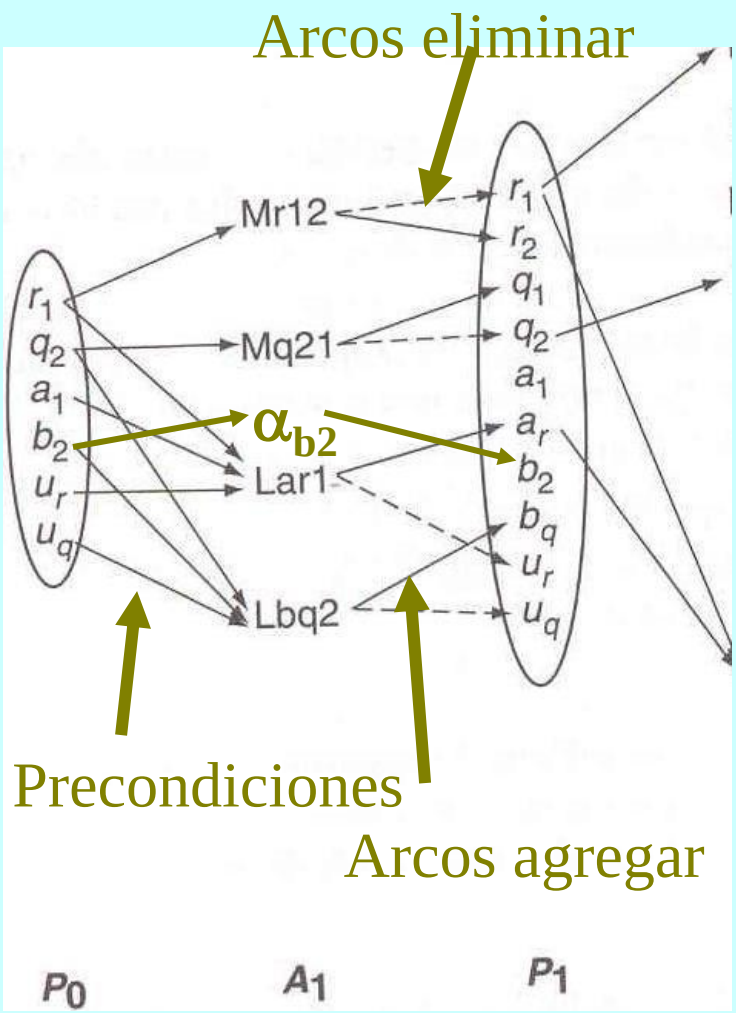
Planning Graphs

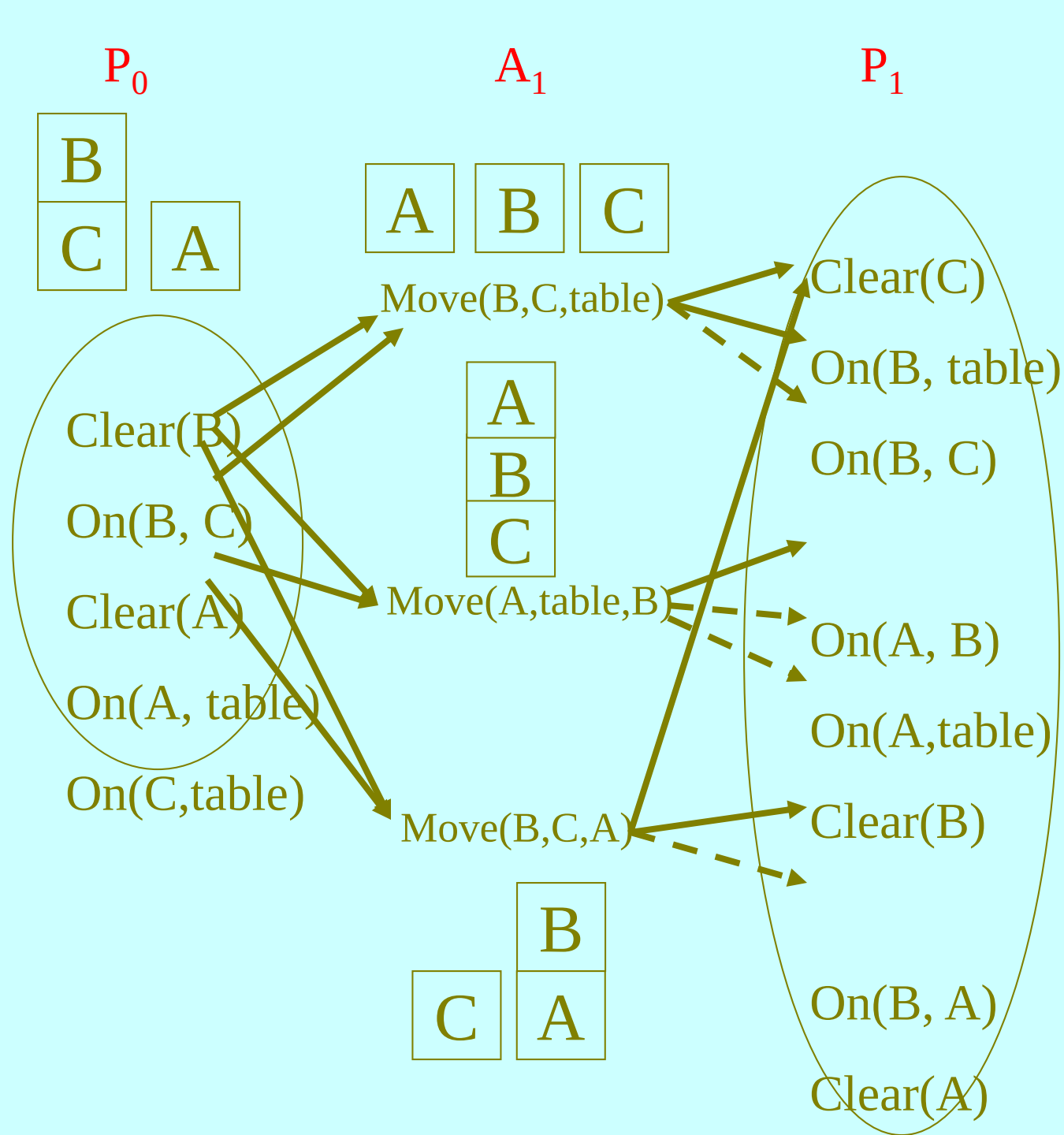
- Planning graph = grafo dirigido en capas con niveles alternos de proposiciones (P) y acciones (A)
- P_0 = edo inicial
- A_n = conjunto de acciones con precondiciones en P_n
- P_n = conjunto de proposiciones que puedne ser verdad despues de n n acciones ejecutadas
  ie: $P_{n-1} \cup \text{effectos}^{+}(A_1)$



Planning Graphs

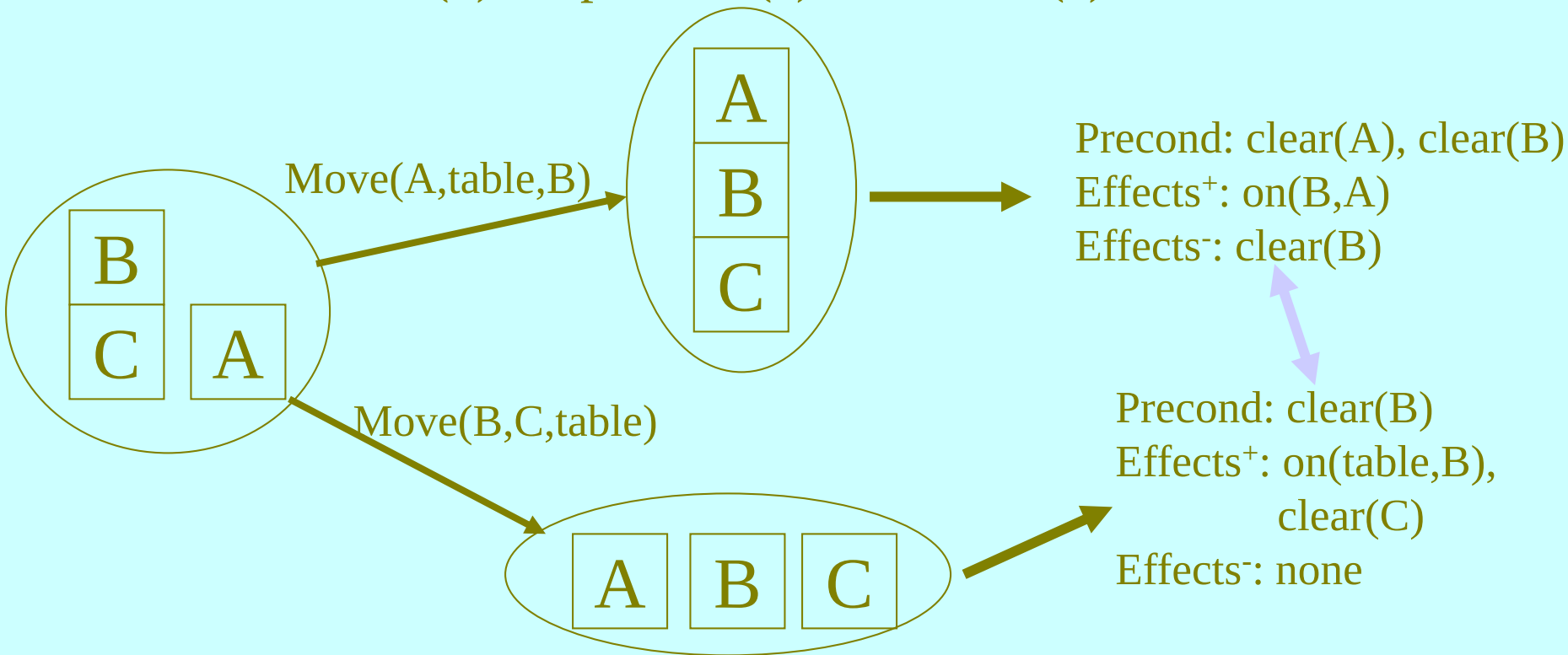
- arcos de pre condición previa pasan de P_n a las acciones asociadas en A_n
- Arcos añadir indican efectos positivos de las acciones
- Arcos borrar indican los efectos negativos de las acciones
- También definir un no-op operador α_p : precond(α_p) = effects$^+$(α_p) = p and effects$^-$(α_p) = ∅
- Tenga en cuenta que los efectos negativos no se quitan, se marcan.
- $P_{n-1} \subseteq P_n$: "principio de la persistencia




P0
A1
P1
B
C
A
A
B
C
Move(B,C,table)
Clear(C)
On(B, table)
On(B, C)
A
B
C
Clear(B)
On(B, C)
Clear(A)
On(A, table)
On(C,table)
Move(A,table,B)
On(A, B)
On(A,table)
Clear(B)
Move(B,C,A)
B
C
A
On(B, A)
Clear(A)

Definiciones

- 1) Dos acciones (a, b) son independientes si::
  - $effects^{-}(a) \cap [precond(b) \cup effects^{+}(b)] = \varnothing$
  - $effects^{-}(b) \cap [precond(a) \cup effects^{+}(a)] = \varnothing$



Definiciones

- 2) Un conjunto de acciones independientes, π, es aplicable a una precond si y sólo si precond(π) $\subseteq$ s
- 3) Un **plan de capas** es una secuencia de grupos de acciones. Un **plan válido**, $\Pi = <\pi_1, \ldots, \pi_n>$, es la solución al problema si y sólo si:
  - Cada conjunto $\pi_i \in \Pi$ es independiente
  - π_n es aplicable a s_n

planning graph explora los resultados de todas las posibles acciones a nivel n:
buscar un plan válido en los n pasos, el plan es un subgrafo del planning graph

Exclusión Mutua

- No se puede tener 2 acciones simultáneas en un nivel que dependen
- Dos acciones en un determinado nivel en planning graph son mutuamente excluyentes ("mutex") si no hay un plan válido que contiene a los 2, o ningún plan podría hacer ambas verdaderas, es decir: que dependen o tienen condiciones incompatibles
  μA_i = acciones que se excluyen mutuamente en el nivel i
  μP_i = proposiciones mutuamente excluyentes en el nivel i

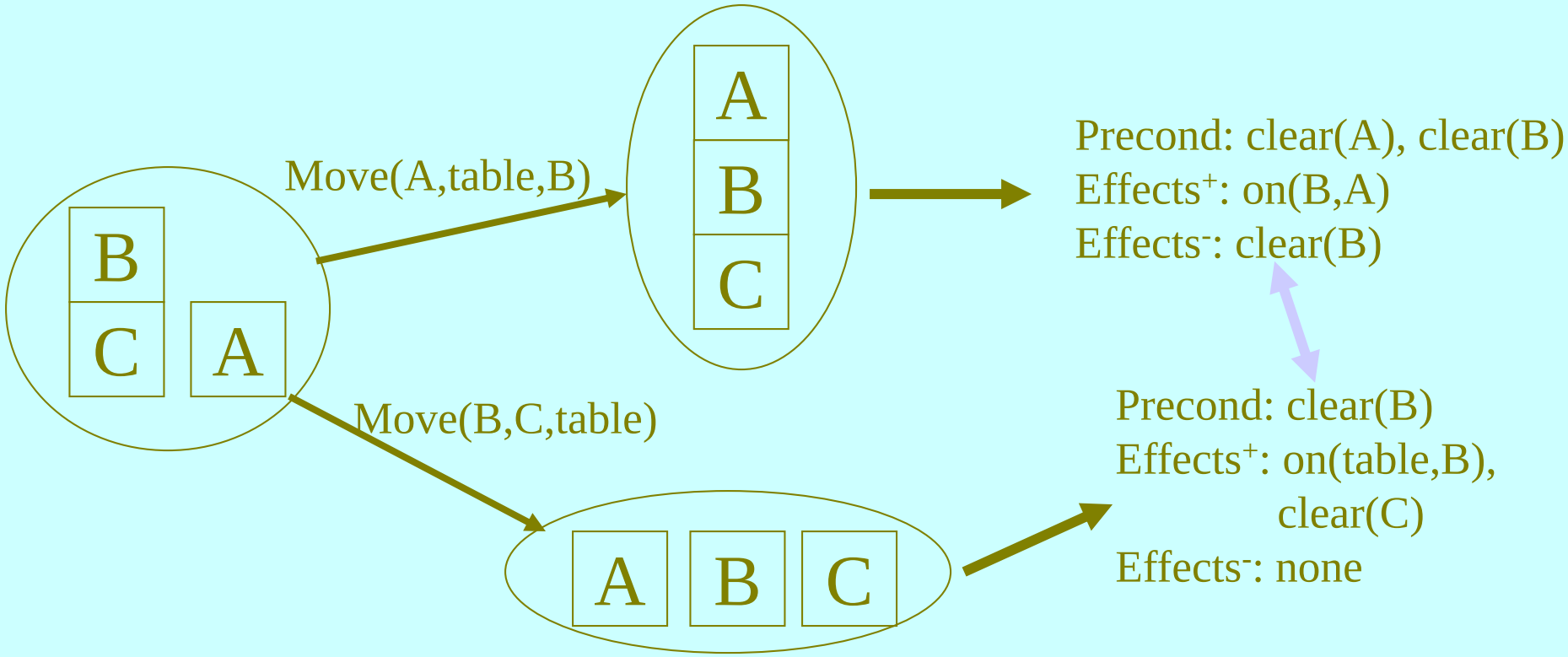
Exclusión Mutua

Dos reglas:

1. Interferencia: si una acción elimina una precondición de otro o elimina un efecto positivo
2. Opuestas necesidades: si las acciones A y B tienen precondiciones que están marcados como exclusión mutua en el nivel de la proposición anterior

Ejemplo de xclusión Mutua

- Interferencia



Mutex Example

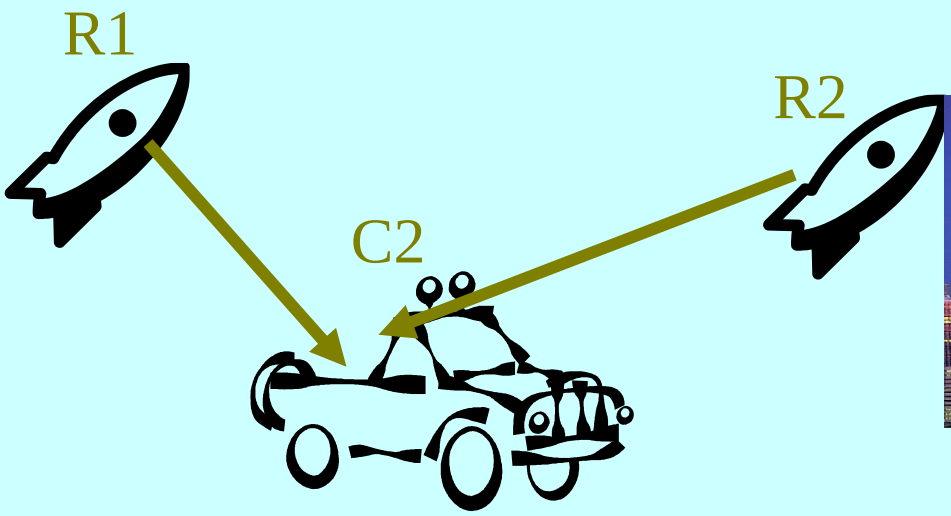
- Opuestas necesidades:

A) Load(R1, C2, St Louis)

B) Load(R2, C2, Seattle)

→Mutex porque C2 no puede estar en St Louis y Seattle en la misma hora

St. Louis



Seattle

Graphplan Algorithm

- Input: Proposition level P_0 containing initial conditions
- Output: valid plan or states no valid plan exists
- Algorithm:

```
while (!done)
{
    Expansion Phase: Expand planning graph to next action and
                     proposition level;
    Search/Extraction Phase: Search graph for a valid plan;
    if (valid plan exists)
        return successful plan;
    else
        continue;
}
```

→ Graphplan es sólido y completo

Graphplan Algorithm

```
Graphplan(A, s0, g)
    i ← 0,    ∇ ← ∅ ,    P0 ← s0
    G ← ⟨P0⟩
    until [g ⊆ Pi and g² ∩ μPi = ∅] or Fixedpoint(G) do
        i ← i + 1
        G ← Expand(G)
    if g ⊈ Pi or g² ∩ μPi ≠ ∅ then return(failure)
    Π ← Extract(G, g, i)
    if Fixedpoint(G) then η ← |∇(κ)|
    else η ← 0
    while Π = failure do
        i ← i + 1
        G ← Expand(G)
        Π ← Extract(G, g, i)
        if Π = failure and Fixedpoint(G) then
            if η = |∇(κ)| then return(failure)
            η ← |∇(κ)|
    return(Π)
end
```

Expansion Algorithm

Expand($\langle P_0, A_1, \mu A_1, P_1, \mu P_1, \ldots, A_{i-1}, \mu A_{i-1}, P_{i-1}, \mu P_{i-1} \rangle$)

$A_i \leftarrow \{a \in A \mid \text{precond}(a) \subseteq P_{i-1} \text{ and } \text{precond}^2(a) \cap \mu P_{i-1} = \emptyset\}$

$P_i \leftarrow \{p \mid \exists a \in A_i : p \in \text{effects}^+(a)\}$

$\mu A_i \leftarrow \{(a, b) \in A_i^2, a \neq b \mid \text{effects}^-(a) \cap [\text{precond}(b) \cup \text{effects}^+(b)] \neq \emptyset$
or $\text{effects}^-(b) \cap [\text{precond}(a) \cup \text{effects}^+(a)] \neq \emptyset$
or $\exists (p, q) \in \mu P_{i-1} : p \in \text{precond}(a), q \in \text{precond}(b)\}$

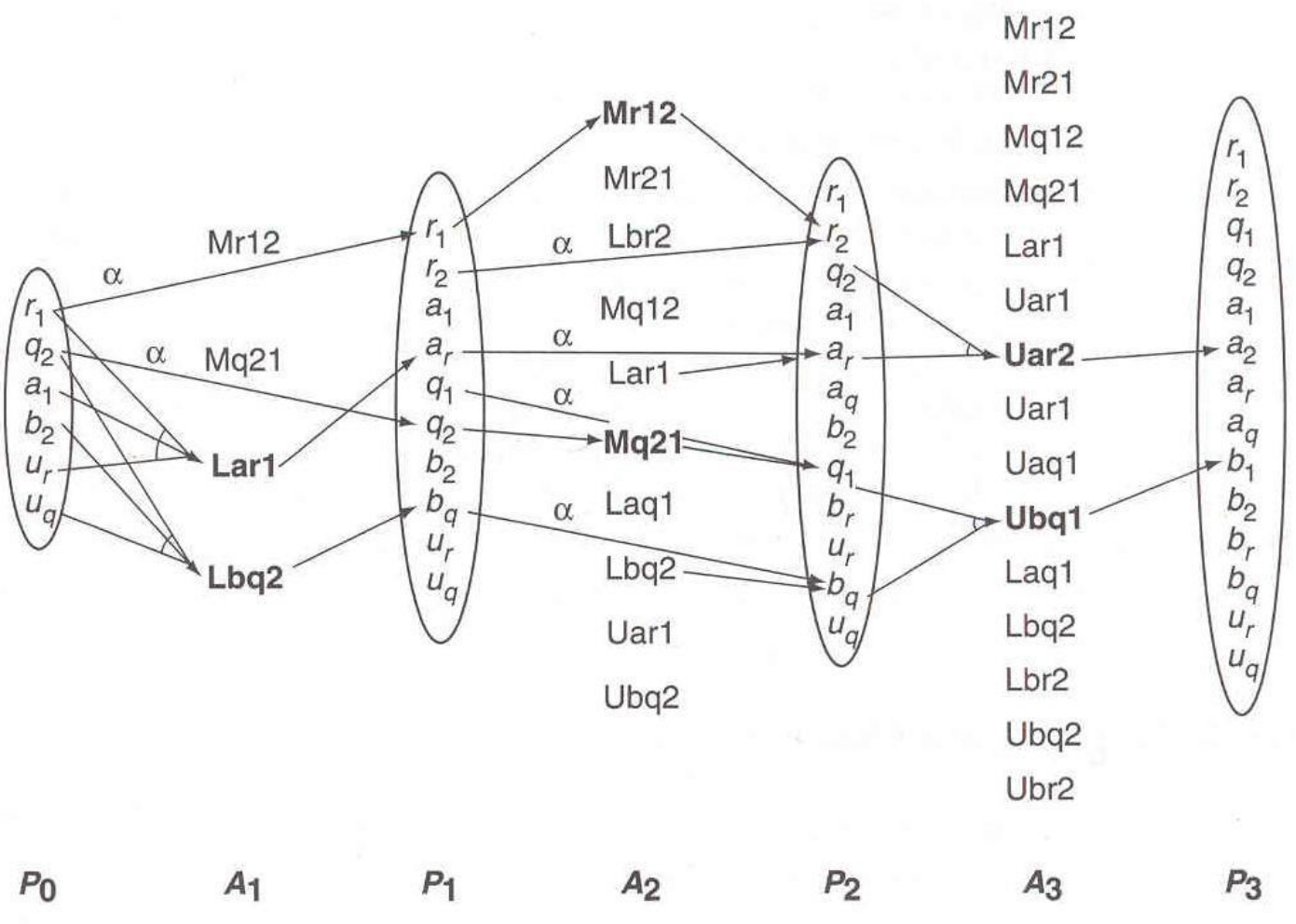
$\mu P_i \leftarrow \{(p, q) \in P_i^2, p \neq q \mid \forall a, b \in A_i, a \neq b :$
$p \in \text{effects}^+(a), q \in \text{effects}^+(b) \Rightarrow (a, b) \in \mu A_i\}$

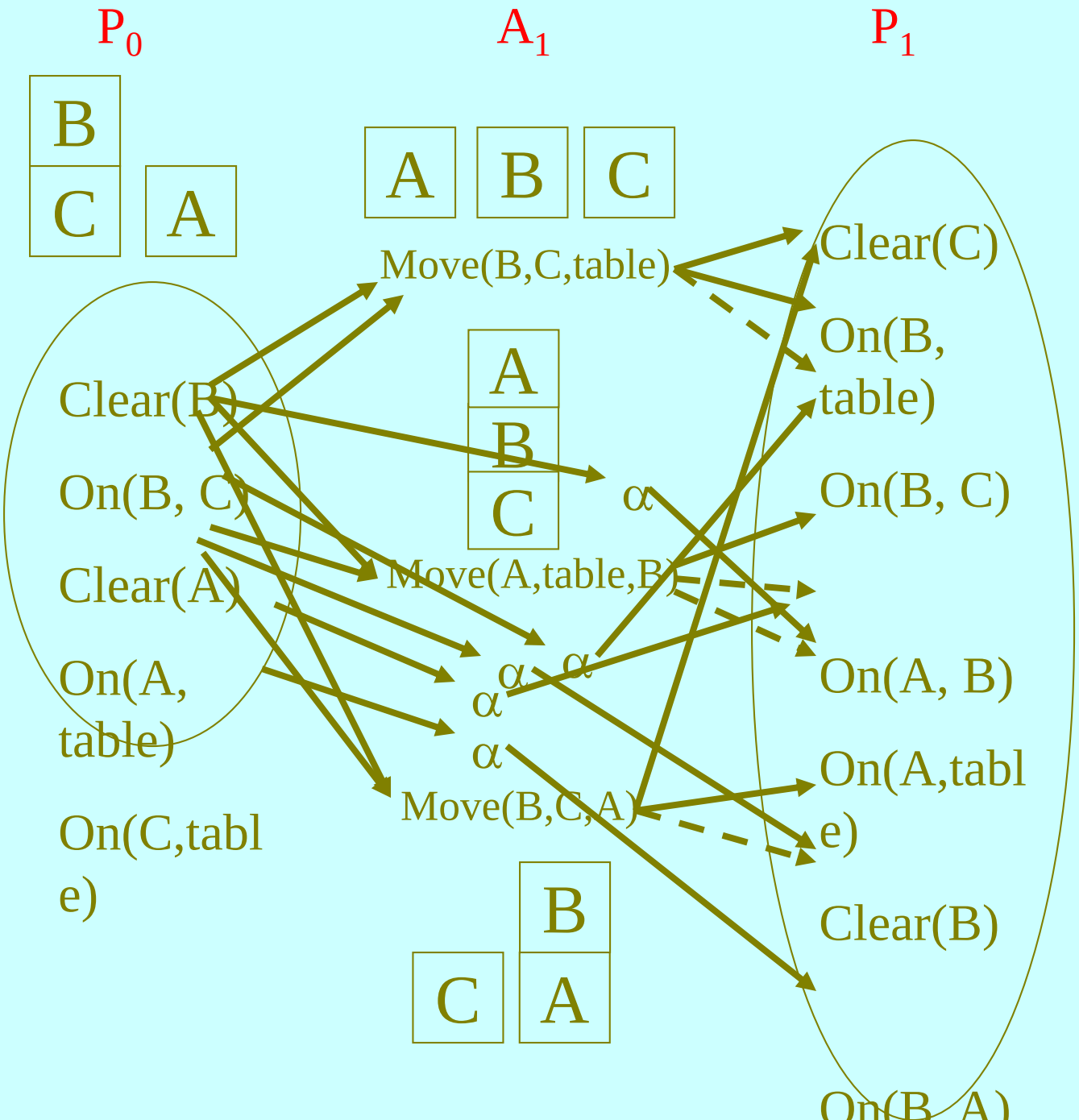
for each $a \in A_i$ do: link a with precondition arcs to $\text{precond}(a)$ in P_{i-1}
positive arcs to $\text{effects}^+(a)$ and negative arcs to $\text{effects}^-(a)$ in P_i

return($\langle P_0, A_1, \mu A_1, \ldots, P_{i-1}, \mu P_{i-1}, A_i, \mu A_i, P_i, \mu P_i \rangle$)

end

Planning Graph





Mutex list for Move(B,C,table):
-Move(A,table,B)
-Move(B,C,A)

Mutex list for Move(A,table,B):
-Move(B,C,table)
-Move(B,C,A)

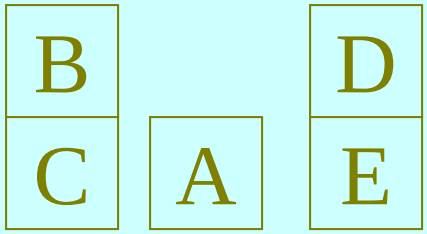
Mutex list for Move(B,C,A):
-Move(B,C,table)
-Move(A,table,B)

Extraction Algorithm

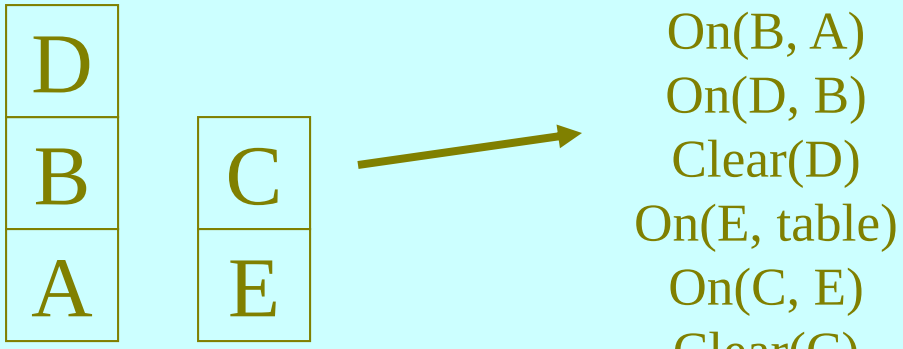
Extract(G, g, i)
    if $i = 0$ then return ($\langle\rangle$)
    if $g \in \nabla(i)$ then return(failure)
    $\pi_i \leftarrow$ GP-Search($G, g, \emptyset, i$)
    if $\pi_i \neq$ failure then return(π_i)
    $\nabla(i) \leftarrow \nabla(i) \cup \{g\}$
    return(failure)
end

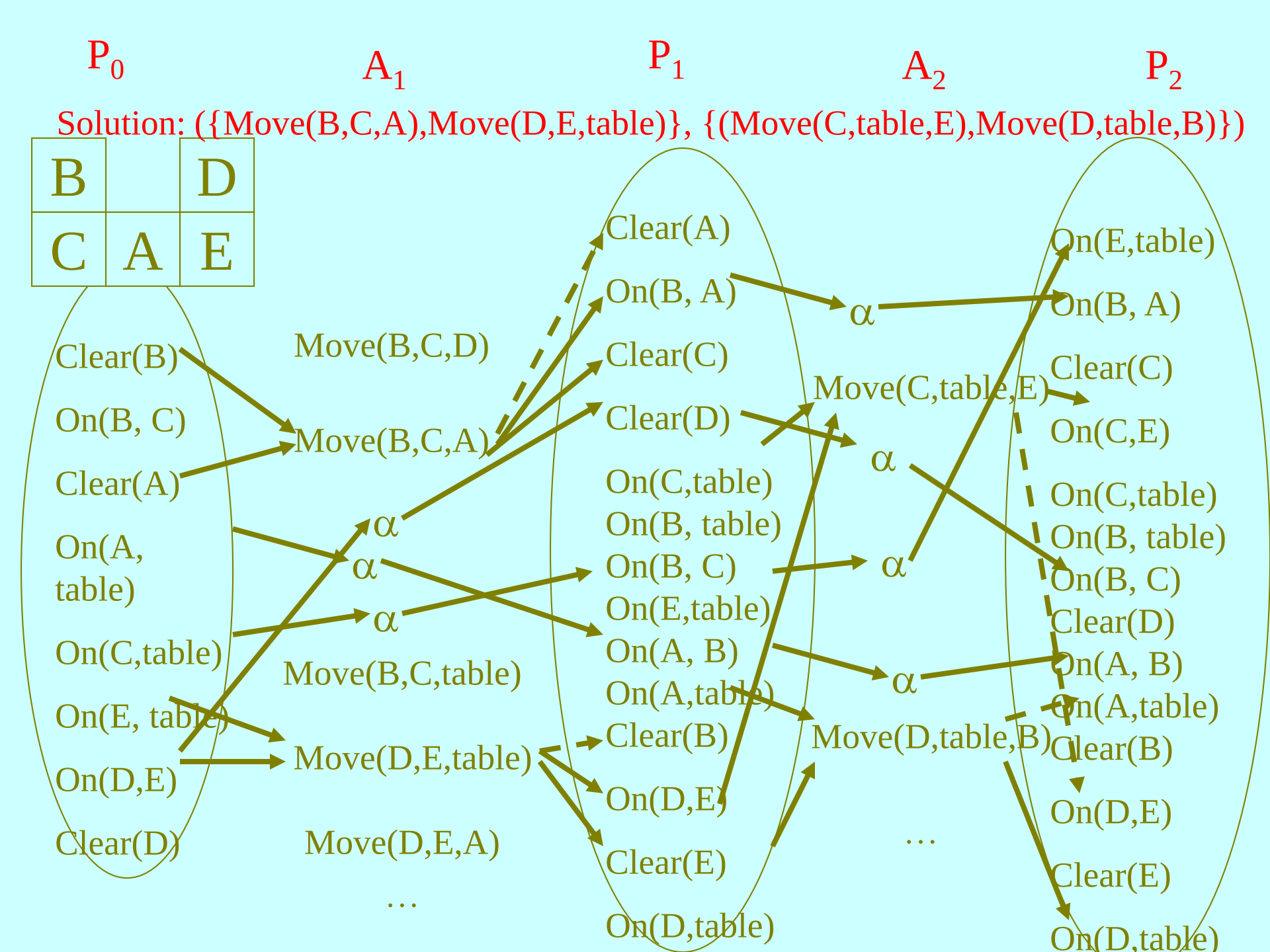
Algorithm Example

- Initial state:



- Goal state:




P0
A1
P1
A2
P2
Solution: ({Move(B,C,A),Move(D,E,table)}, {(Move(C,table,E),Move(D,table,B)})
B
D
C
A
E
Clear(B)
On(B, C)
Clear(A)
On(A, table)
On(C,table)
On(E, table)
On(D,E)
Clear(D)
Move(B,C,D)
Move(B,C,A)
α
α
α
Move(B,C,table)
Move(D,E,table)
Move(D,E,A)
…
Clear(A)
On(B, A)
Clear(C)
Clear(D)
On(C,table)
On(B, table)
On(B, C)
On(E,table)
On(A, B)
On(A,table)
Clear(B)
On(D,E)
Clear(E)
On(D,table)
α
Move(C,table,E)
α
α
α
Move(D,table,B)
…
On(E,table)
On(B, A)
Clear(C)
On(C,E)
On(C,table)
On(B, table)
On(B, C)
Clear(D)
On(A, B)
On(A,table)
Clear(B)
On(D,E)
Clear(E)
On(D,table)

Problemas abiertos

la cognición, la planificación y la ejecución del plan deben todos ser intercalados. El curso de c/u afecta al de los otros.

- Un agente racional debe realizar todas estas tareas en paralelo

En IA a veces se separa la planificación y la programación

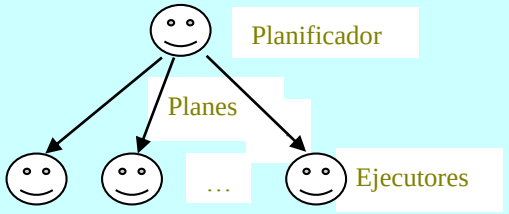
- Primero se adopta un plan, y luego se construyen un programa para cuándo se ejecute el plan
- Esto parece inadecuada, incluso en los humanos.

Otras Técnicas de Planificación en SMA

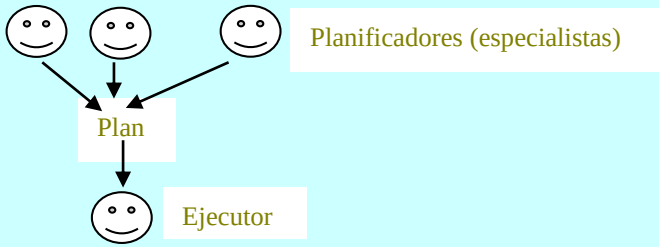
- Planificación Centralizada para Planes Distribuidos
- Planificación Distribuida para Planes Centralizados
- Planificación Distribuida para Planes Distribuidos

Otras Técnicas de Planificación en SMA

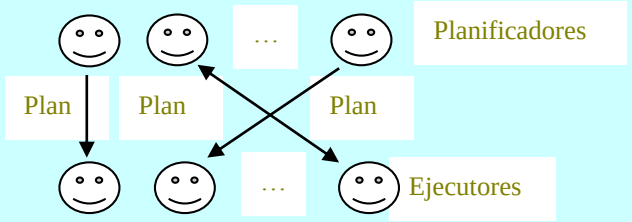
- Planificación Centralizada para Planes Distribuidos



- Planificación Distribuida para Planes Centralizados



- Planificación Distribuida para Planes Distribuidos



Planificación Centralizada para Planes Distribuidos

- Esquema:
  - Planificación en orden parcial que puede ser ejecutada en paralelo (diferentes agentes hacen ramas diferentes).
  - Se debe garantizar no condiciones de dependencia.

- Algoritmo:
  1. realización de un plan general (el planificador)
  2. identificación ramificaciones que se pueden realizar en paralelo (POP).
  3. con ellas descomponer el plan en subplanes (ramificaciones)
  4. insertar acciones de sincronización entre los subplanes
  5. asignar subplanes a los agentes (asignacion (estatica o dinamica) a los agentes)

Planificación Distribuida para Planes Centralizados

Formular planes requiere colaboración de especialistas

1. Distribuir la tarea de construir planes entre los especialistas
2. Mezclar los planes obtenidos en paralelos verificando la coherencia entre ellos

Debe haber una coordinación centralizada para unir los planes parciales

- CADA AGENTE HACE SU PLAN PARCIAL
- EL COORDINADOR FUSIONA LOS PLANES (RELACION ENTRE LAS ACCIONES)

Planificación Distribuida para Planes Distribuidos

- Caso General
  1. Asignar objetivos a cada agente
  2. C/u formula planes locales
  3. C/u ejecuta sus planes

- Problemas
  - Resolución de Conflictos: negociación
  - Suposiciones de tiempo de ejecución de tareas

Planificación Distribuida para Planes Distribuidos

- Otro Algoritmo:
  1. Intercambiar entre agentes descripción de los planes
  2. Aprobar planes sin conflictos (si todos lo cumplen, resuelto el problema)
  3. Refinar planes con conflictos (cambios en los planes locales para eliminarlos)
  4. Ejecutar localmente planes sin conflicto

Proyecto

- Planificador general (cuerpo principal determinista)
  - Organizar las tareas que debe realizar el agente
  - Definir secuencia de realización de las tareas

- Planificador como tarea (dinámico)
  - Definir problema de planificación
  - Describir vocabulario de condiciones, operadores, y objetos
  - Establecer planificador (usar Graphplan)