# Planning

Inteligencia Artificial

Prof. Dra. Silvia Schiaffino

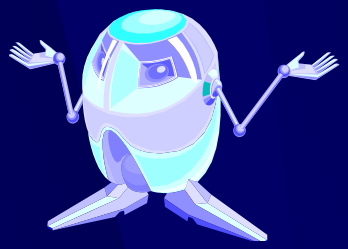
ISISTAN - UNCPBA

# Agenda

- Problema de Planning
- Tipos de algoritmos
- Planning de Orden Parcial
- Algoritmo POP
- Algoritmo UCPOP
  - Ejemplo

# Planning en agentes

- Necesidad de actuar en el mundo
  - Agentes Racionales
  - Agentes Deliberativos



# El problema de Planning

Formalmente, posee tres entradas

- Una descripción del mundo

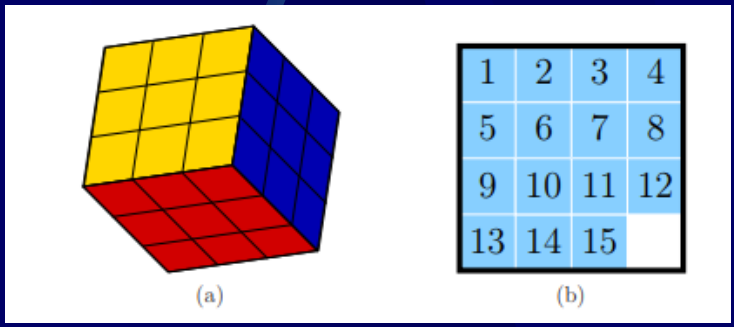
- Una descripción de las metas

- Una descripción de las posibles acciones que pueden ser realizadas

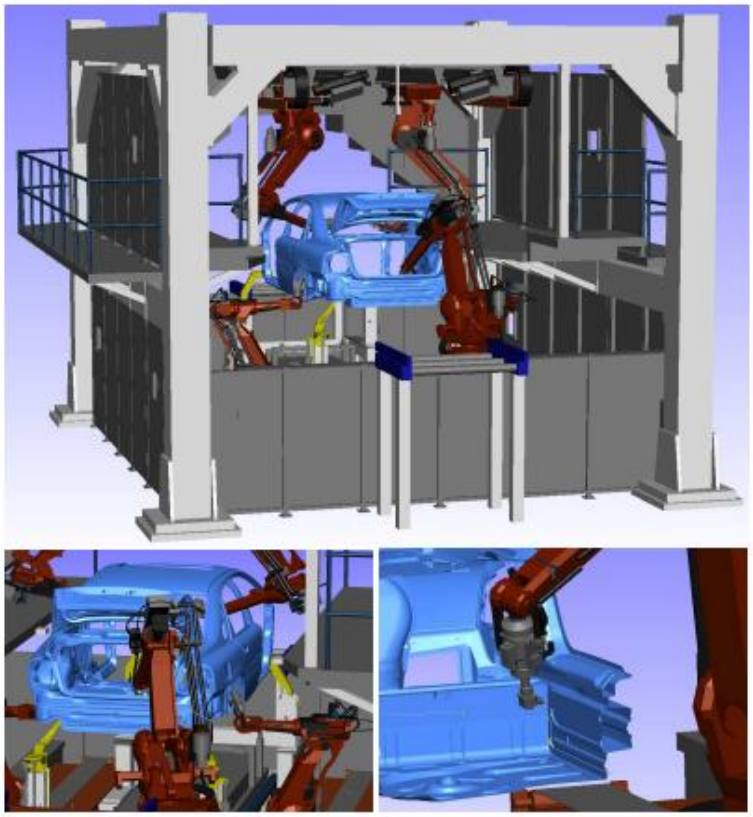
# Problema de Planning

Salida del problema

- Una secuencia de acciones que cuando son ejecutadas en un mundo que satisface las condiciones iniciales alcanzará las metas.



# Ejemplos de aplicación



# Ejemplos de aplicación

(a)

(b)

# Simplificaciones

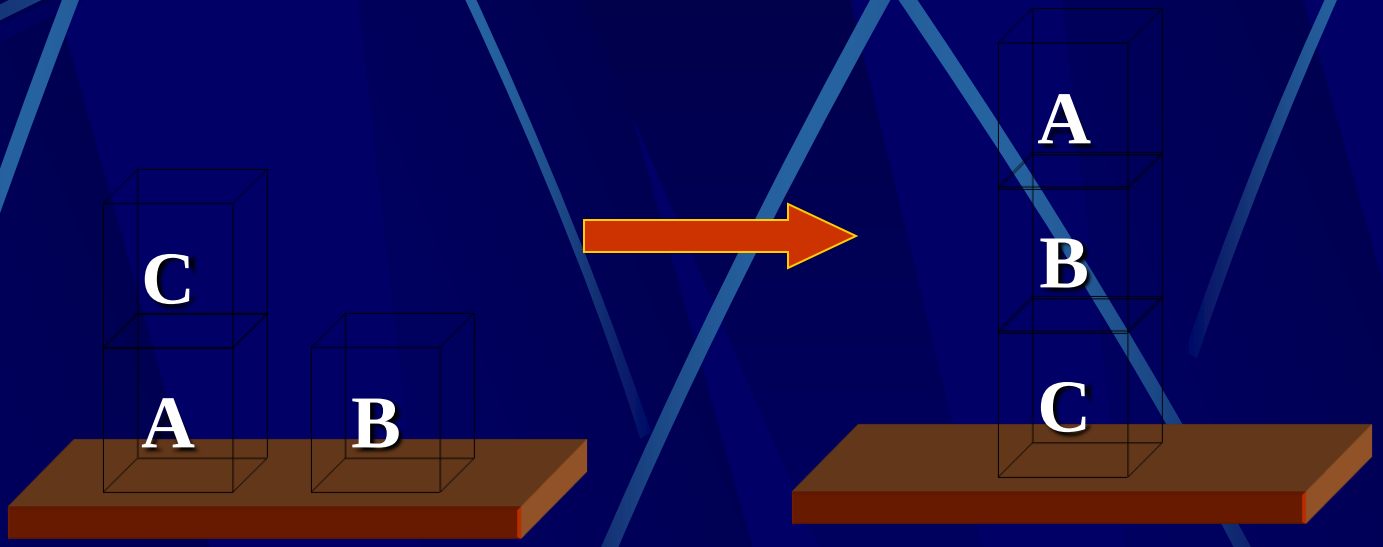
- Tiempo atómico: acciones en un instante dado; tiempo indivisible
- Efectos determinísticos
- Omnisciencia: conocimiento del estado del mundo y naturaleza de acciones
- Única causa de cambio: acciones

# Lenguaje de representación

Lenguaje de representación proposicional STRIPS (STanford Reserch Institute Problem Solver)

Se describe el estado inicial del mundo mediante un conjunto de literales “ground”

# Lenguaje de representación



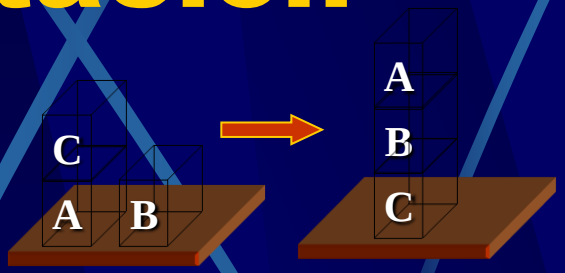

Estado Inicial

on(A, Table) on(C, A) on(B, Table) clear(B) clear(C)

# Lenguaje de representación

Estado Final

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




Acciones:

Se definen mediante pre y post condiciones

move-C-from-A-to-Table

*Pre*: on(C, A), clear(C)

*Post:* on(C, Table), not(on(C, A)), clear (A)

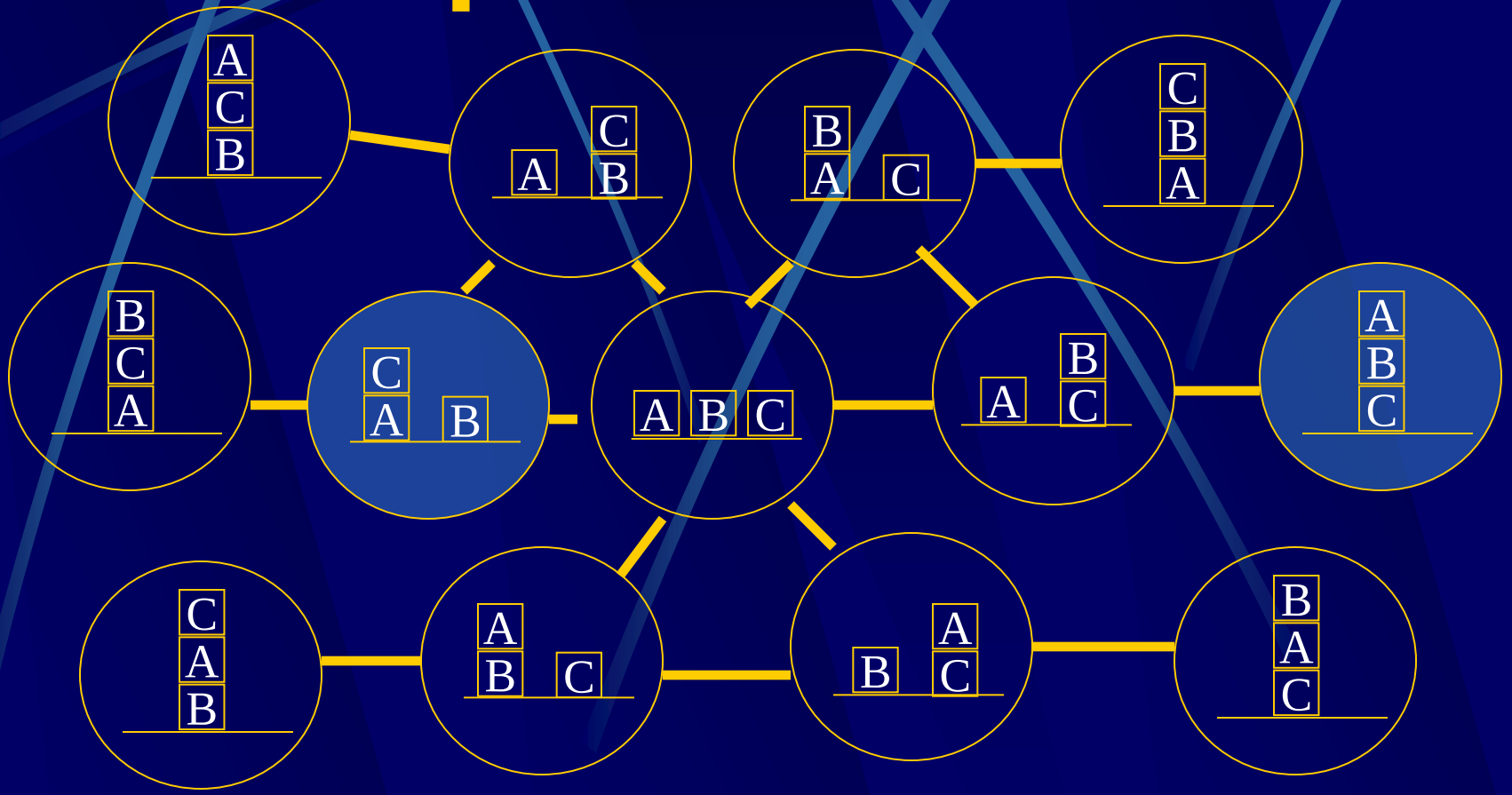
# Agenda

- Problema de Planning
- Tipos de algoritmos
- Planning de Orden Parcial
- Algoritmo POP
- Algoritmo UCPOP

# Tipos de algoritmos

- Buscar a través del espacio del mundo
  - Progresión (hacia adelante)
  - Regresión (hacia atrás)
- Bucar a través del espacio de planes
  - De orden total
  - De orden parcial

# Búsqueda a través del espacio del mundo



# Progresión

**Algorithm:** PROGWS(world-state, goal-list, $\Lambda$, path)

1. If world-state satisfies each conjunct in goal-list,
2. Then return path,
3. Else let Act = **choose** from $\Lambda$ an action whose precondition is satisfied by world-state:
   (a) If no such choice was possible,
   (b) Then return failure,
   (c) Else let S = the result of simulating execution of Act in world-state and return PROGWS(S, goal-list, $\Lambda$, Concatenate(path, Act)).

# Regresión

**Algorithm:** REGWS(init-state, cur-goals, $\Lambda$, path)

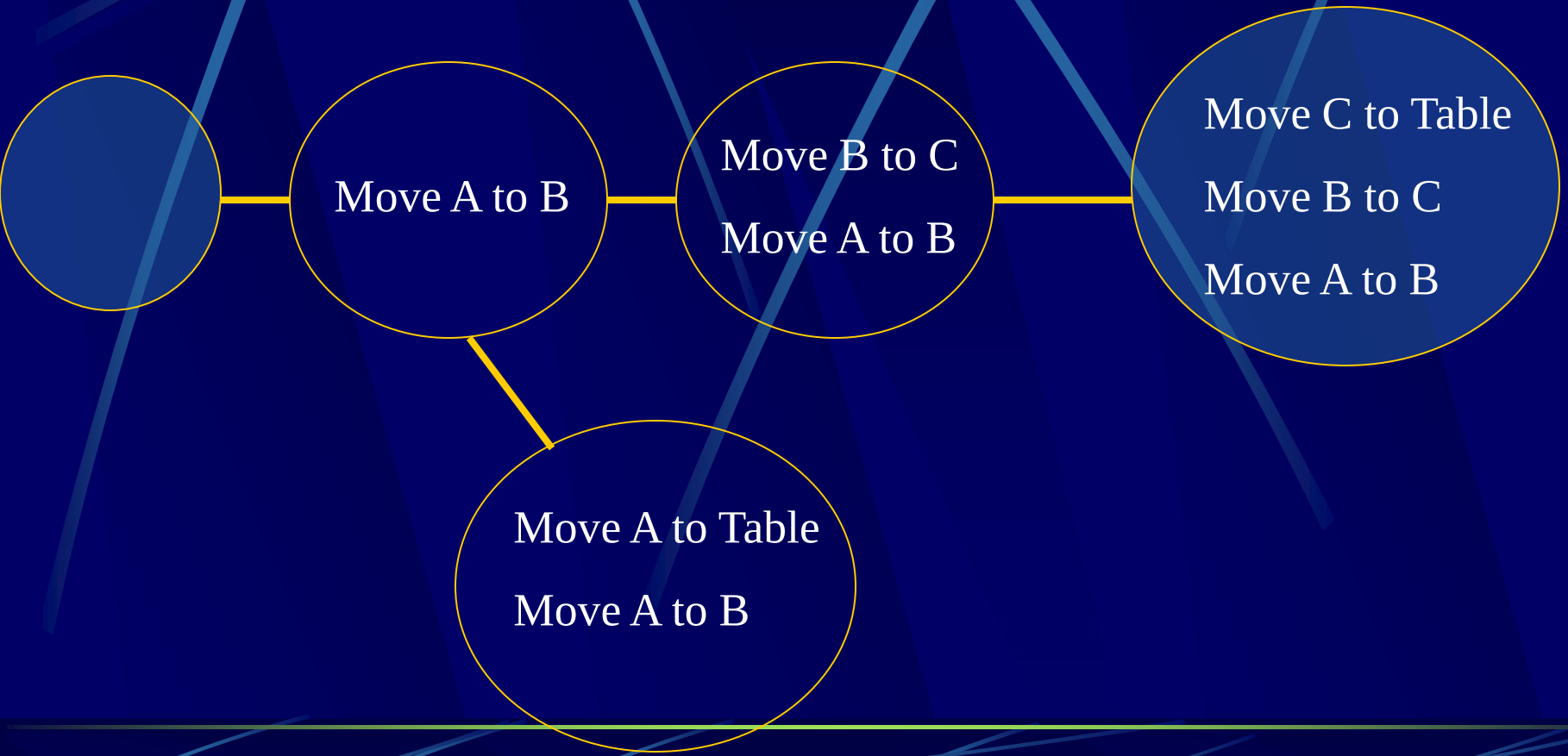
1. If **init-state** satisifes each conjunct in **cur-goals**,
2. Then return **path**,
3. Else do:
   (a) Let **Act** = **choose** from $\Lambda$ an action whose effect matches at least one conjunct in **cur-goals**.
   (b) Let G = the result of regressing `cur-goals` through `Act`.
   (c) If no choice for **Act** was possible or G is undefined, or G $\supset$ **cur-goals**,
   (d) Then return failure,
   (e) Else return REGWS(init-state, G, $\Lambda$, Concatenate(Act, path)).

# Buscar a través del espacio de planes

Los nodos representan planes parcialmente especificados

Los arcos representan operaciones de refinamiento, como el agregado de una acción al plan

# Buscar a través del espacio de planes



# Agenda

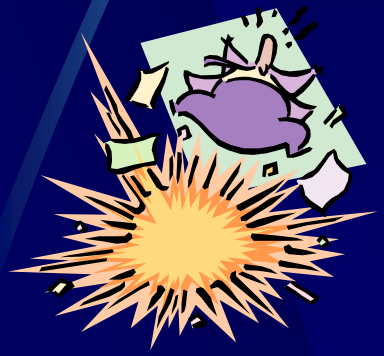
- Problema de Planning
- Tipos de algoritmos
- Planning de Orden Parcial
- Algoritmo POP
- Algoritmo UCPOP

# Planning de orden parcial

Los planes son representados como una secuencia parcialmente ordenada de acciones. Sólo las decisiones de orden esenciales son recordadas

# Representación

- Planes
- Links Causales
- Amenazas



# Planes

Los planes se representan como una tres- upla < ***A*** , ***O, L*** >

***A :*** conjunto de acciones

***O :*** conjunto de restricciones de orden sobre las acciones

***L :*** conjunto de links causales

# Links causales

Mantienen una pista de las decisiones realizadas

Dejan un registro explícito de las dependencias entre las acciones

Ap $\xrightarrow{Q}$ Ac

# Amenazas

Si $A_p \xrightarrow{Q} A_c$ pertenece a ***L,*** sea $A_t$ una acción diferente se dice que $A_t$ amenaza al link si:

***O*** U { $A_p < A_t < A_c$} es consistente

$A_t$ tiene como efecto a ¬Q

# Amenaza: Ejemplo

Ap tiene como efecto Q = (on A B), que es una precondición de Ac

At sería considerada una amenaza si saca a A de arriba de B, y el orden de las acciones no evita que At se ejecute entre Ap y Ac

# Amenazas

Resolución de una amenaza:

Democión: At < Ap

Promoción: Ac < At

# Representación

Un aspecto clave en estos algoritmos es mantener la uniformidad

Se representa de igual manera a planes completos que incompletos o nulos

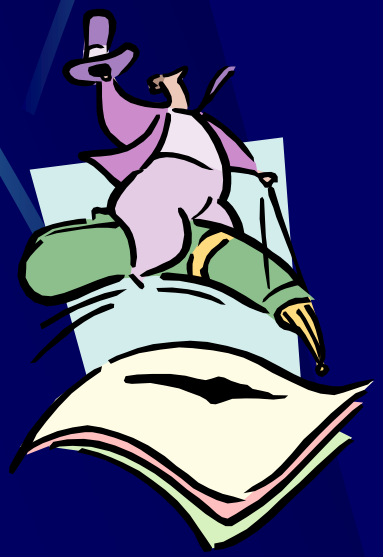
Se representa al estado inicial y a las metas como una tres-upla  llamada plan nulo

# Plan inicial

$\boldsymbol{A} = \{ A_0, A\infty \}$

$\boldsymbol{O} = \{ A_0 < A\infty \}$

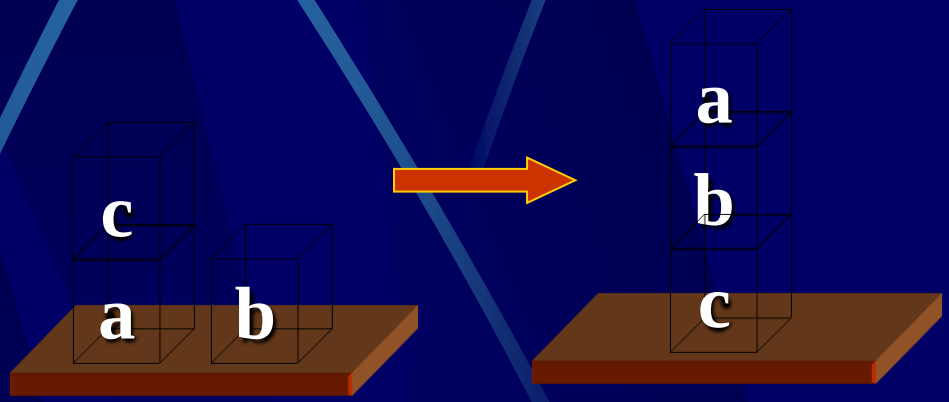
$\boldsymbol{L} = \{ \}$



# Ejemplo

*Start*

on(c,a) on(a, table) on(b,table) clear(c) clear(b)




on(a, b)  on(b,c)

*End*

# Agenda

- Problema de Planning
- Tipos de algoritmos
- Algoritmo POP (Partial Order Planner)
- Algoritmo UCPOP

# Algoritmo POP ( < *A, O, L* >, agenda, Ac )

1. Terminación: si la agenda está vacía devolver <A,O,L>
2. Selección de Objetivo: tomar un par <Q, $A_{need}$>
3. Selección de Acción: $A_{add}$ = elegir una acción que satisfaga Q. Si no existe la acción devolver falso

   $L' = L \cup \{ A_{add} \xrightarrow{Q} A_{need} \}$

   $O' = O \cup \{A_{add} < A_{need}\}$

   Si $A_{add}$ es una nueva acción $A' = A \cup \{A_{add}\}$

4. Actualización de Objetivos:

   agenda' = agenda – { <Q,$A_{need}$>}

   Si la acción era nueva se agregan las precondiciones de la misma a la agenda

# Algoritmo POP (cont.)

5. Protección de los links causales: por cada acción At que pueda amenazar a un link

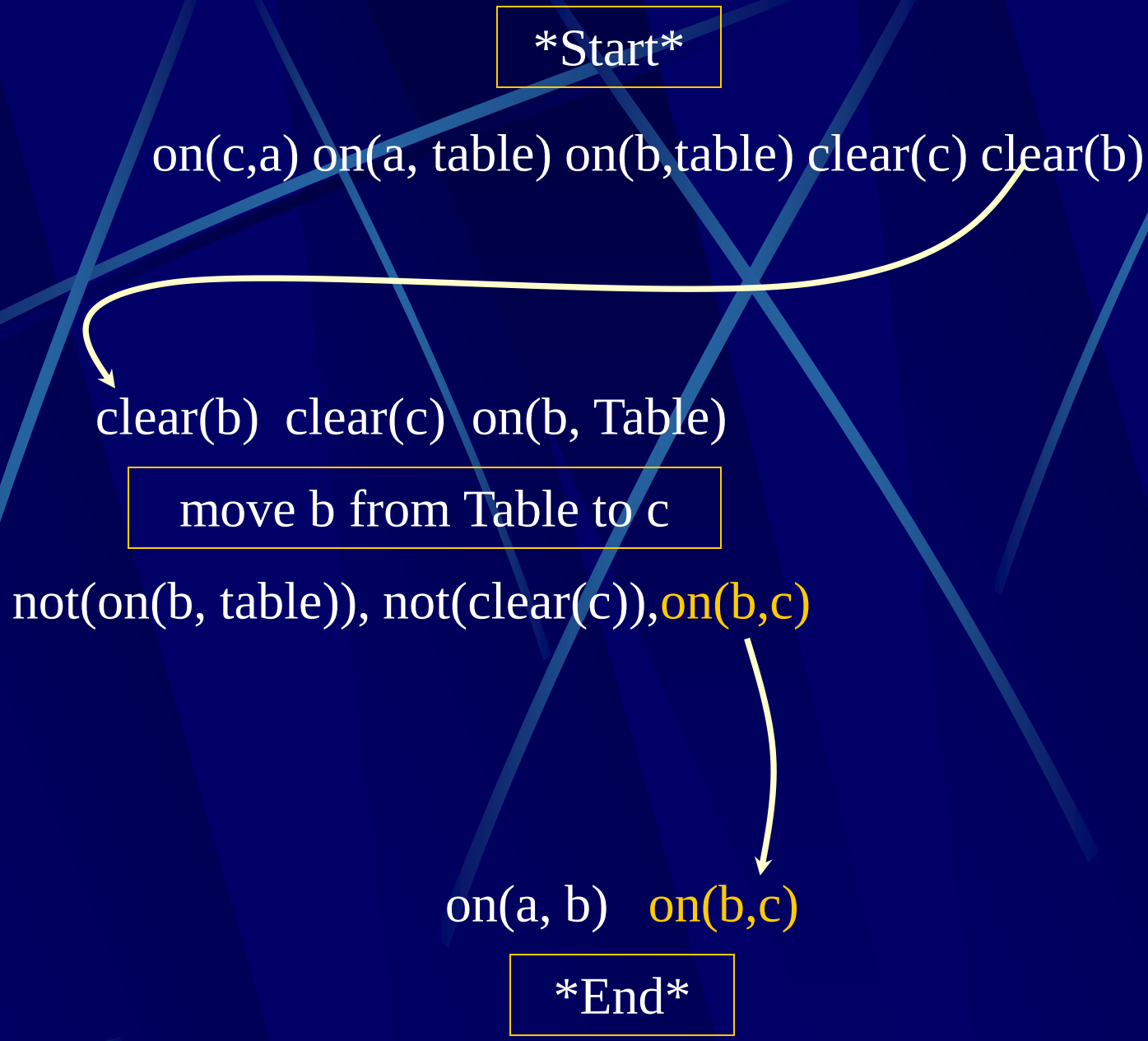
$A_p \xrightarrow{R} A_c$ elegir un orden consistente entre:

(a) Democión: Agregar At < Ap a ***O'***

(b) Promoción: Agregar Ac < At a ***O'***

Si ninguna de las dos es posible devolver falso

6. Invocación recursiva: POP ***(<A',O',L' >,*** agenda' A)


*Start*
on(c,a) on(a, table) on(b,table) clear(c) clear(b)
clear(b) clear(c) on(b, Table)
move b from Table to c
not(on(b, table)), not(clear(c)),on(b,c)
on(a, b) on(b,c)
*End*

*Start*

on(c,a) on(a, table) on(b,table) clear(c) clear(b)

clear(a) clear(b) on(a, Table)

move a from Table to b

not(on(a, table)), not(clear(b)),on(a,b)

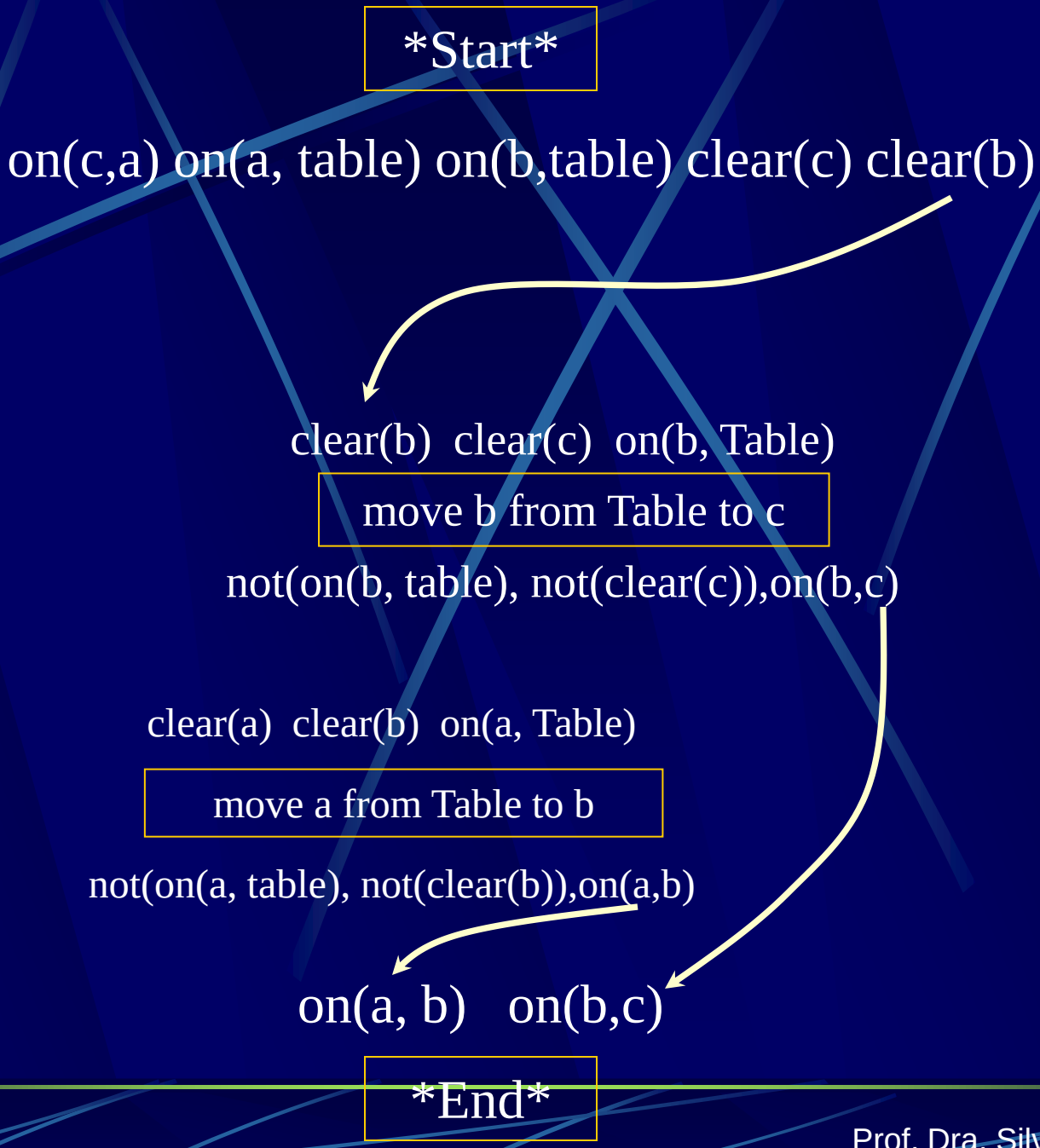
clear(b) clear(c) on(b, Table)

move b from Table to c

not(on(b, table)), not(clear(c)),on(b,c)

on(a, b) on(b,c)

*End*


*Start*
on(c,a) on(a, table) on(b,table) clear(c) clear(b)
clear(b) clear(c) on(b, Table)
move b from Table to c
not(on(b, table), not(clear(c)),on(b,c)
clear(a) clear(b) on(a, Table)
move a from Table to b
not(on(a, table), not(clear(b)),on(a,b)
on(a, b) on(b,c)
*End*

# Agenda

- Problema de Planning
- Tipos de algoritmos
- Algoritmo POP
- Algoritmo UCPOP

# Algoritmo UCPOP

- Extensión del POP con:
  - Manejo de Variables
  - Precondiciones disyuntivas
  - Cuantificación universal
  - Efectos condicionales

# Agregado de Variables

- No sólo facilitan la creación de acciones, sino que postergan decisiones
- Se permiten restricciones sobres las variables

move(X,Y,Z) :
Condiciones: notEqual(X,Z), notEqual(X,Y), notEqual(Y,Z)
notEqual(Z,table).
Pre-Condiciones: clear(X), clear(Z), on(X,Y)
Efectos: on(X,Z), not(on(X,Y)), not (clear(Z)) clear(Y)

# Agregado de variables

- Beneficios de representación
  - Una representación uniforme de las acciones
  - Menor cantidad de espacio utilizado

# Modificaciones necesarias al POP

- Aparece una tabla de variables
  - <A, O, L, B>, B={} al inicio
  - Permitir unificación
- La selección de acciones debe poder elegir entre acciones existentes o esquemas de acciones que satisfagan el objetivo
- Si se agrega un nuevo link causal se agregan también los cambios en la tabla de variables

# Modificaciones

- Protección de los links causales
  - not (clear (?X)) amenaza link con clear(B) ??
- Asegurarse que se hayan instanciado todas las variables al devolver el plan.

# Efectos condicionales

- Surge una cláusula when que posee un antecedente y un consecuente

“ la acción tendrá el efecto del consecuente solo si antes de su ejecución el antecedente era verdadero”

# Ejemplo

move(X,Y,Z) :

Condiciones: notEqual(X,Z), notEqual(X,Y), notEqual(Y,Z)

Pre-Condiciones: clear(X), clear(Z), on(X,Y)

Efectos: on(X,Z), not(on(X,Y)), clear(Y)

when( (notEqual(Z,Table)), not(clear(Z)) )

# Modificaciones a POP

- Si se selecciona una acción que satisfaga a Q en el consecuente de su condición, debe agregarse el antecedente a la agenda
- Protección de los links, surge confrontación: agrego el antecedente del when negado en la agenda, para asegurarme que no se cumpla, si el consecuente del when hace peligrar un link

# Precondiciones disyuntivas

- Sólo las precondiciones pueden poseer disyunciones
- Deben usarse con moderación

  or (clear(a), equal(a,table) )
- Al detectar un objetivo disyuntivo a la agenda se elige uno de los objetivos y se agrega a la agenda.

# Cuantificación universal

- Aumenta el poder de expresión del planning
- forAll (block(X), ( clear(X) , on(X,table ))

# Simplificaciones

- Se asume que el mundo modelado tiene un universo de objetos estático y finito.
- Cada objeto tiene un tipo

# Base universal

- Mapear las fórmulas con cuantificadores a cláusulas ground.

$f(\Delta) = \Delta$ ..Si $\Delta$ no tiene cuantificadores

$f(\forall_{t1}\ x\ \Delta(x)) = f(\Delta_1),\ldots f(\Delta_n)$

donde $\Delta_i$ corresponde a cada $\Delta(x)$

# Base universal
# Ejemplo

block (a), block(b)

forAll(block(X), [clear(X),on(X,table)])

clear(a), on(a,table) , clear(b) , on(b,table)

# En UCPOP...

- Consideraciones
  - Si un objetivo o precondición es una sentencia universalmente cuantificada, se calcula su base y se planea sobre ella
  - Si un efecto involucra cuantificación universal, no se calcula la base inmediatamente. Se genera a medida que se la asocia mediante links
  - Se cambia la definición de amenazas para tratar con efectos cuantificados

# Definición de amenazas

- Si una acción posee un efecto cuantificado, se debe controlar que el mismo no amenace al link. Lo nuevo es que se permite que existan variables sobre la base.

- Se agrega un nuevo método de resolución, confrontación.

# UCPOP (<***A,O,L,B***>,agenda, acciones)

1 Terminación: Si la agenda está vacía <A,O,L,B>

2 Reducción de objetivos: tomar <$Q_i$,$A_c$> de la agenda

1. Si $Q_i$ está cuantificada calcular la base, agregarla a la agenda, ir al paso 2
2. Si Qi es una conjunción agregar cada elemento en la agenda, ir al paso 3
3. Si Qi es una disyunción elegir un componente, agregarlo a la agenda, ir al paso 3
4. Si Q es un literal y existe un link con ¬Q entre Ap y Ac. Fallar, no es un plan posible

3 Selección de la acción: elegir una acción Ap, existente o no, que sea consistente con ***O*** y que posea un efecto R que unifique con Q. Si R pertenece a un consecuente, agregar el antecedente a la agenda.

Actualizar ***L, O, B*** **.**

4 Habilitar nuevas acciones y efectos: Se actualiza ***A*** y la agenda

5 Protección de los links causales: por cada amenaza elegir

Promoción

Democión

Confrontación

6 Invocación Recursiva

# Agenda

- Problema de Planning
- Tipos de algoritmos
- Planning de Orden Parcial
- Algoritmo POP
- Algoritmo UCPOP
  - Ejemplo: Problema de los cohetes

# El Problema de los Cohetes

- Una carga puede ser colocada en un cohete, si dicha carga y el cohete están en el mismo lugar.
- Un cohete se mueve si tiene combustible.
- Acciones:
  - **move(Rocket,From,To)**
    *preconditions:* ¬equal(From,To), at(From,Rocket), hasFuel(Rocket)
    *effects:* at(To,Rocket), ¬(at(From,Rocket), ¬hasFuel(Rocket)
  - **unload(Rocket,Place,Charge)**
    *preconditions:* at(Place,Rocket), in(Rocket,Charge)
    *effects:* ¬in(Rocket,Charge), at(Place,Charge)
  - **load(Rocket,Place,Charge)**
    *preconditions:* at(Place,Rocket), at(Place,Charge)
    *effects:* ¬at(Place,Charge), in(Rocket,Charge)

# Situación Inicial en UCPOP

**INICIAL**

**at(a,r1), at(a,r2),**
**at(a,c1), at(a,c2)**
**hasfuel(r1), hasfuel(r2)**

**OBJETIVO**

**at(b,c1), at(b,c2)**

* start (Ao) *

**at(a,r1) at(a,r2) at(a,c1) at(a,c2)**
**hasfuel(r1) hasfuel(r2)**

**at(b,c1) at(b,c2)**

* end (A∞)*

A = { Ao , A∞ }

O = { Ao < A∞ }

L = {}

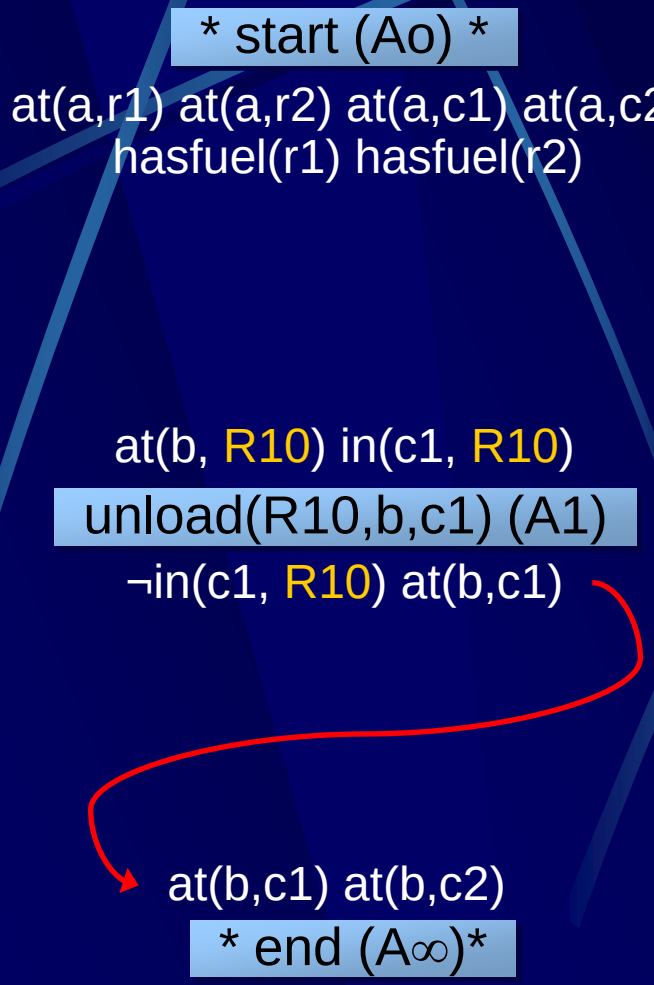
agenda = {

( at(b,c1) , A∞ )

( at(b,c2) , A∞ )

}

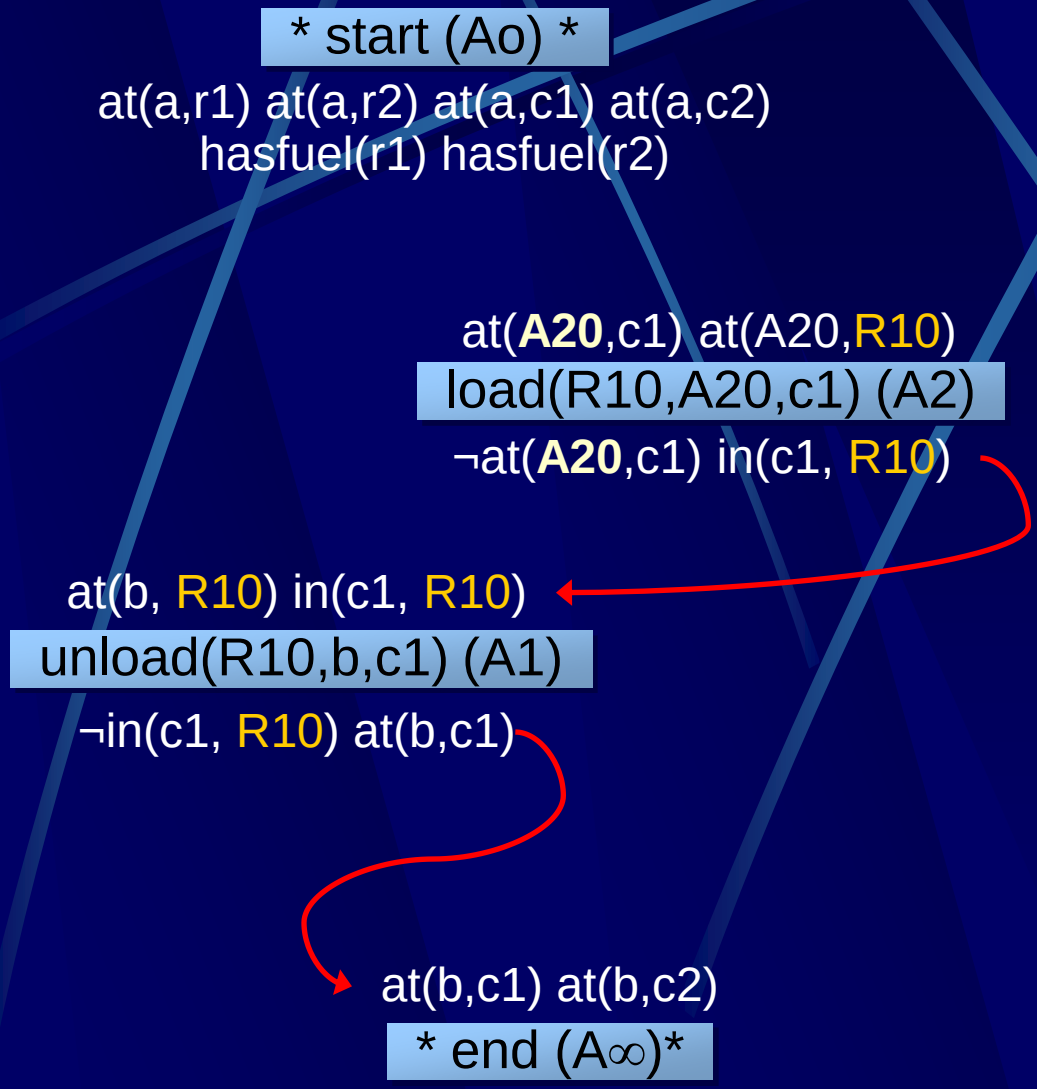
# Llamada Inicial a UCPOP




agenda = {
**( at(b,c1) , A∞ )**
( at(b,c2) , A∞ ))
}
Q = at(b,c1) $A_{need}$ = A∞
$A_{add}$ = **unload(R10,b,c1) = A1**
$link_1$ = A1 - Q -> A∞

agenda´ = {
**( at(b,R10) , A1 )**
**( in(c1, R10) , A1 )**
( at(b,c2) , A∞ )
}
O´ = { Ao < A∞, **A1 < A∞** }

# Segunda Llamada a UCPOP

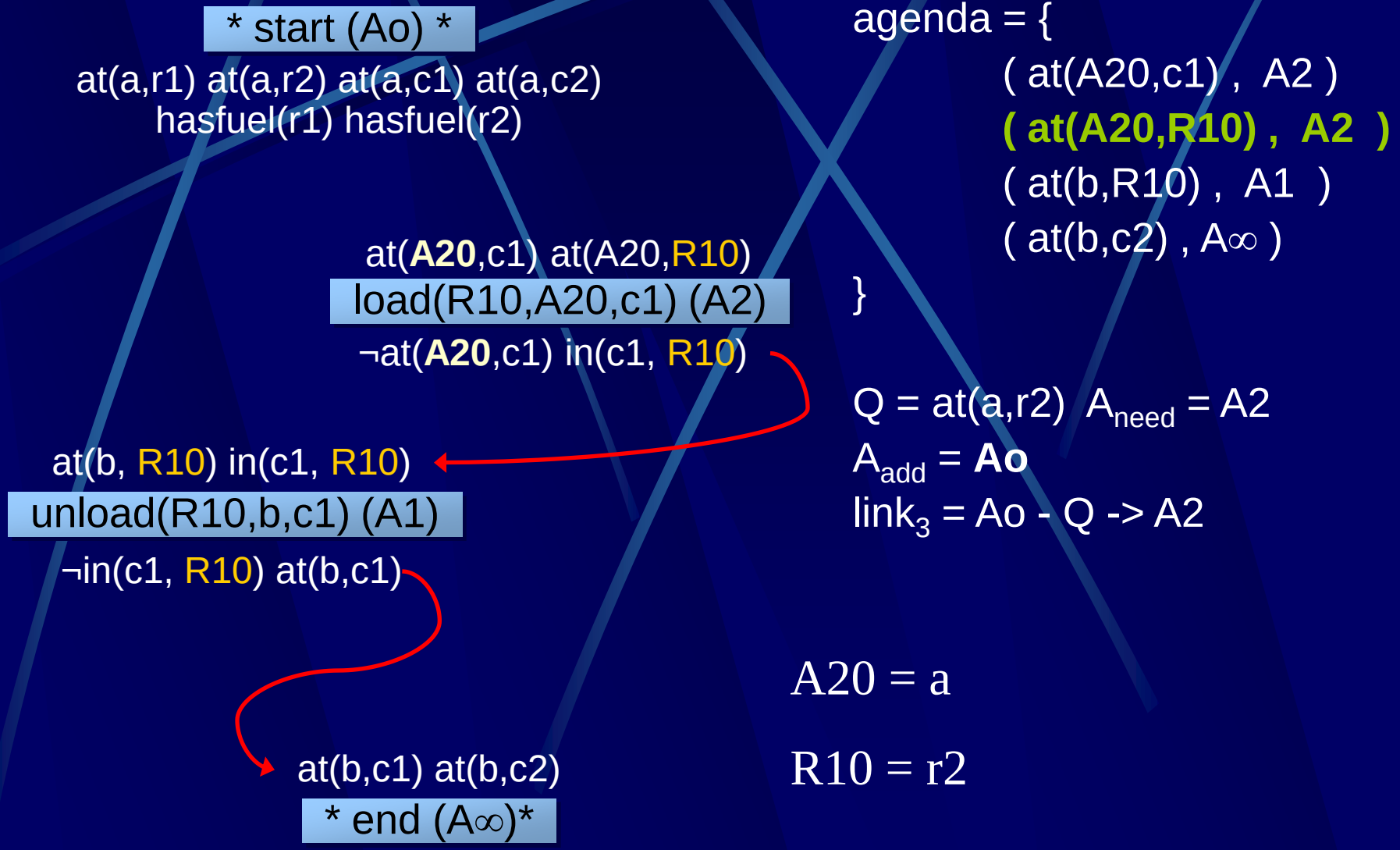



agenda = {
( at(b,R10) , A1 )
**( in(c1, R10,) , A1 )**
( at(b,c2) , A∞ )}

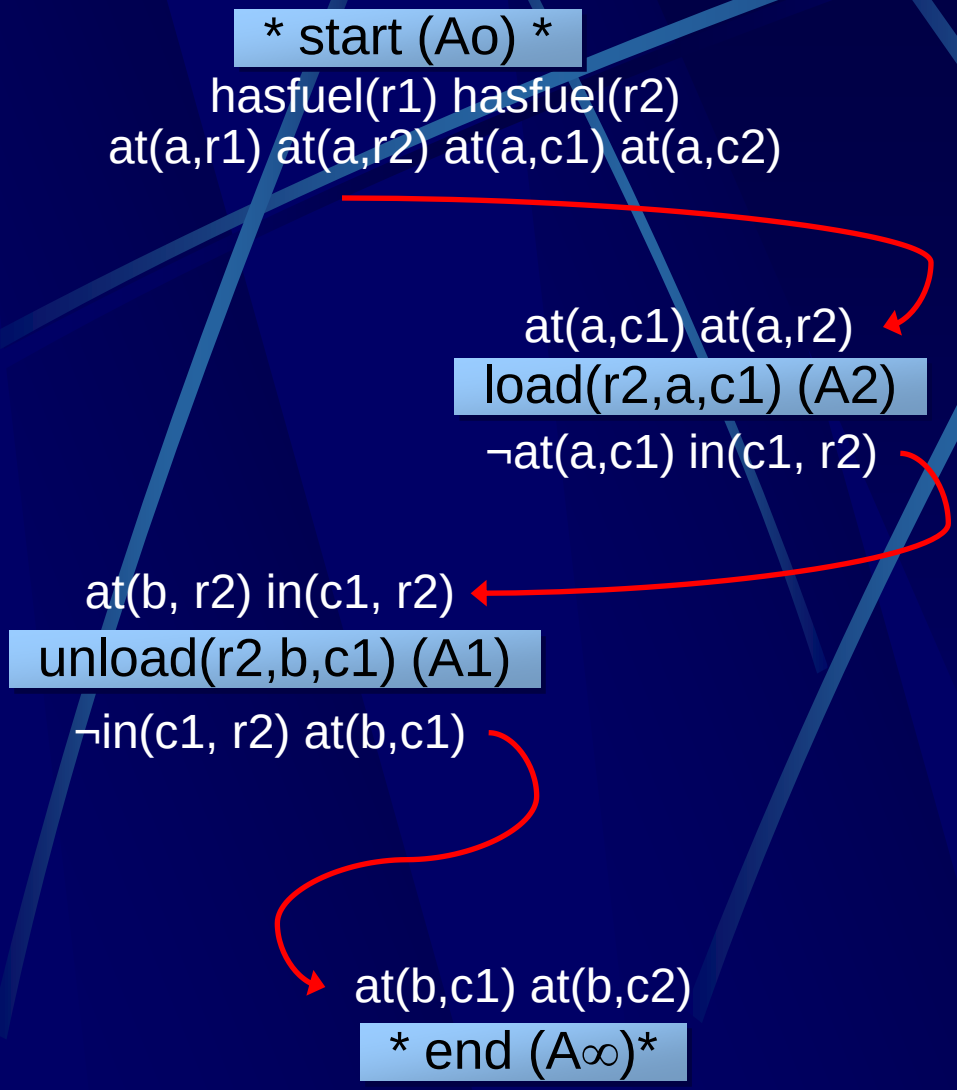
Q = in(c1,R10) $A_{need}$ = A1
$A_{add}$ = **load(R10,A20,c1) = A2**
$link_2$ = A2 - Q -> A1

agenda´ = {
**( at(A20,c1) , A2 )**
**( at(A20,R10) , A2 )**
( at(b, R10) , A1 )
( at(b,c2) , A∞ )
}
O´ = { Ao < A∞, A1 < A∞,
~~**A2 < A1**~~ }

# Tercera Llamada a UCPOP



# Tercera Llamada a UCPOP




agenda = {

( at(A20,c1) , A2 )

**( at(A20,R10) , A2 )**

( at(b,R10) , A1 )

( at(b,c2) , A∞ )

}

Q = at(a,r2) $A_{need}$ = A2

$A_{add}$ = **Ao**

$link_3$ = Ao - Q -> A2

agenda´ = {

( at(a,c1) , A2 )

( at(b,r2) , A1 )

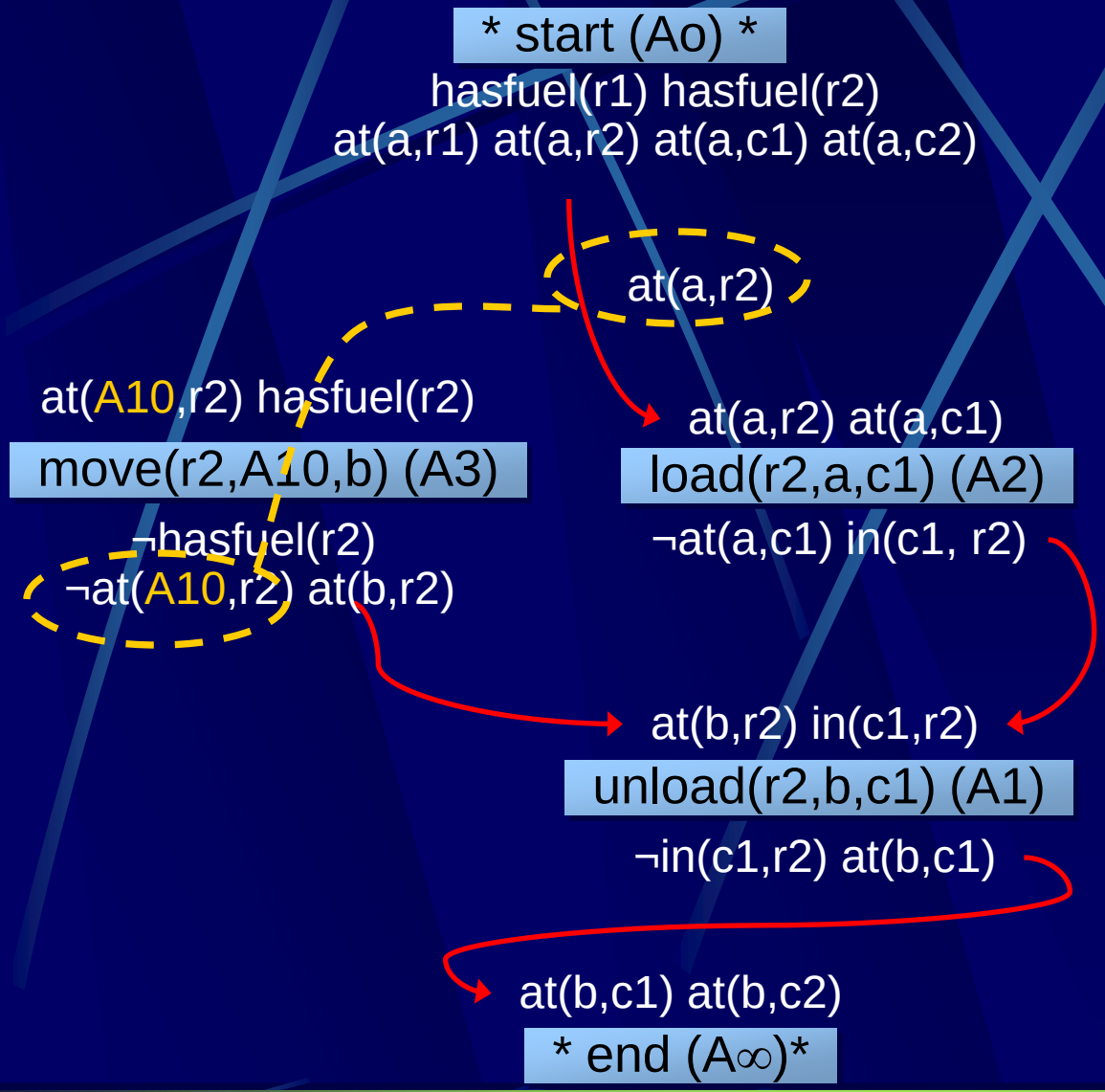
( at(b,c2) , A∞ )

}

O´ = { Ao < A∞, A1 < A∞,

A2 < A1, **Ao < A2** }

# Cuarta Llamada a UCPOP




agenda = {

( at(a,c1) , A2 )

**( at(b,r2) , A1 )**

( at(b,c2) , A∞ )

}

Q = at(b,r2) $A_{need}$ = A1

$A_{add}$ = **move(r2,A10,b) = A3**

$link_4$ = A3 - Q -> A1

agenda´ = {

**( at(A10,r2) , A3 )**

**( hasfuel(r2) , A3 )**

( at(a,c1) , A2 )

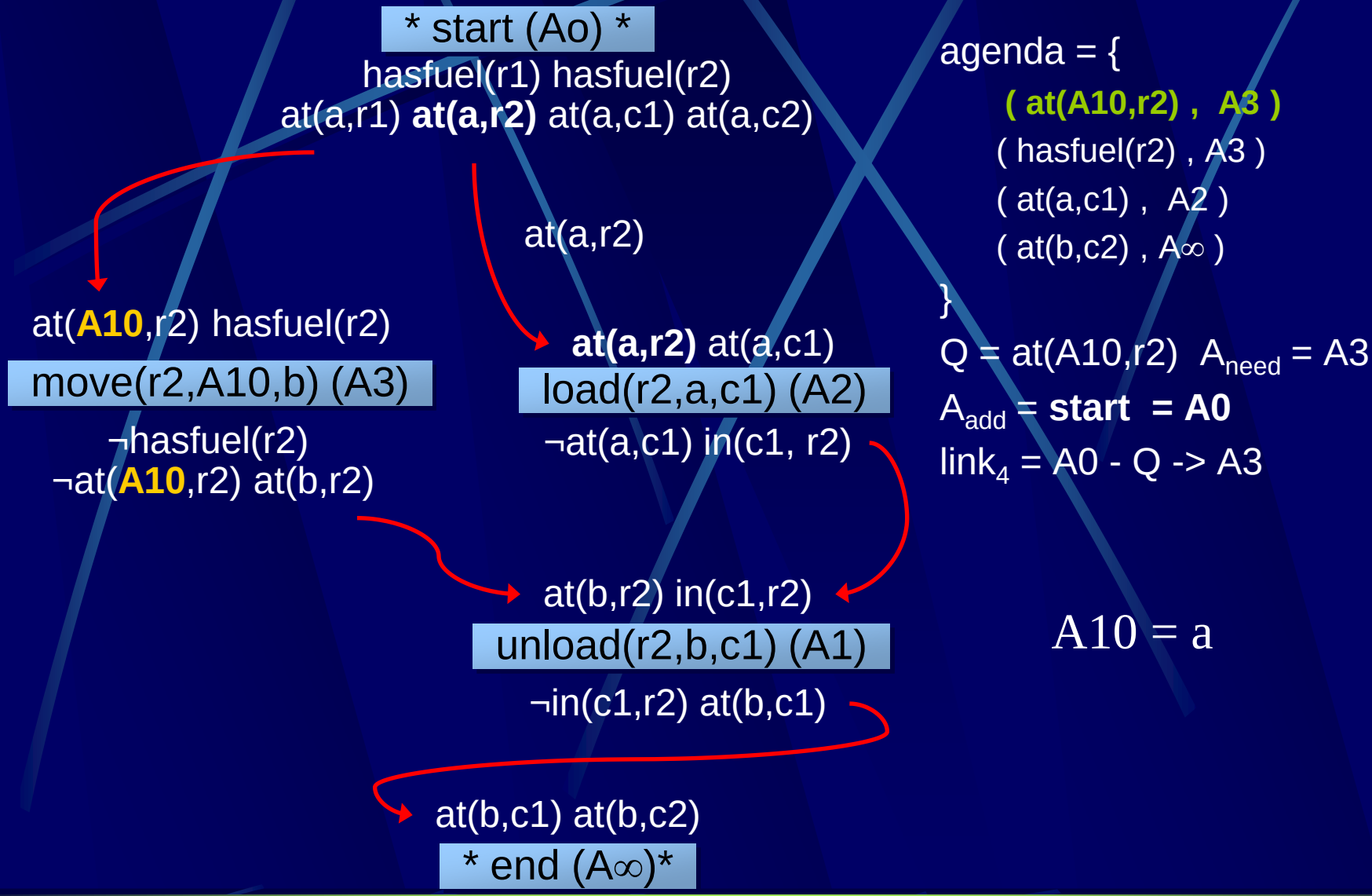
( at(b,c2) , A∞ )

}

O´ = { Ao < A∞, A1 < A∞,

A2 < A1, Ao < A2, **A3 < A1** }

# Cuarta Llamada a UCPOP




agenda = {

( at(A10,r2) , A3 )

( hasfuel(r2) , A3 )

( at(a,c1) , A2 )

( at(b,c2) , A∞ )

}

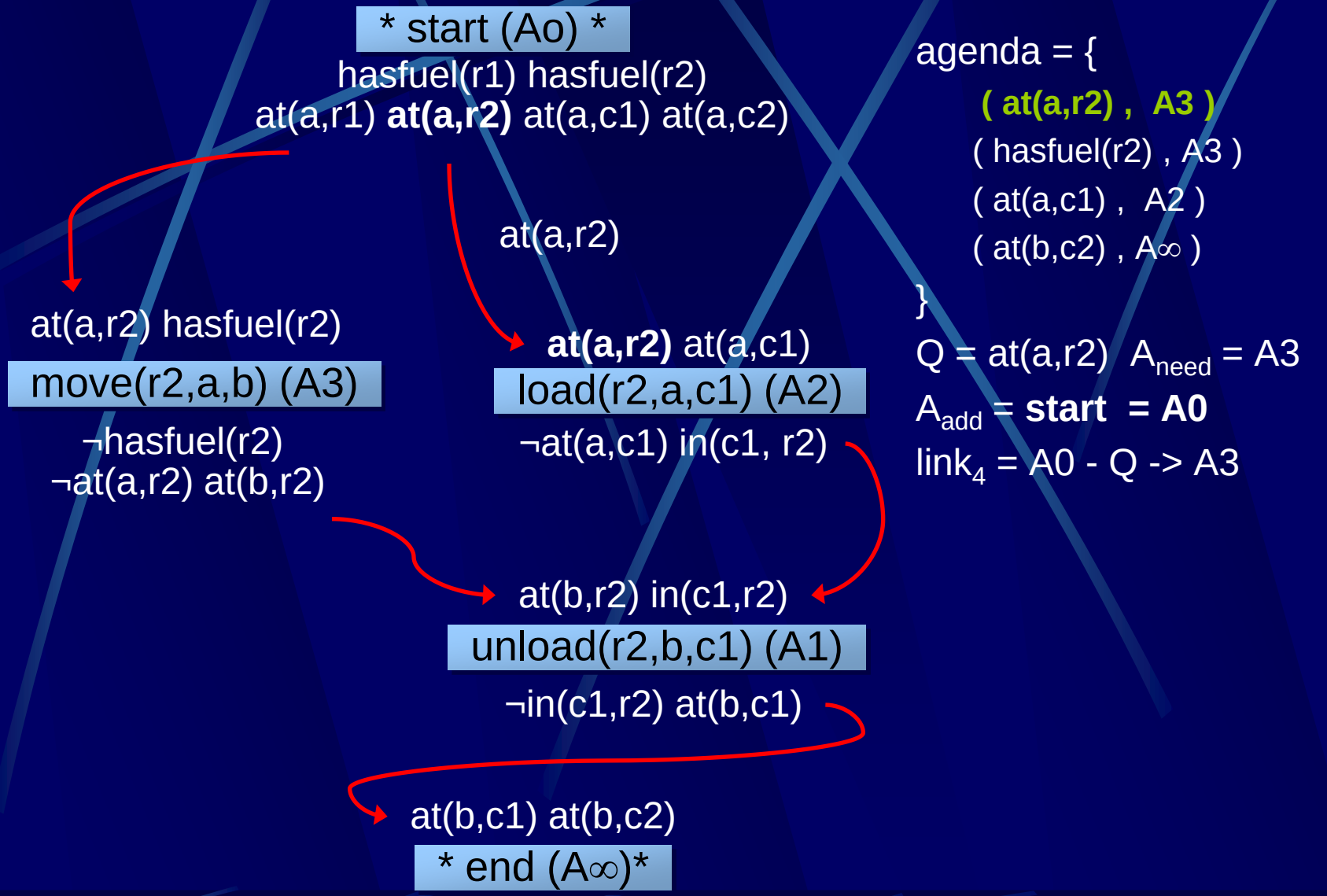
Q = at(A10,r2) $A_{need}$ = A3

$A_{add}$ = **start = A0**

$link_4$ = A0 - Q -> A3

A10 = a

# Cuarta Llamada a UCPOP




agenda = {

( at(a,r2) , A3 )

( hasfuel(r2) , A3 )

( at(a,c1) , A2 )

( at(b,c2) , A∞ )

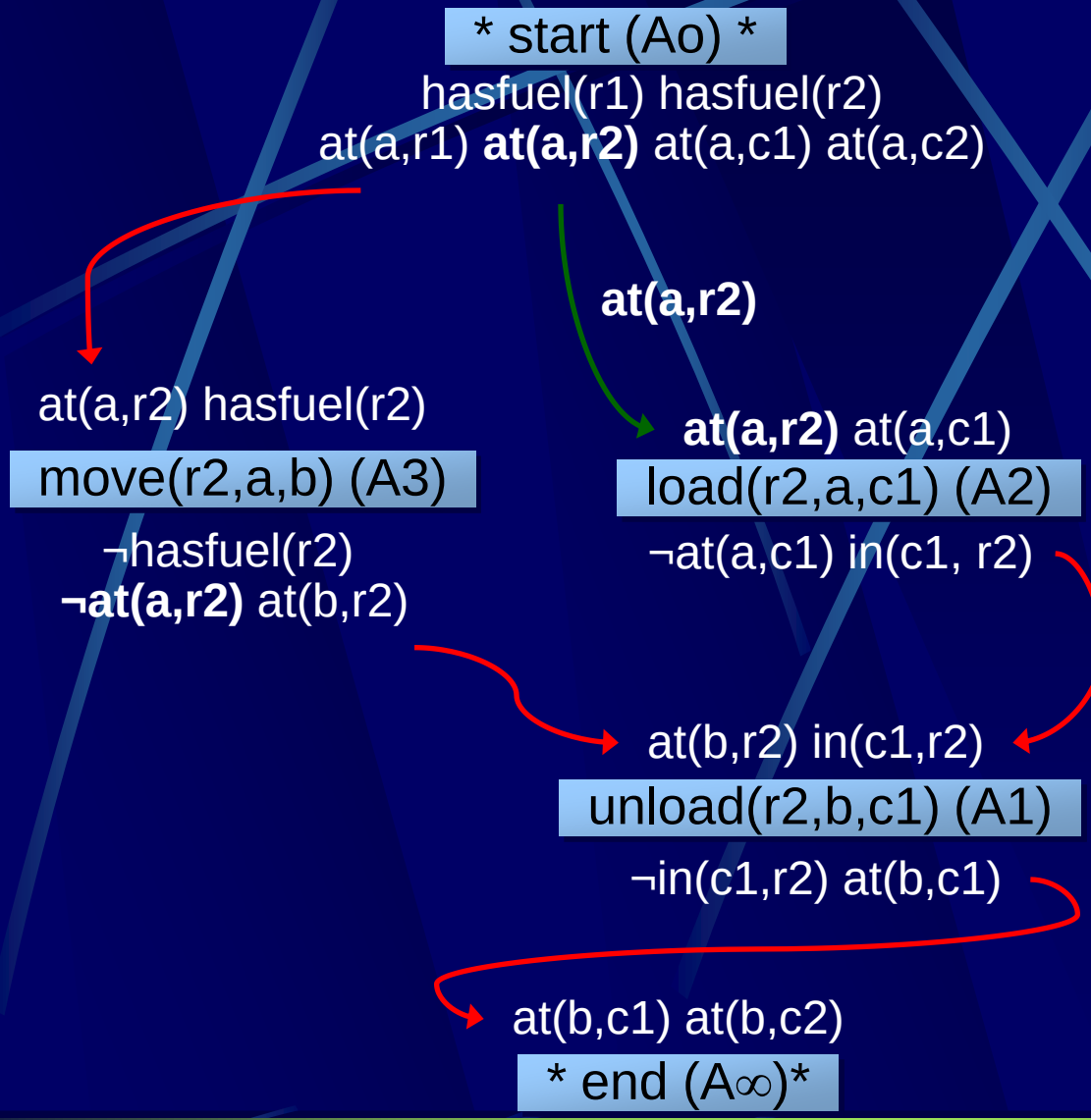
}

Q = at(a,r2) $A_{need}$ = A3

$A_{add}$ = **start = A0**

$link_4$ = A0 - Q -> A3

# Cuarta Llamada a UCPOP




agenda = {
( **at(a,r2)** , **A3** )
( hasfuel(r2) , A3 )
( at(a,c1) , A2 )
( at(b,c2) , A∞ )
}

Q = at(a,r2) $A_{need}$ = A3

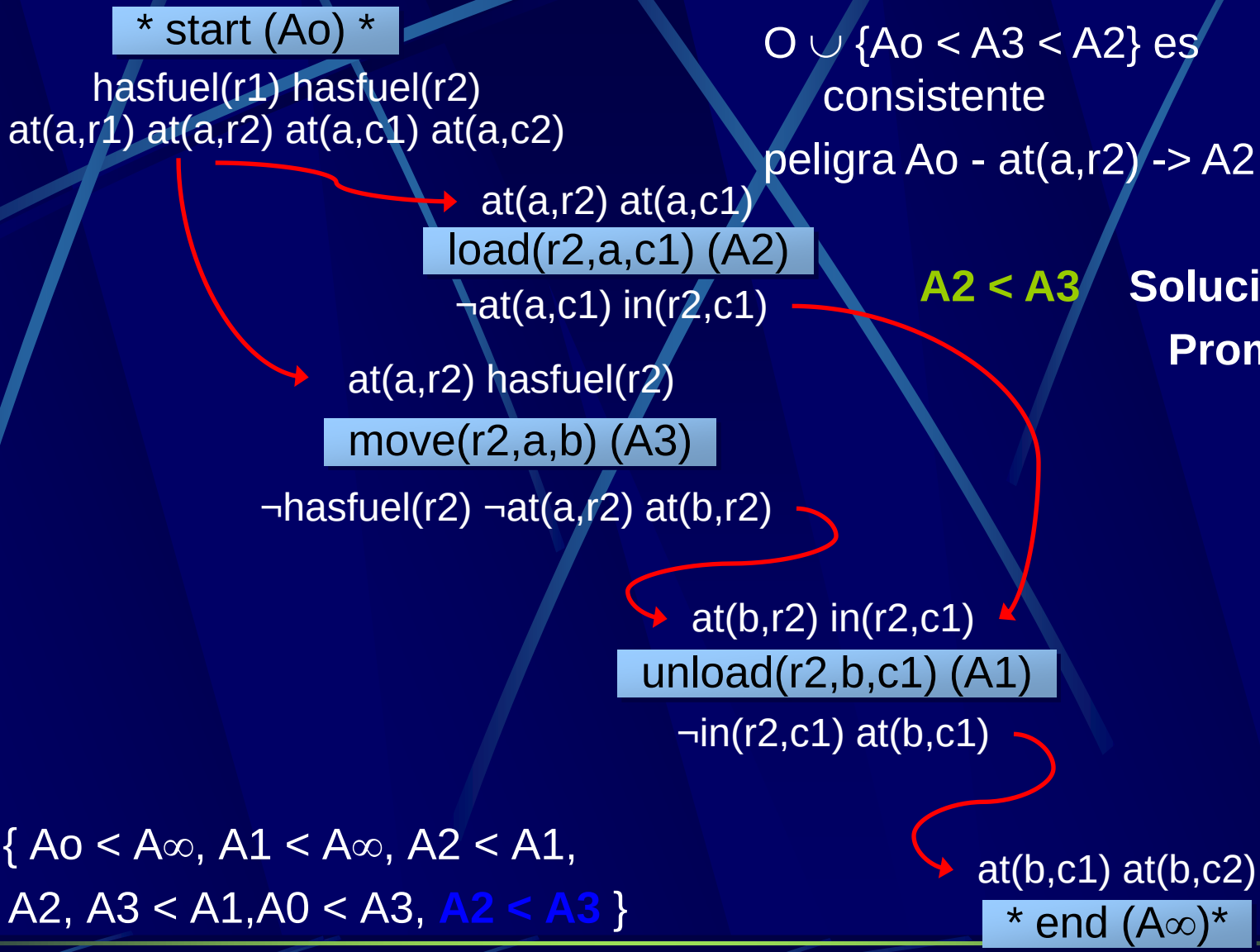
$A_{add}$ = **start = A0**

$link_4$ = A0 - Q -> A3

agenda´ = {
(hasfuel(r2) , A3 )
( at(a,c1) , A2 )
( at(b,c2) , A∞ )
}

O´ = { Ao < A∞, A1 < A∞,
A2 < A1, Ao < A2, A3 < A1
**A0 < A3**}

# Resolución del Conflicto




$O \cup \{Ao < A3 < A2\}$ es consistente

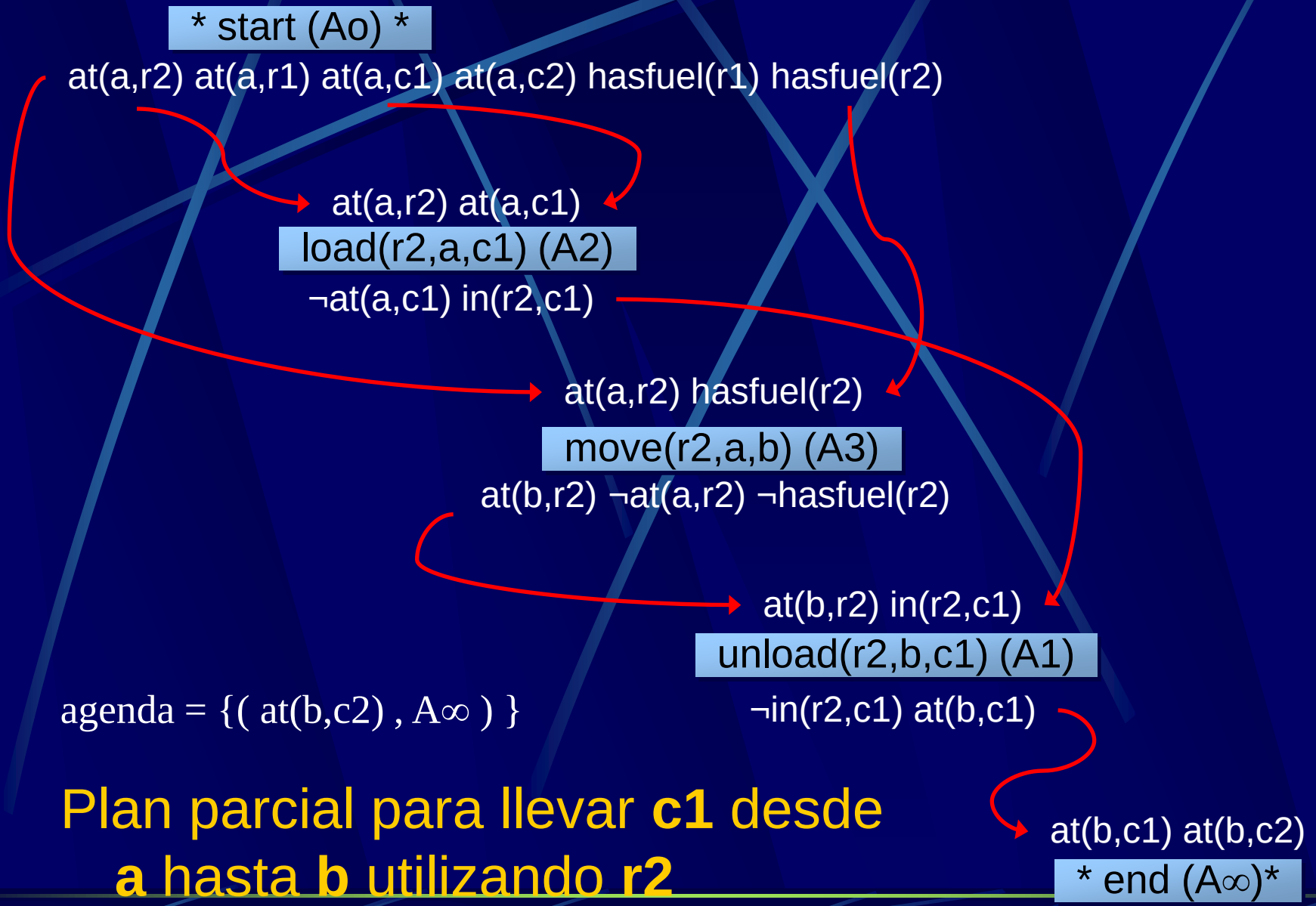
peligra Ao - at(a,r2) -> A2

**A2 < A3** **Solución por Promoción**

$O' = \{ Ao < A\infty, A1 < A\infty, A2 < A1,$

$Ao < A2, A3 < A1, A0 < A3,$ **$A2 < A3$** $\}$

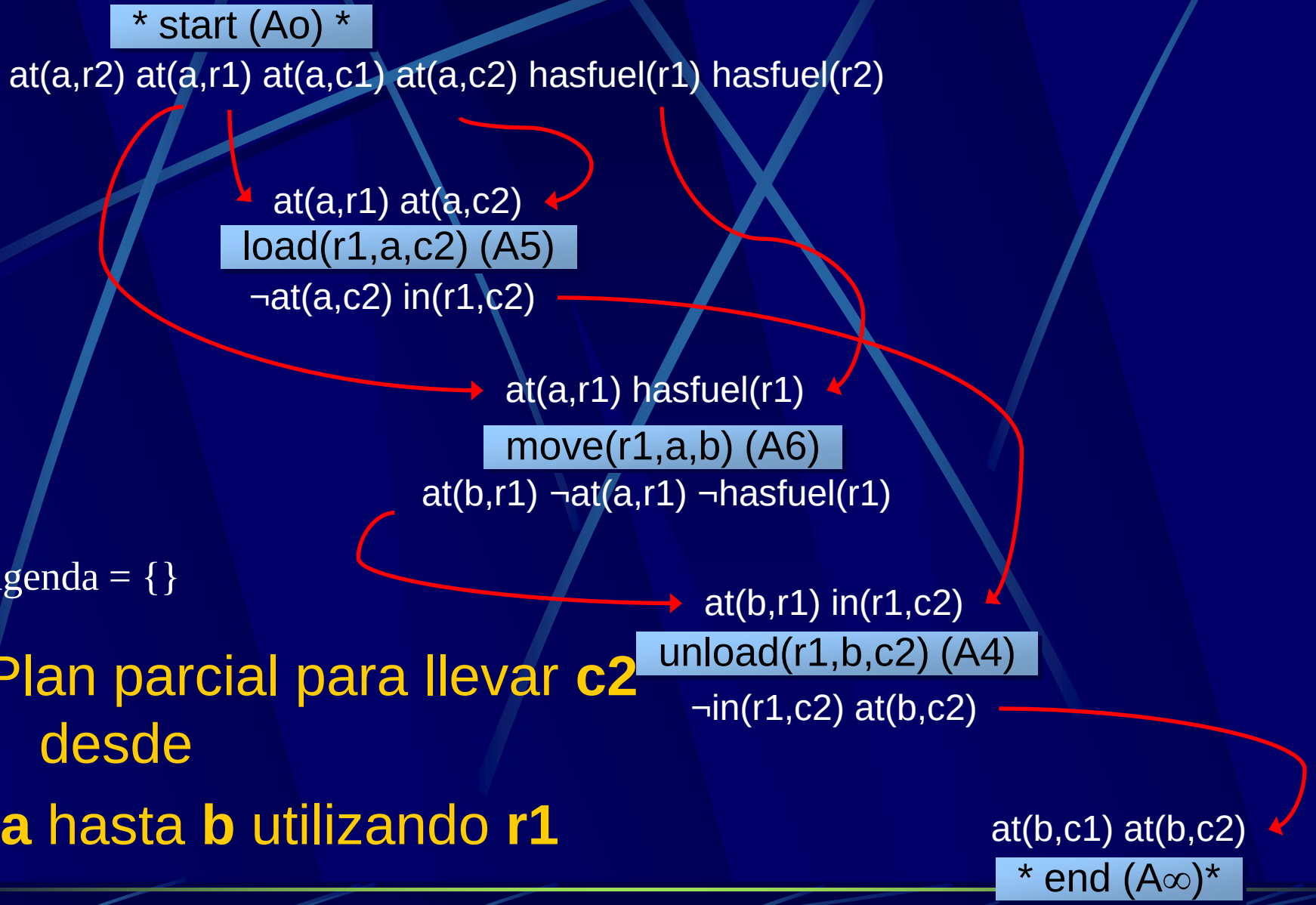
# Seguimiento UCPOP (1)




agenda = {( at(b,c2) , A∞ ) }

Plan parcial para llevar **c1** desde **a** hasta **b** utilizando **r2**

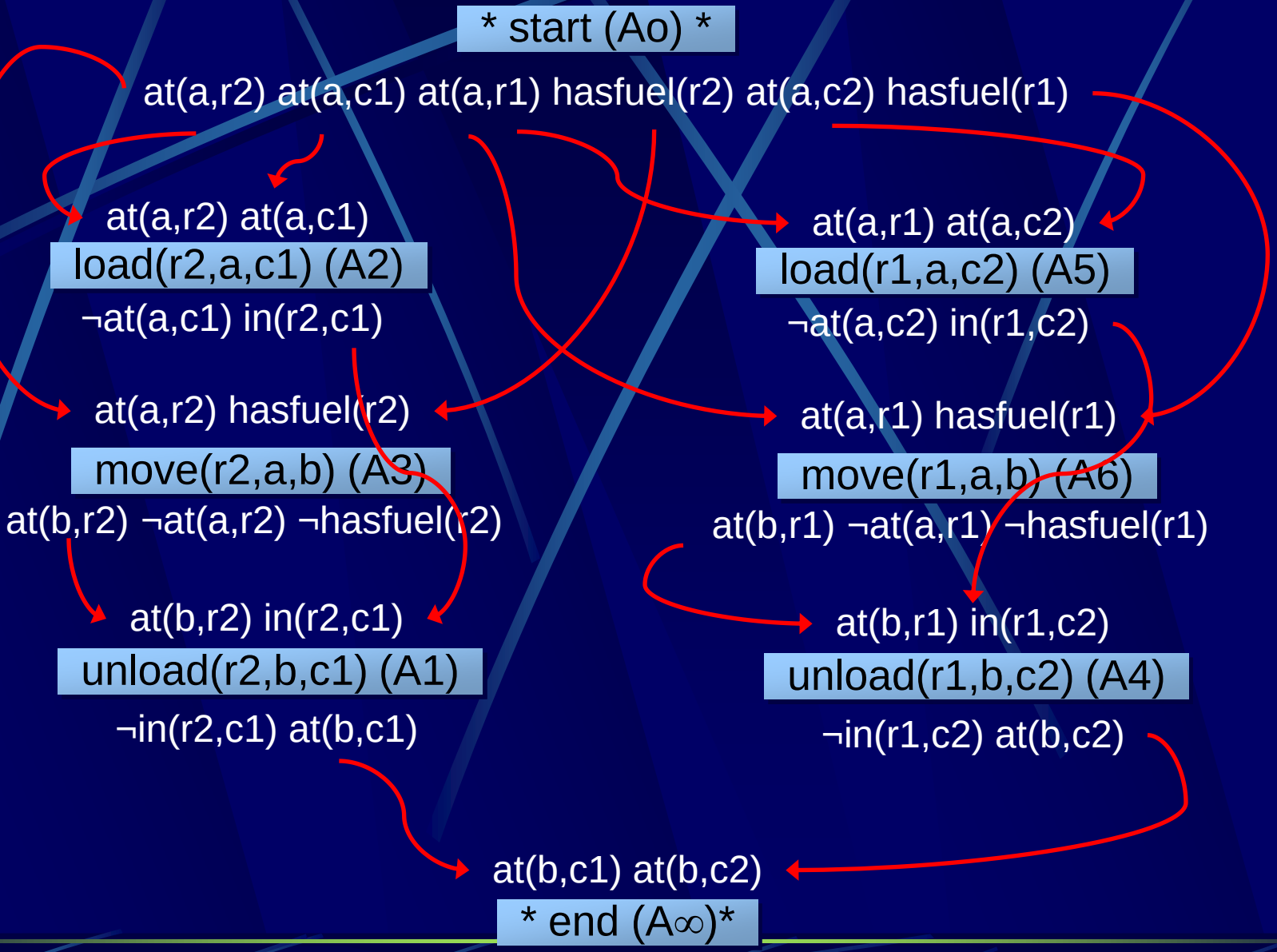
# Seguimiento UCPOP (2)



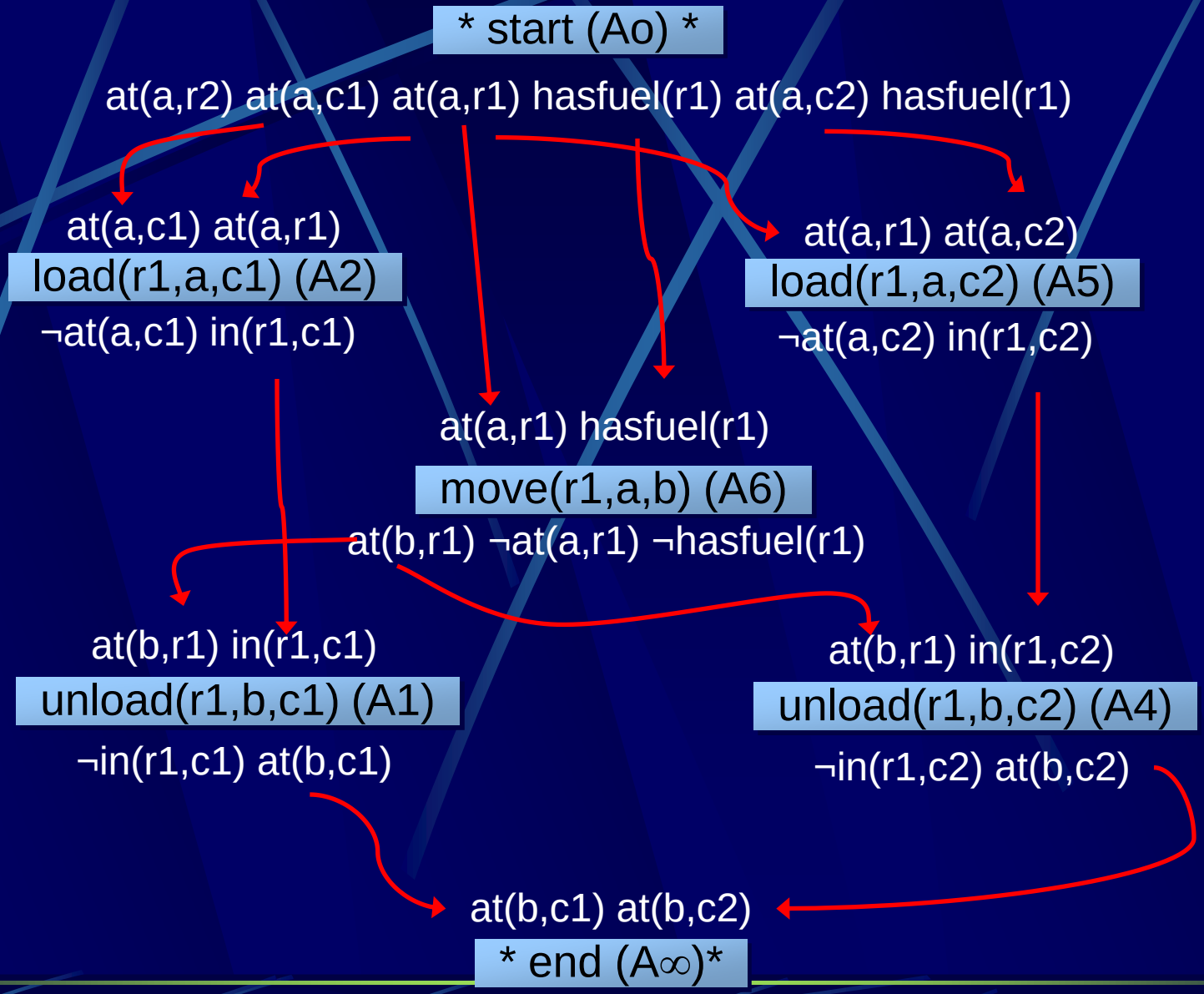

Plan parcial para llevar **c2** desde

**a** hasta **b** utilizando **r1**

# Plan Completo



# Plan Completo (2)



# Otros enfoques

- GraphPlan
- Motion planning (sample-based motion planning)
- Basados en Teoría de Decisión
- Planning probabilístico
- Planning basado en preferencias
- Planning heurístico
- Planning con métodos temporales

# Bibliografía


- An Introduction to Least Commitment Planning - D. Weld - AI Magazine, 15:4 - 1994 - pp. 27- 61
- UCPOP: A sound, complete, partial order planner for ADL - J.S. Penberthy, D. Weld - Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning, October 1992, pp. 103-114
- A. Blum and M. Furst, "Fast Planning Through Planning Graph Analysis" , *Artificial Intelligence*, 90:281--300 (1997).

# Bibliografía


- Planning Algorithms – Steven LaValle – 2006 - Cambridge University Press, 842 pags.
- Automated Planning and Acting © Malik Ghallab, Dana Nau and Paolo Traverso. Published by Cambridge University Press, ISBN: 9781107037274, August 2016.