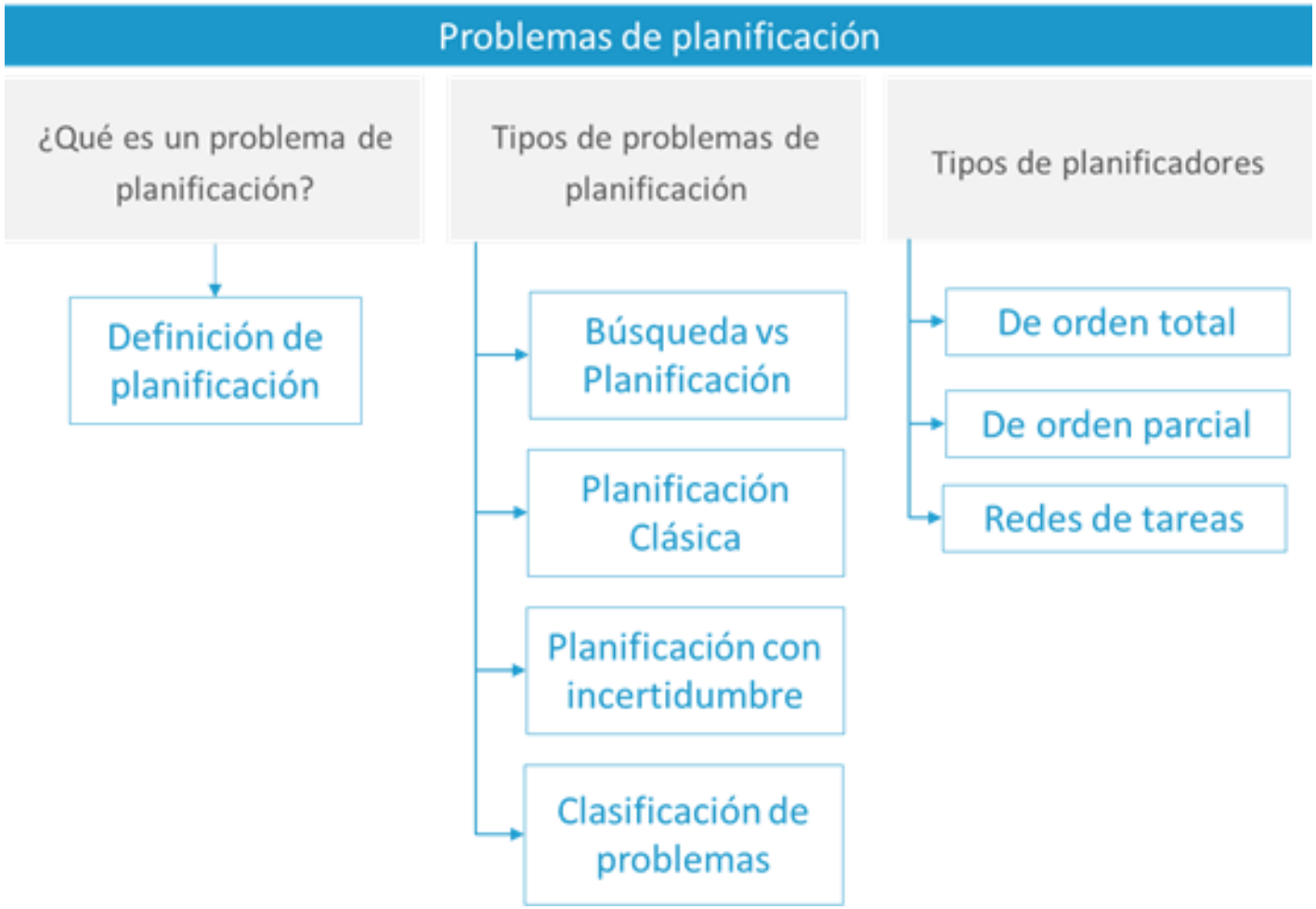
Razonamiento y planificación automática

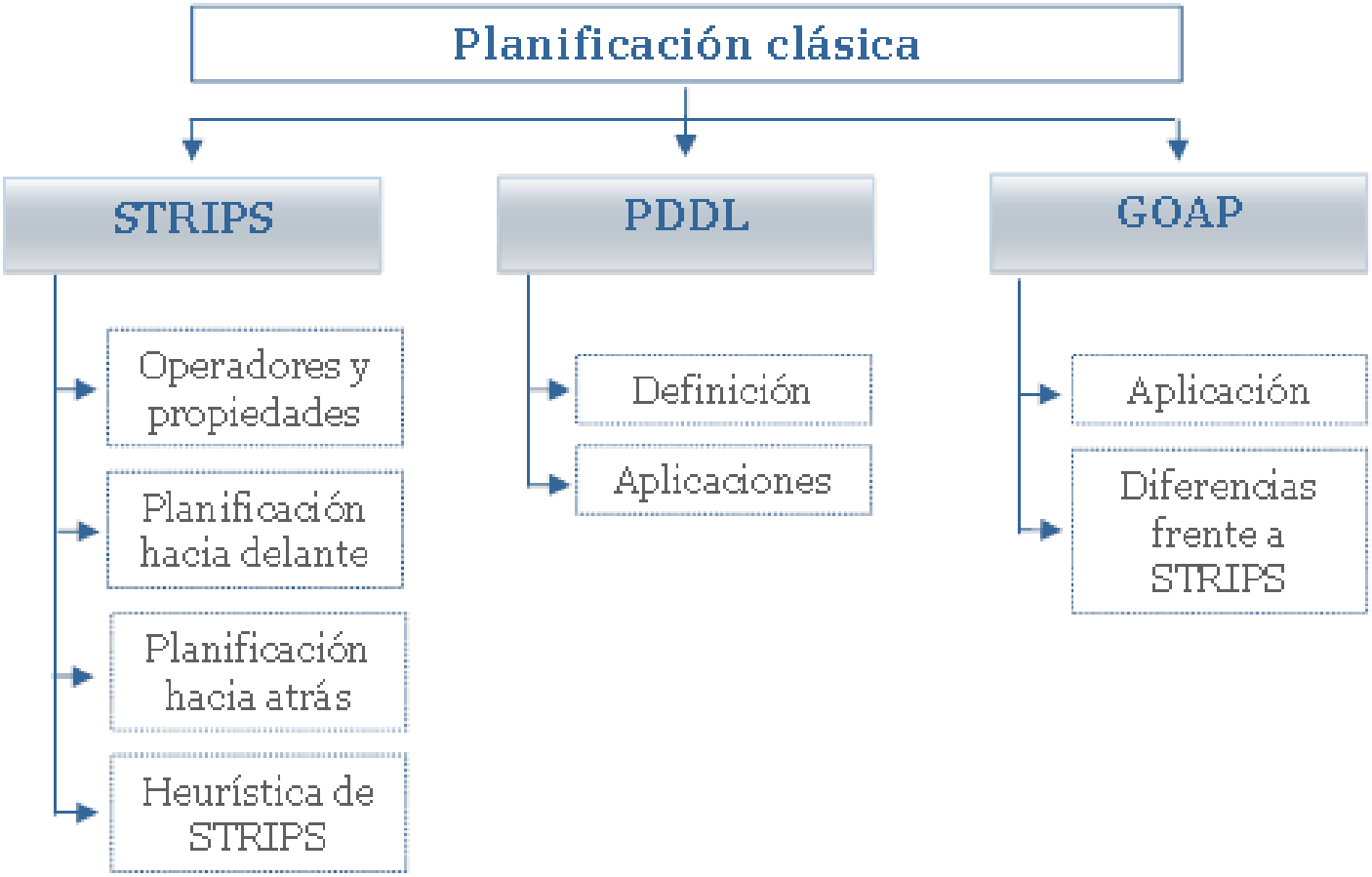
Nerea Luis Mingueza / Alejandro Cervantes

Tema 8: Sistemas basados en STRIPS


Problemas de planificación
¿Qué es un problema de planificación?
Tipos de problemas de planificación
Tipos de planificadores
Definición de planificación
Búsqueda vs Planificación
Planificación Clásica
Planificación con incertidumbre
Clasificación de problemas
De orden total
De orden parcial
Redes de tareas

Índice

- Problemas de planificación
  - STRIPS
  - PDDL
  - GOAP


Planificación clásica
STRIPS
PDDL
GOAP
Operadores y propiedades
Planificación hacia delante
Planificación hacia atrás
Heurística de STRIPS
Definición
Aplicaciones
Aplicación
Diferencias frente a STRIPS

Repaso Planificación

Componentes de la planificación

- **Estado actual del entorno:** es una representación estructura que crea el agente al momento de percibir su entorno. Como, por ejemplo, su posición actual, si llueve o no llueve, la posición de una mesa, etc.
- **Meta:** es cualquier condición que un agente quiera satisfacer. Un agente puede tener varias posibles metas, pero en un instante determinado solo una puede estar activa, controlando el comportamiento.
- **Acción:** es un paso simple y atómico dentro de un plan que hace que un agente haga algo (ir a un punto, activar un objeto, etc.)
- **Plan:** secuencia de acciones.
- **Proceso de planificación:** Un agente proporciona a un sistema (planificador) un estado actual del entorno, un conjunto de acciones y una meta que desea satisfacer, y el planificador busca un plan que con la ejecución de sus acciones consiga esta meta.

Mundo de los bloques

- Un conjunto de bloques, una mesa, y un brazo de un robot
- Todos los bloques son iguales de tamaño, forma y color, diferenciándose en el nombre
- La mesa tiene extensión ilimitada
- Cada bloque puede estar encima de la mesa, encima de un solo bloque, o sujeto por el brazo del robot
- El brazo de robot sólo puede sujetar un bloque cada vez

Resolver problemas supone pasar de una configuración (estado) inicial a un estado en el que sean ciertas unas metas

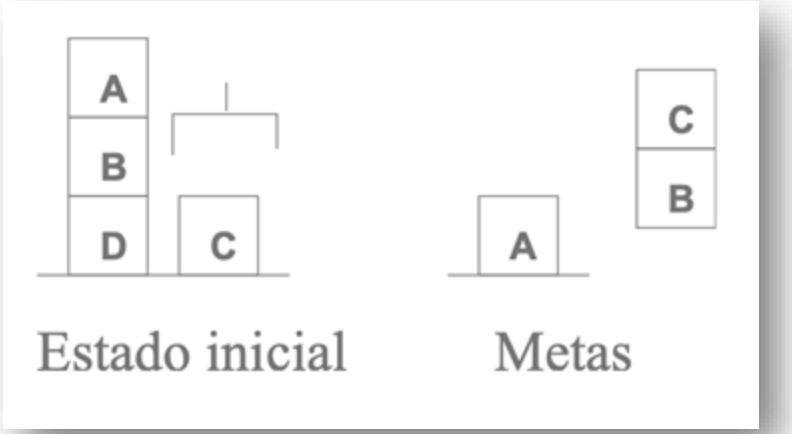
Representación en mundo de los bloques

Se podrían utilizar los siguientes **predicados:**

- *encima(x,y):* el bloque x esta encima del y
- *en-mesa(x):* el bloque x esta encima de la mesa
- *libre(x):* el bloque x no tiene ningun bloque encima
- *sujeto(x):* el brazo del robot tiene cogido al bloque x
- *brazo-libre:* el brazo del robot no tiene sujeto ningún bloque

Estado inicial:
encima(A,B),encima(B,D),en-mesa(D),
en-mesa(C),libre(A),libre(C),brazo-libre

Metas:
en-mesa(A),encima(C,B)



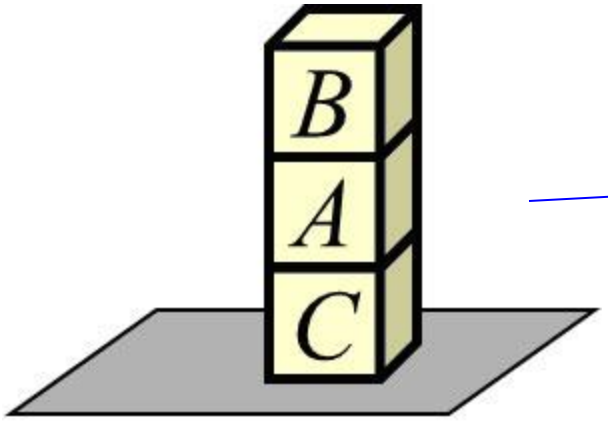
STRIPS

Timeline

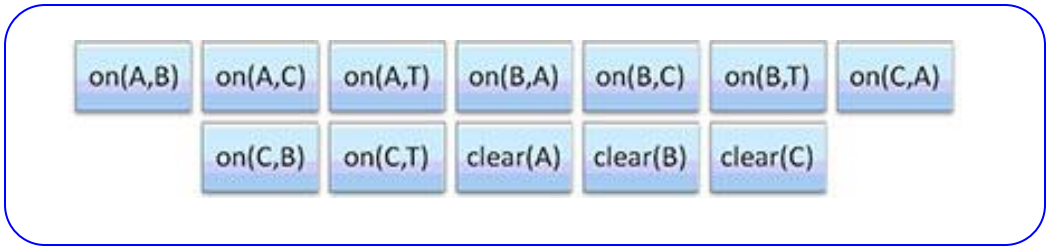
- STRIPS (Fikes y Nilsson, 1971)
- Lenguaje de definición de planes PDDL (McDermott, Ghallab, Howe, Knoblock, Ram y Veloso, 1998).
- Sistemas con reacción en tiempo real: GOAP (Jeff Orkin, 2003)

STRIPS

Primera estandarización conocida de una tarea de planning



- **Proposiciones** Conjunto global de proposiciones F



Estado actual (subconjunto de F)

- **Operadores**

Expresan las acciones que el agente contempla en el problema y por las cuales desea resolver el estado actual y transformar el entorno para alcanzar la meta o estado final.

https://people.cs.pitt.edu/~milos/courses/cs1571-Fall2010/Lectures/Class18.pdf

Operadores en STRIPS

<nombre_operador>(<PC>,<A>,>E>)		
PC	Precondición	Conjunto de proposiciones que deben estar presentes en un estado para poder aplicar dicho operador.
A	Lista de añadir	Conjunto de proposiciones que se añadirán al estado tras aplicar el operador.
E	Lista de eliminar	Son aquellas proposiciones que se quitarán del estado tras la aplicación del operador. **Debería ser un subconjunto de PC.**

A: effects E: deletes

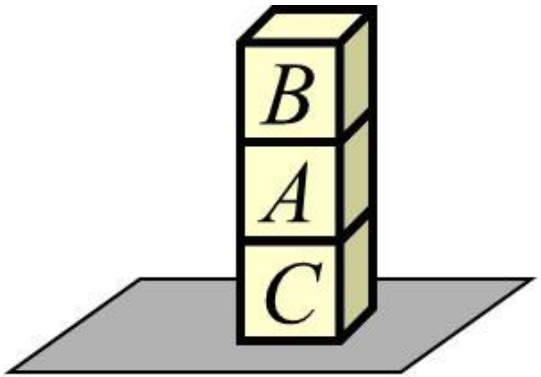
Operadores en STRIPS

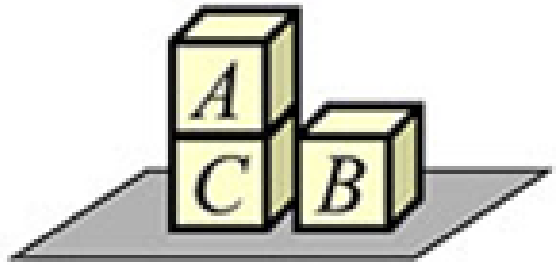
Move (?block1, ?block2, ?table)

Precondition: on(?block1, ?block2), clear(?block1)

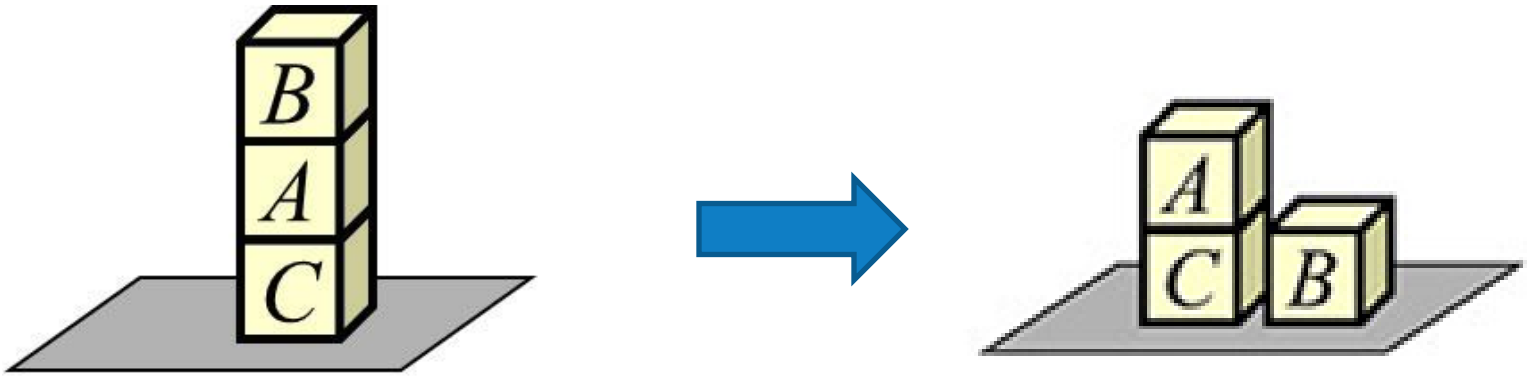
Deletes: on(?block1,?block2)

Effects: on(?block1, ?table), clear(?block2)





Instanciación de un operador




```
Move (?block1, ?block2, ?table)
   Precondition: on(?block1, ?block2), clear(?block1)
   Deletes: on(?block1,?block2)
   Effects: on(?block1, ?table), clear(?block2)
```

Sólo se cumplen las precondiciones de una forma:

```
?block1 = B
?block2 = A

?table = T
```

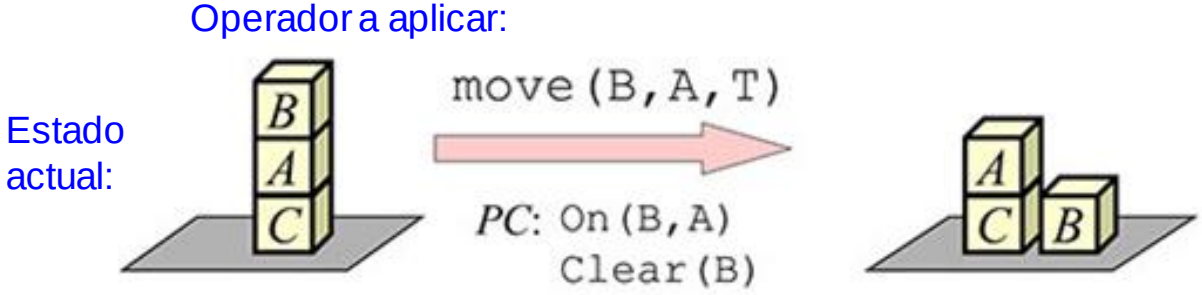
Precondiciones

Instancia del operador

```
Move(B,A,T)
```

```
Deletes: on(B,A)
Effects:
on(B, T), clear(A)
```

Aplicando operadores a estados en STRIPS



Move (?block1, ?block2, ?table)

Precondition: on(?block1, ?block2), clear(?block1)

Deletes: on(?block1,?block2)

Effects: on(?block1, ?table), clear(?block2)

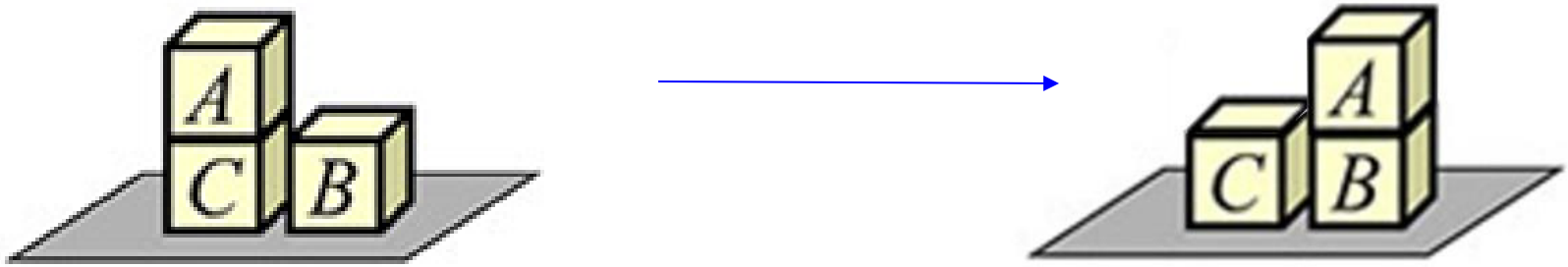
Operadores en STRIPS

Move2 (?block1, ?block2, ?block3)

Precondition: on(?block1, ?block2), clear(?block1), clear(?block3), ?block1 != ?block3

Deletes: on(?block1,?block2), clear(?block3)

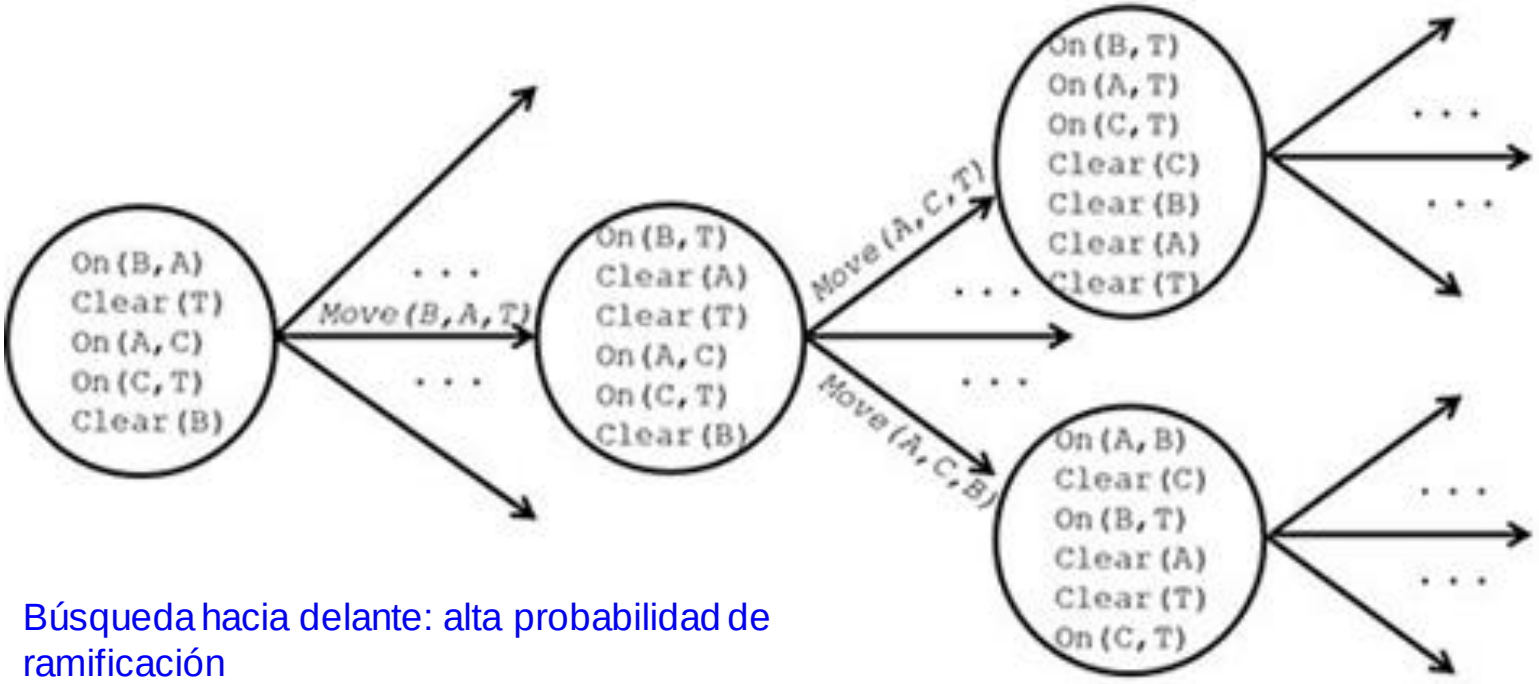
Effects: on(?block1, ?block3), clear(?block2)



Aplicando operadores a estados en STRIPS

- La única forma de cambiar el estado actual es aplicando un operador.
- Solo cambian las proposiciones que reflejadas en los effects y deletes.
- Se genera un árbol de búsqueda que, mediante heurísticas predefinidas (o sin ellas), estudiará qué operador aplicar en cada momento.
- Con esta idea, aplicamos algoritmos como los de búsqueda en amplitud, profundidad e incluso búsquedas heurísticas de A*.

Búsqueda hacia adelante

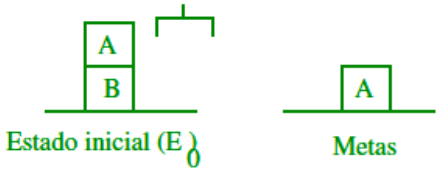


Búsqueda hacia delante: alta probabilidad de ramificación

¿Qué tipo de heurísticas se nos ocurren?

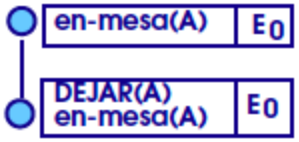
Búsqueda hacia atrás

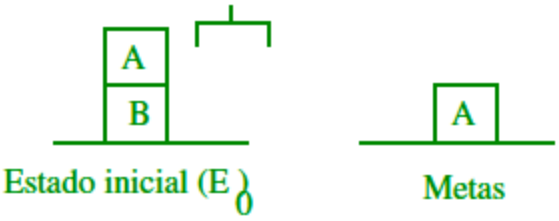
Paso 1



en-mesa(B)
brazo-libre
encima(A,B)

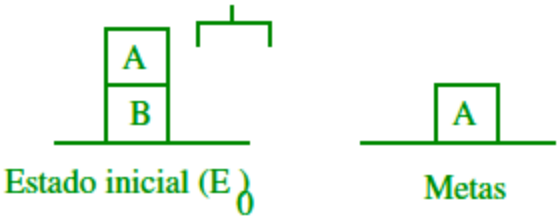
Paso 2





DEJAR(?b)
P: sujeto(?b)
D: sujeto(?b)
E: en-mesa(?b)

Paso 3



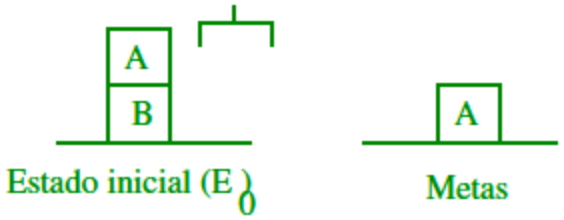
Añado encima los operadores y sus precondiciones

Búsqueda hacia atrás

LEVANTAR(?b)
P: en-mesa(?b),brazo-libre
D: en-mesa(?b),brazo-libre
E: sujeto(?b)

QUITAR(?b1,?b2)
P: libre(?b1),encima(?b1,?b2),brazo-libre
D: encima(?b1,?b2)
E: libre(?b2),sujeto(?b1)

Paso 3



¿Cuántos sucesores hay?

Búsqueda hacia atrás

LEVANTAR(?b)

P: en-mesa(?b),brazo-libre

D: en-mesa(?b),brazo-libr

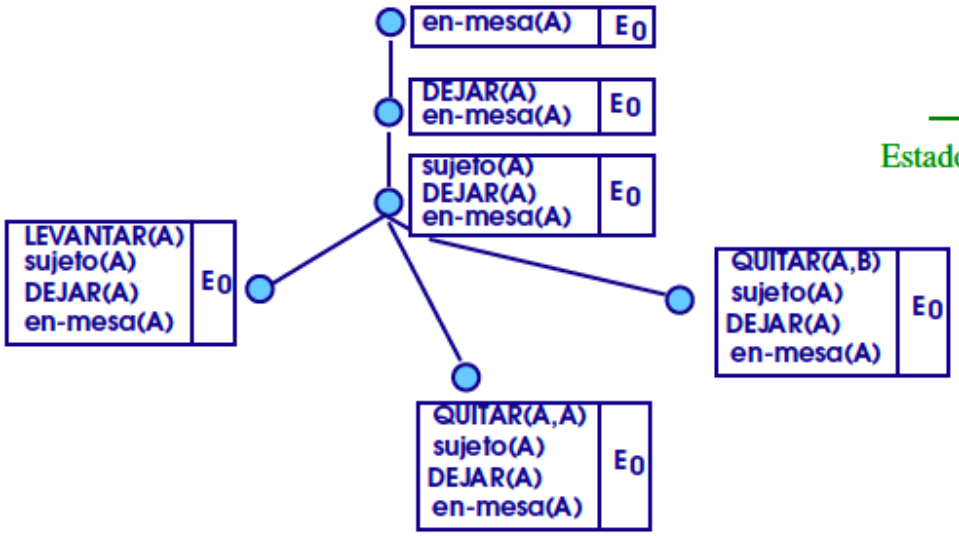
E: sujeto(?b)

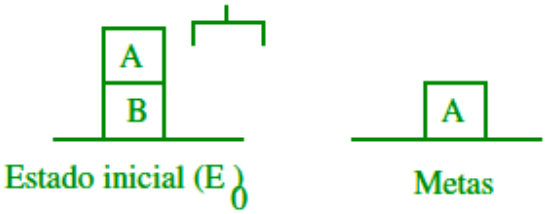
QUITAR(?b1,?b2)

P: libre(?b1),encima(?b1,?b2),brazo-libre

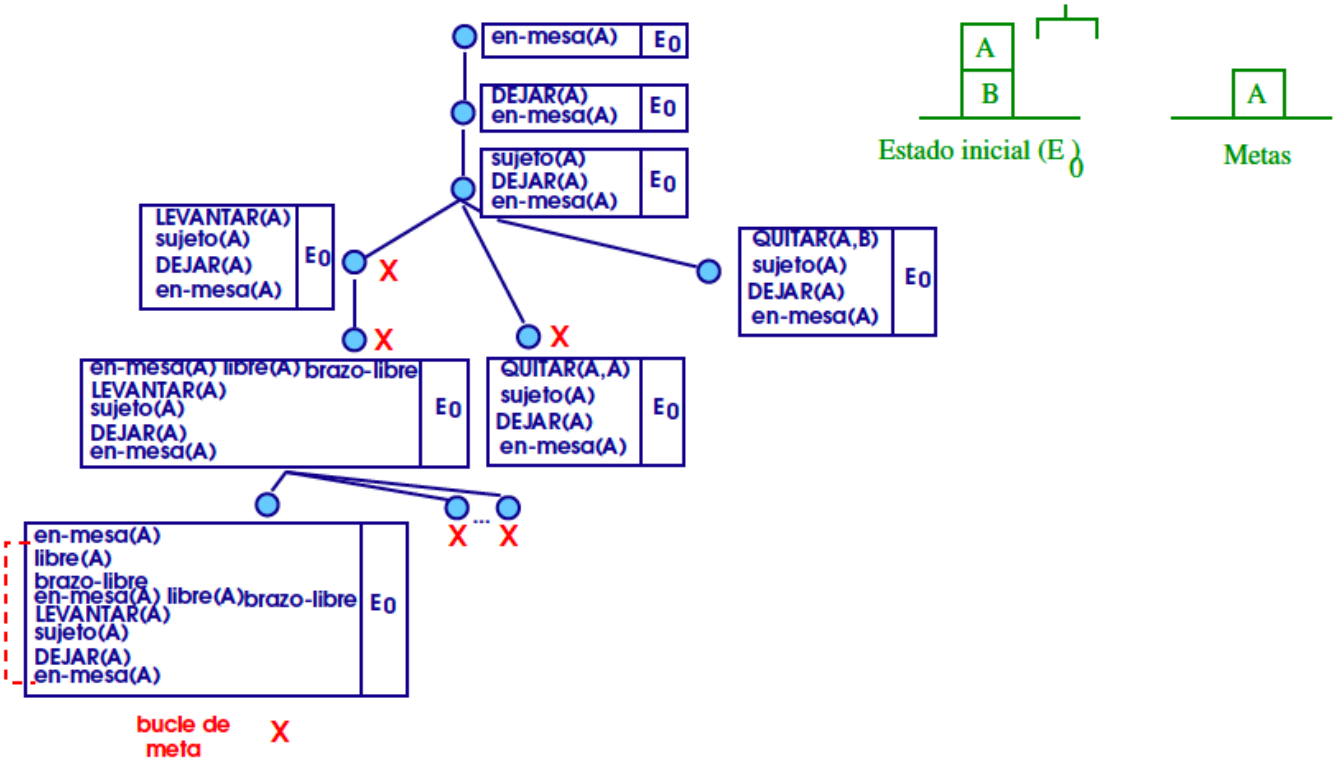
D: encima(?b1,?b2)

E: libre(?b2),sujeto(?b1)

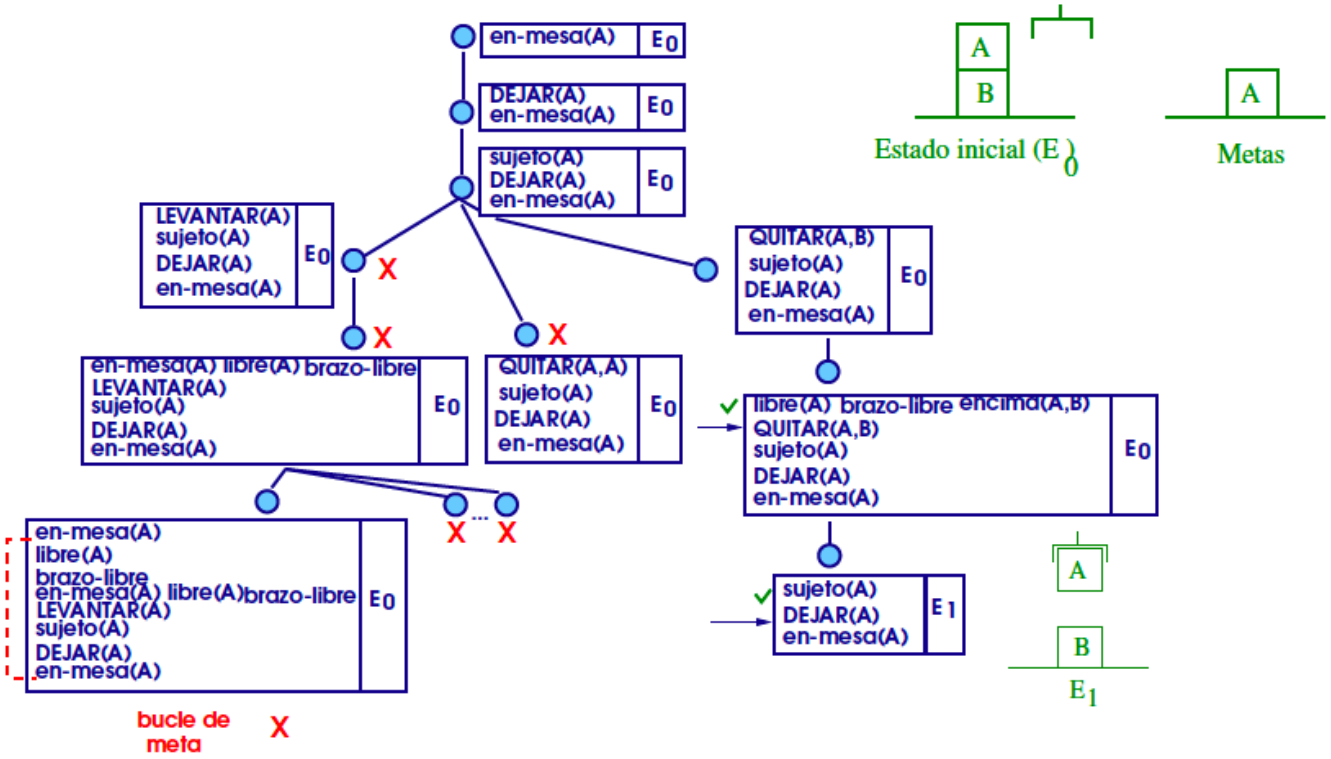




Búsqueda hacia atrás



Búsqueda hacia atrás



El problema de la linealidad en STRIPS

STRIPS asume independencia entre las metas, por lo que las trata linealmente: hasta que no encuentra un plan para obtener una meta, no pasa a las siguientes metas.

Método:

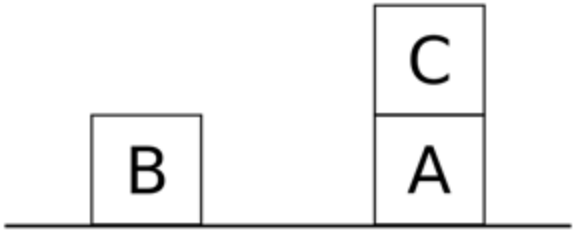
1. Divide las metas en submetas
2. Busca un subplan para cada meta
3. Concatena los subplanes

¡Anomalía de Sussman!

Anomalía de Sussman

La planificación de orden total tiene una limitación para poder dividir en subplanes, cuando los objetivos que debemos alcanzar en cada subplan interactúan entre sí.

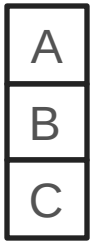
Estado inicial:



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

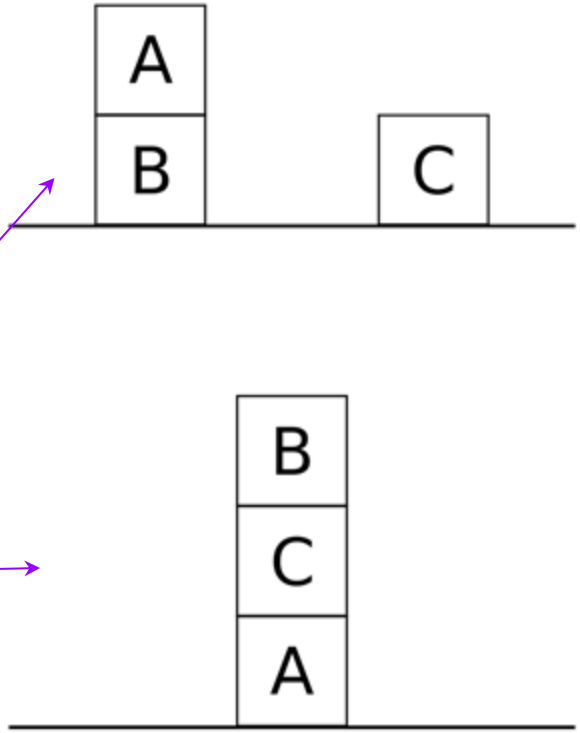
Meta:

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



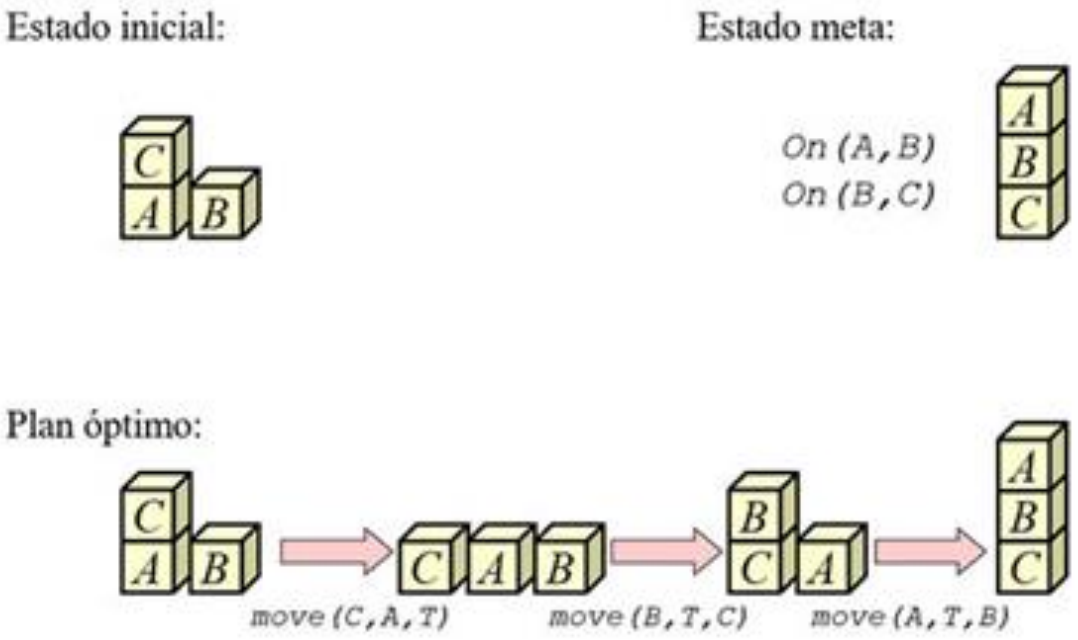
Sub metas:

1.On(A, B)
2.On(B, C)



Anomalía de Sussman

La planificación de orden total tiene principalmente una limitación cuando los objetivos que debemos alcanzar en un problema interactúan entre sí.



https://en.wikipedia.org/wiki/Sussman_anomaly

PDDL

PDDL

- El Planning Domain Description Language (PDDL) fue creado por Drew McDermott y su equipo. Se basaba en los estándares especificados de STRIPS
- El objetivo inicial era crear un lenguaje común y estándar de creación de planes para emplear como benchmark entre planificadores, y así poder comparar agentes de planificación en competiciones o estudios.
- En la actualidad es un estándar que presenta varias versiones, desde la 1.0 a la 3.1, cada una de ellas con diferentes niveles de expresividad
- Los planificadores parten de la especificación de un dominio y un problema en PDDL. Pueden soportar una u otras versiones de PDDL.

Algunos trucos para reducir expresividad de versiones avanzadas a versiones básicas de PDDL: https://maumagnaguagno.github.io/pddl

Definición del dominio

- El dominio es la información sobre el mundo en general
- En el dominio no aparecen proposiciones instanciadas

```
(define (domain DOMAIN_NAME)

(:requirements [:strips] [:equality] [:typing] [:adl])

(:predicates
          (PREDICATE_1_NAME [?A1 ?A2 ... ?AN])
          (PREDICATE_2_NAME [?A1 ?A2 ... ?AN]) ...)

(:action ACTION_1_NAME
          [:parameters (?P1 ?P2 ... ?PN)]
          [:precondition PRECOND_FORMULA]
          [:effect EFFECT_FORMULA] )
(:action ACTION_2_NAME ...)

...)
```

Ejemplo:

```
(:action LOAD-TRUCK
  :parameters (?obj ?truck ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
   (at ?truck ?loc) (at ?obj ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?truck)))
```

!! En PDDL agrupamos los effects y los deletes de STRIPS en :effect

Notación prefija para funciones: (and ?x ?y) (ver LISP)

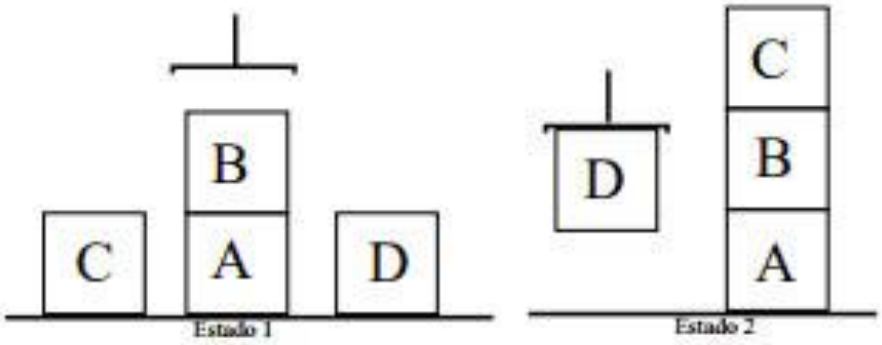
Definición del problema

- El problema es una instancia del mundo

```
(define (problem PROBLEM_NAME)
  (:domain DOMAIN_NAME)
  (:objects OBJ1 OBJ2 ... OBJ_N)
  (:init ATOM1 ATOM2 ... ATOM_N)
  (:goal CONDITION_FORMULA) )
```

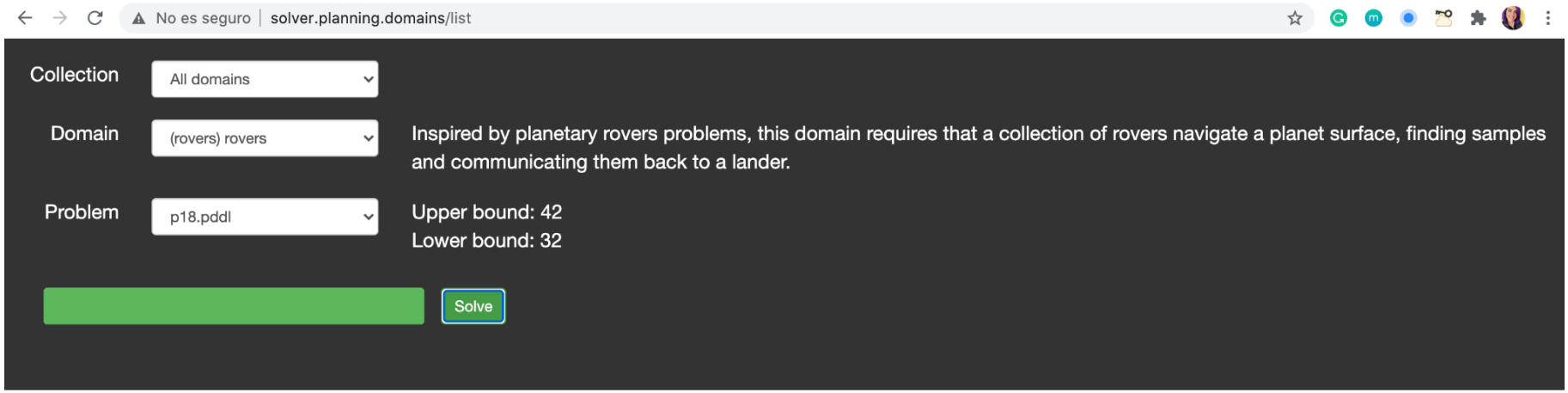
- En el problema de planificación aparecerán instanciadas la mayoría de las proposiciones. Contiene el estado inicial y meta
- Usamos elementos atómicos para formar literales que pueden ser afirmativos o negativos.
- Los **fluents** son átomos instanciados con objetos del estado del mundo.
- Al igual que en la descripción de STRIPS, la hipótesis del mundo cerrado hace que los fluents que no sean nombrados explícitamente en una descripción sean considerados como falsos.

Definición del objetivo



- Ejemplos de objetivos:
  - `SOBRE(B,A)`, `SOBRELAMESA(A)`, `-SOBRE(C,B)` es satisfecho por el estado 1 y no por el estado 2
  - `SOBRE(x,A)`, `DESPEJADO(x)`, `BRAZOLIBRE()` es satisfecho por el estado 1 pero no por el estado 2
  - `SOBRE(x,A)`, `SOBRE(y,x)` no es satisfecho por el estado 1 pero sí por el estado 2
  - El objetivo `SOBRE(x,A)`, `-SOBRE(C,x)` es satisfecho por el estado 1 pero no por el estado 2

planning.domains




```
--- OK.
Match tree built with 1837 nodes.

PDDL problem description loaded:
        Domain: ROVER
        Problem: ROVERPROB4621
        #Actions: 1837
        #Fluents: 370
Landmarks found: 11
Starting search with IW (time budget is 60 secs)...
rel_plan size: 32
#RP_fluents 36
Caption
{#goals, #UNnachieved,  #Achieved} -> IW(max_w)

{11/11/0}:IW(1) -> [2][3]rel_plan size: 30
#RP_fluents 33
{11/10/1}:IW(1) -> [2][3][4]rel_plan size: 30
#RP_fluents 33
{11/9/2}:IW(1) -> [2][3][4]rel_plan size: 28
#RP_fluents 31
{11/8/3}:IW(1) -> [2][3][4]rel_plan size: 25
#RP_fluents 28
{11/7/4}:IW(1) -> [2][3][4]rel_plan size: 23
```

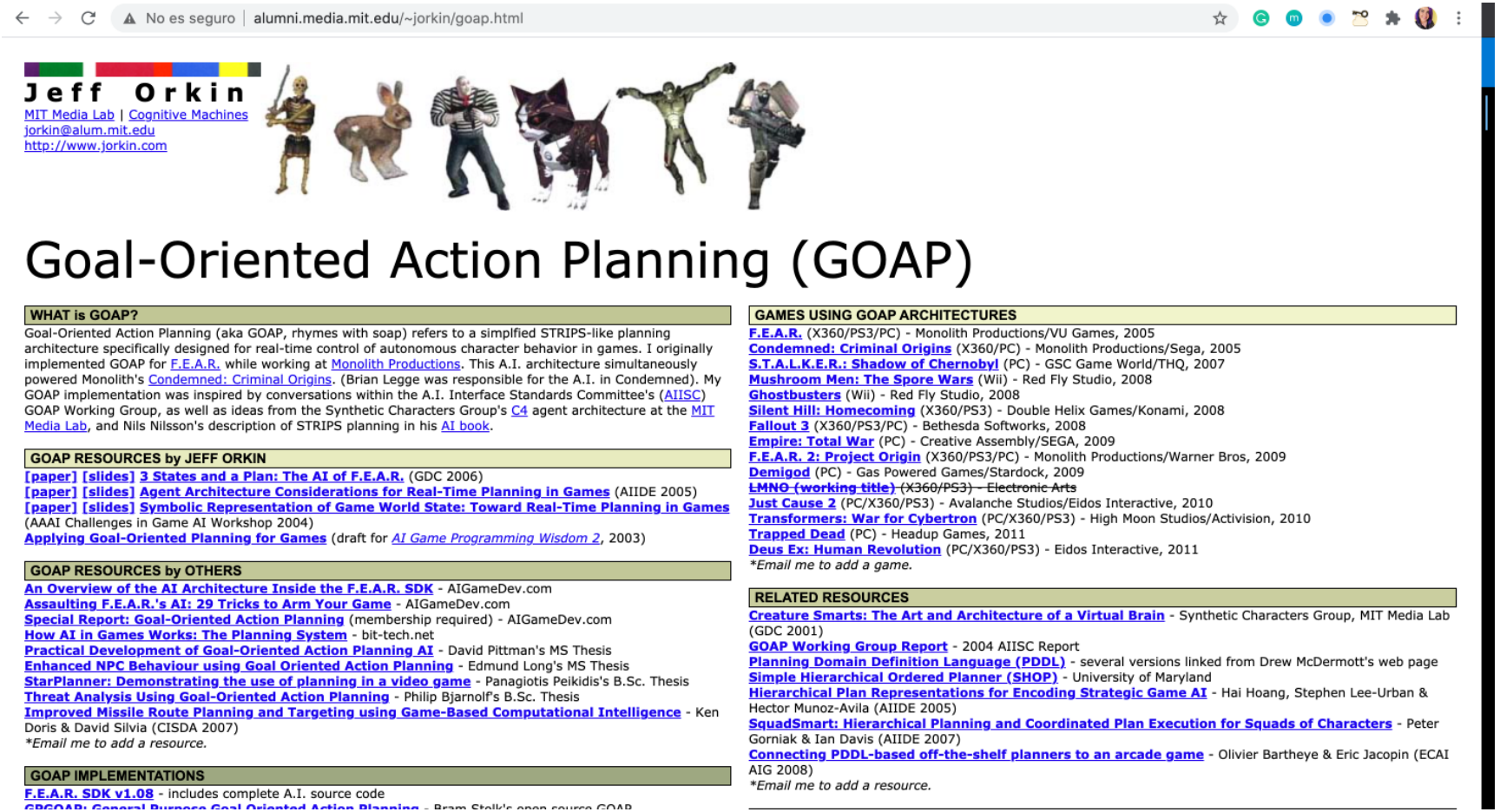
1. (sample_soil rover2 rover2store waypoint0)
2. (communicate_soil_data rover2 general waypoint0 waypoint0 waypoint17)
3. (sample_rock rover0 rover0store waypoint2)
4. (navigate rover0 waypoint2 waypoint0)
5. (communicate_rock_data rover0 general waypoint2 waypoint0 waypoint17)
6. (calibrate rover5 camera6 objective6 waypoint0)
7. (take_image rover5 waypoint0 objective3 camera6 low_res)
8. (communicate_image_data rover5 general objective3 low_res waypoint0 waypoint17)
9. (calibrate rover5 camera6 objective6 waypoint0)
10. (take_image rover5 waypoint0 objective4 camera6 high_res)
11. (communicate_image_data rover5 general objective4 high_res waypoint0 waypoint17)
12. (calibrate rover2 camera1 objective6 waypoint0)
13. (take_image rover2 waypoint0 objective2 camera1 colour)
14. (communicate_image_data rover2 general objective2 colour waypoint0 waypoint17)
15. (calibrate rover2 camera1 objective6 waypoint0)
16. (take_image rover2 waypoint0 objective5 camera1 colour)
17. (communicate_image_data rover2 general objective5 colour waypoint0 waypoint17)
18. (navigate rover1 waypoint9 waypoint4)
19. (sample_rock rover1 rover1store waypoint4)
20. (navigate rover1 waypoint4 waypoint11)
21. (communicate_rock_data rover1 general waypoint4 waypoint11 waypoint17)
22. (navigate rover5 waypoint0 waypoint1)
23. (navigate rover5 waypoint1 waypoint6)

GOAP

GOAP

http://alumni.media.mit.edu/~jorkin/goap.html

Sistemas en tiempo real



No es seguro | alumni.media.mit.edu/~jorkin/goap.html

Jeff Orkin
MIT Media Lab | Cognitive Machines
jorkin@alum.mit.edu
http://www.jorkin.com

Goal-Oriented Action Planning (GOAP)

WHAT is GOAP?

Goal-Oriented Action Planning (aka GOAP, rhymes with soap) refers to a simplfied STRIPS-like planning architecture specifically designed for real-time control of autonomous character behavior in games. I originally implemented GOAP for F.E.A.R. while working at Monolith Productions. This A.I. architecture simultaneously powered Monolith's Condemned: Criminal Origins. (Brian Legge was responsible for the A.I. in Condemned). My GOAP implementation was inspired by conversations within the A.I. Interface Standards Committee's (AIISC) GOAP Working Group, as well as ideas from the Synthetic Characters Group's C4 agent architecture at the MIT Media Lab, and Nils Nilsson's description of STRIPS planning in his AI book.

GOAP RESOURCES by JEFF ORKIN

[paper] [slides] **3 States and a Plan: The AI of F.E.A.R.** (GDC 2006)
[paper] [slides] **Agent Architecture Considerations for Real-Time Planning in Games** (AIIDE 2005)
[paper] [slides] **Symbolic Representation of Game World State: Toward Real-Time Planning in Games** (AAAI Challenges in Game AI Workshop 2004)
Applying Goal-Oriented Planning for Games (draft for *AI Game Programming Wisdom 2*, 2003)

GOAP RESOURCES by OTHERS

An Overview of the AI Architecture Inside the F.E.A.R. SDK - AIGameDev.com
Assaulting F.E.A.R.'s AI: 29 Tricks to Arm Your Game - AIGameDev.com
Special Report: Goal-Oriented Action Planning (membership required) - AIGameDev.com
How AI in Games Works: The Planning System - bit-tech.net
Practical Development of Goal-Oriented Action Planning AI - David Pittman's MS Thesis
Enhanced NPC Behaviour using Goal Oriented Action Planning - Edmund Long's MS Thesis
StarPlanner: Demonstrating the use of planning in a video game - Panagiotis Peikidis's B.Sc. Thesis
Threat Analysis Using Goal-Oriented Action Planning - Philip Bjarnolf's B.Sc. Thesis
Improved Missile Route Planning and Targeting using Game-Based Computational Intelligence - Ken Doris & David Silvia (CISDA 2007)
**Email me to add a resource.*

GOAP IMPLEMENTATIONS

F.E.A.R. SDK v1.08 - includes complete A.I. source code

GAMES USING GOAP ARCHITECTURES

F.E.A.R. (X360/PS3/PC) - Monolith Productions/VU Games, 2005
Condemned: Criminal Origins (X360/PC) - Monolith Productions/Sega, 2005
S.T.A.L.K.E.R.: Shadow of Chernobyl (PC) - GSC Game World/THQ, 2007
Mushroom Men: The Spore Wars (Wii) - Red Fly Studio, 2008
Ghostbusters (Wii) - Red Fly Studio, 2008
Silent Hill: Homecoming (X360/PS3) - Double Helix Games/Konami, 2008
Fallout 3 (X360/PS3/PC) - Bethesda Softworks, 2008
Empire: Total War (PC) - Creative Assembly/SEGA, 2009
F.E.A.R. 2: Project Origin (X360/PS3/PC) - Monolith Productions/Warner Bros, 2009
Demigod (PC) - Gas Powered Games/Stardock, 2009
~~**LMNO (working title)** (X360/PS3) - Electronic Arts~~
Just Cause 2 (PC/X360/PS3) - Avalanche Studios/Eidos Interactive, 2010
Transformers: War for Cybertron (PC/X360/PS3) - High Moon Studios/Activision, 2010
Trapped Dead (PC) - Headup Games, 2011
Deus Ex: Human Revolution (PC/X360/PS3) - Eidos Interactive, 2011
**Email me to add a game.*

RELATED RESOURCES

Creature Smarts: The Art and Architecture of a Virtual Brain - Synthetic Characters Group, MIT Media Lab (GDC 2001)
GOAP Working Group Report - 2004 AIISC Report
Planning Domain Definition Language (PDDL) - several versions linked from Drew McDermott's web page
Simple Hierarchical Ordered Planner (SHOP) - University of Maryland
Hierarchical Plan Representations for Encoding Strategic Game AI - Hai Hoang, Stephen Lee-Urban & Hector Munoz-Avila (AIIDE 2005)
SquadSmart: Hierarchical Planning and Coordinated Plan Execution for Squads of Characters - Peter Gorniak & Ian Davis (AIIDE 2007)
Connecting PDDL-based off-the-shelf planners to an arcade game - Olivier Bartheye & Eric Jacopin (ECAI AIG 2008)
**Email me to add a resource.*

GOAP

Posee cuatro diferencias con STRIPS:

- Establece costes a las acciones con el fin de poder asignar prioridad a unas acciones frente a otras, que es una consideración necesaria para el diseño de agentes en videojuegos.
- Elimina las listas de añadir y eliminar objetos y las convierte en una única lista de modificaciones del estado en la que las proposiciones pueden ser modificadas de manera más flexible que por simple operación *booleana*.
- Añade precondiciones procedurales que permiten mayor flexibilidad a la hora de expresar condiciones que se deban dar en el entorno para poder aplicar un operador. Ejecutan código.
- Añade efectos procedurales con la misma filosofía de poder modificar el entorno con mayor flexibilidad. Ejecutan código.

Recursos

http://planning.domains/#

Editor.planning.domains

http://solver.planning.domains/list

https://people.cs.pitt.edu/~milos/courses/cs1571-Fall2010/Lectures/Class18.pdf

www.unir.net