CSP

Constraint Satisfaction Problem

Introducción a los algoritmos de planeamiento

2018

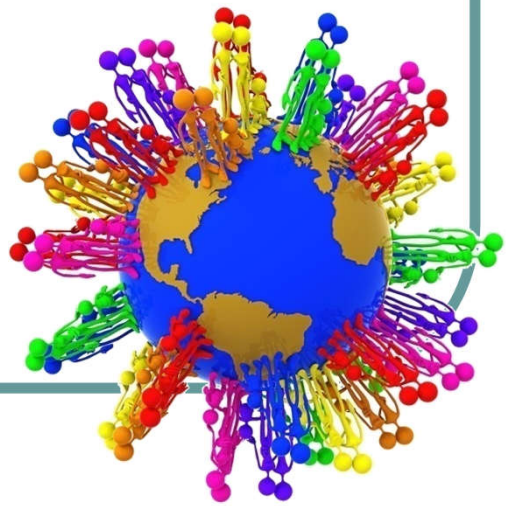
Definición general de CSP

- Un conjunto de variables con sus respectivos dominios.
- Un conjunto de restricciones sobre los valores que estas variables pueden tomar
- Problema:

"encontrar un valor para cada variable (dentro de su dominio) de manera tal que se cumplan todas las restricciones"

Satisfacción de Restricciones

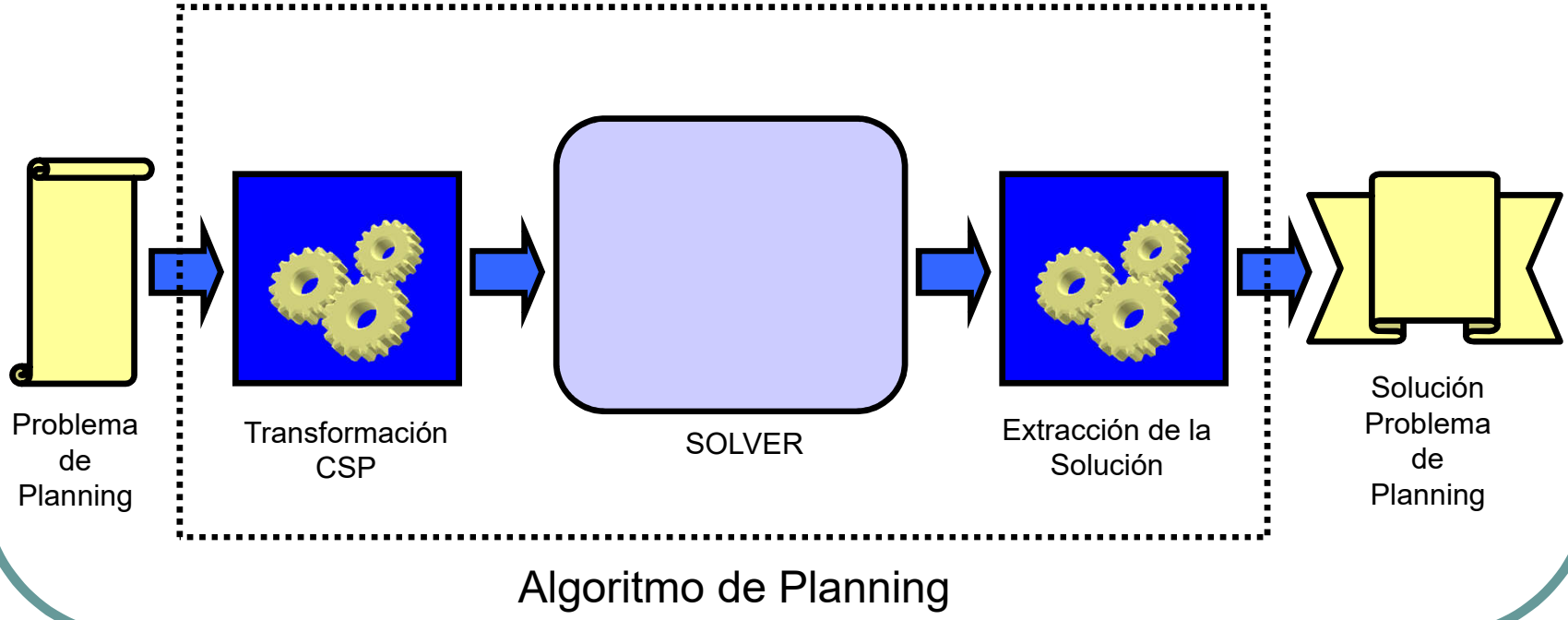
- Es un paradigma general de resolución de problemas que es aplicable a un conjunto amplio de áreas
  - Reconocimiento de patrones
  - Sistemas de soporte de decisiones
  - Planning
  - Scheduling



Procedimiento - Planning

- Similar al utilizado en SAT

También son utilizados en heurísticas y en extensiones



Definiciones CSP

- Sobre dominios finitos se lo define como:

  $P = (X, D, C)$

  $X = \{x_1, x_2, \ldots x_n\}$ un conjunto finito de n variables

  $D = \{D_1, D_2, \ldots D_n\}$ es el conjunto finito de dominio de las variables, $x_i \in D_i$

  $C = \{c_1, c_2, \ldots c_m\}$ conjunto finito de restricciones

Definiciones CSP

- Una solución al CSP *(X, D, C)* es una n-upla $(v_1 \ldots v_n)$ tal que $v_i \in D_i$ y los valores de las variables $x_i = v_i$ para todo $1 \leq i \leq n$ cumplen las restricciones en *C*

Restricciones

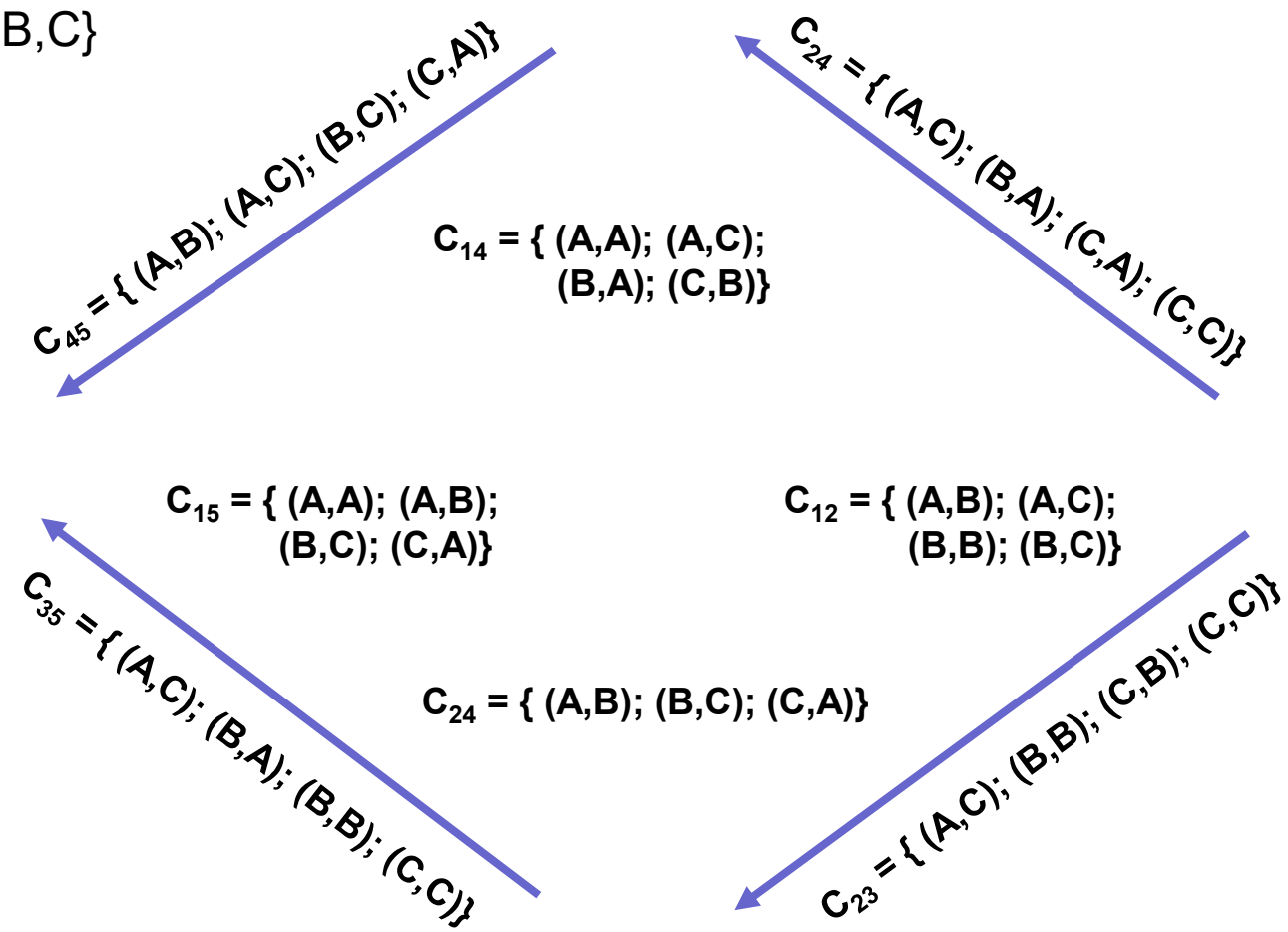
- Explicitas
  - Se lista el conjunto de tuplas permitidas, o el de tuplas prohibidas.
- Implicitas
  - Se utiliza uno o mas símbolos relacionales
- Universal
  - Es satisfecha por toda tupla (dentro de los dominios de las variables)
- Vacía
  - Todas las tuplas son prohibidas y no puede ser satisfecha.

CSP Binario

- CSP Binario
  - Todas las restricciones son relaciones binarias
  - Puede ser representado con o una red de restricciones
    - Cada Nodo es una variable X_i etiquetado con el Dominio de la variable
    - Cada arco arco (x_i, x_j) es etiquetado con la restriccion de las 2 variables.

CSP Binario

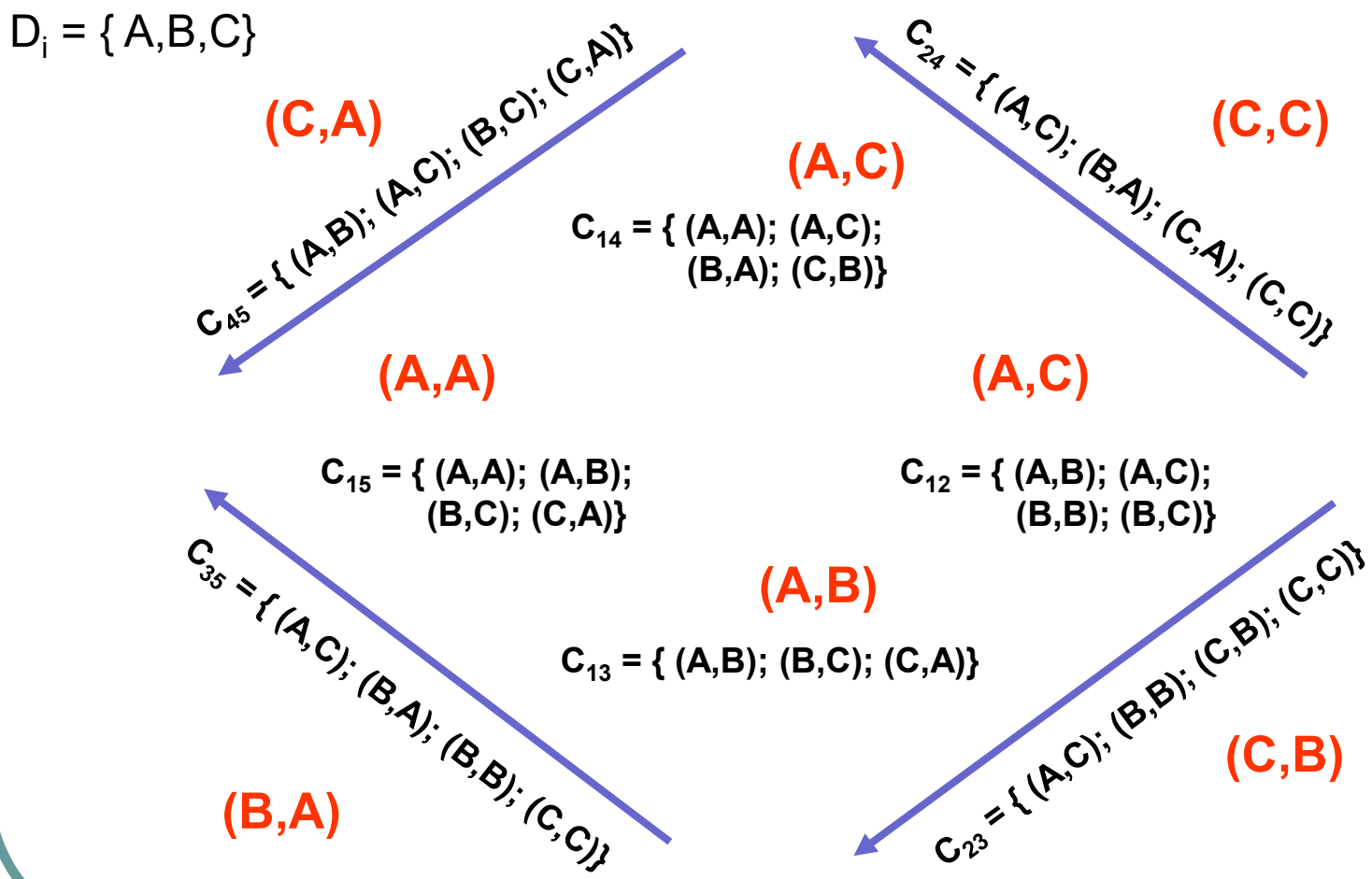
D_i = { A,B,C}



Solución = {A,C,B,C,A}

CSP Binario

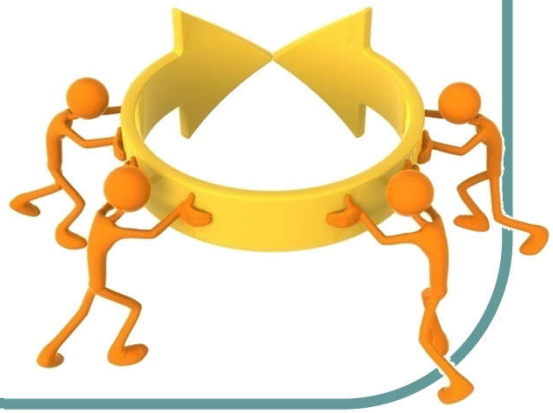
{A,B,B,A,B} {A,C,B,A,B}
{B,C,C,A,C}



Solución = {A,C,B,C,A}

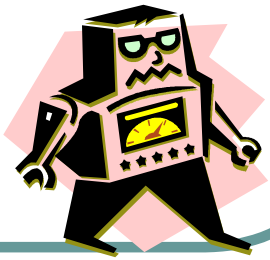
Definiciones

- Problema de planning (State-Variable)
  - Se utiliza otra representación para poder llevar el problema de planning a un problema CSP de manera más sencilla.
  - Se utiliza:
    - Constantes
    - Variables de objetos
    - Variables de estados
    - Relaciones



Símbolos constates

- Clases disjuntas que se corresponden con los objetos del dominio
  - Robots
  - Lugares
  - Contenedores



Variables de objetos

- Son variables tipadas, cada una sobre una clase o unión de clases del dominio
  - R $\in$ Robots
  - L $\in$ Lugares

Variables de estado

- Funciones del conjunto de estados y uno o mas conjunto de constantes a un conjunto de constantes

  - rloc: Robots x S → Lugares
  - rload: Robots x S → Contenedores **U** {nil}
  - cpos: Contenedores x S → Lugares **U** Robots

Relaciones

- Relaciones rígidas sobre las constates que no varían de estado a estado, para un dominio de planning dado
  - adyacentes(a , b)
  - impar(piso1)

Operadores

- Un operador se define como una tripla
  - O = (nombre(o), pre(o), eff(o))

  donde:

  pre(o) es un conjunto de expresiones que son condiciones sobre variables de estados y/o sobre relaciones rígidas

  eff(o) es un conjunto de asignaciones de valores a variables de estados

Ejemplo

- Contenedores , Robots y Lugares
  - move (R, L, M)
    - **Pre**: rloc(R) = L, adyacente(L, M)
    - **Eff**: rloc(R) ←M
  - load(C, R, L)
    - **Pre**: rloc(R) = L, cpos(C)=L, rload(R)=nil
    - **Eff**: rload(R)←C, cpos(C)←R
  - unload(C, R, L)
    - **Pre**: rloc(R) = L, rload(R) = C
    - **Eff**: rload(R) ← nil, cpos(C) ← L

Transformación

- Dada una representación State-Variable de problema de planning limitado a **k** pasos transformarlo a un problema CSP
  1) Definición de las variables de CSP
  2) Definición de las restricciones para S_o y S_g
  3) Codificación de los operadores
  4) Codificación de los axiomas

Paso 1: variables CSP

- Por cada variable de estado x_i sobre un dominio D_i y por cada $0 \leq j \leq k$ se convierte en una variable de CSP $x_i(j, v_u, \ldots v_m)$ cuyo dominio es D_i
- Por cada $0 \leq j \leq k-1$ existe una variable act(j) cuyo dominio es el conjunto de todas las posibles acciones en el dominio (se agrega no-op)

Paso 1: Ejemplo

- Dado el ejemplo anterior con un robot r1, tres contenedores c1, c2, c3 y tres lugares l1, l2, l3, con k=4 las variables en CSP quedarían definidas de la siguiente forma:
  - **rloc(j, r1)** E {l1, l2 , l3} para $0 \leq j \leq k$
  - **rload(j, r1)** E {c1, c2, c3, nil} para $0 \leq j \leq k$
  - **cpos(j, C)** E {l1, l2, l3, r1}, para C E {c1, c2, c3} para $0 \leq j \leq k$
  - **act(j)** E {move(R, L, M), load(C,R, L), unload(C,R,L), no-op} para todas las posibles instancias de los operadores y para $0 \leq j \leq k-1$

Paso 2: Codificación de S_0 y S_g

- Cada variable de estado x_i cuyo valor en s_0 es v_i se transforma a una restricción unaria de la correspondiente variable CSP para j = 0.

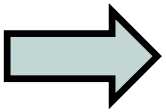
  $(x_i(0) = v_i)$

- Para S_g sucede lo mismo solo que j = k

  $(x_i(k) = v_i)$

Paso 2: Ejemplo

- Dados
  - S_0= {rloc(r1)=l1, rload(r1)=nil, cpos(c1)=l1, cpos(c2)=l2, cpos(c3)=l2}
  - S_g= {cpos(c1)=l2, cpos(c2)=l1}



Se agregarían las siguientes restricciones:

- rloc(0,r1)=l1, rload(0,r1)=nil, cpos(0,c1)=l1, cpos(0,c2)=l2, cpos(0,c3)=l2
- cpos(4,c1)=l2, cpos(4,c2)=l1

Paso 3: Las acciones

- De instancias de acciones a restricciones
  - $a(v_u,..v_m)$ inst. de un operador o
  - $v_u,..v_m$ cumplen las relaciones rígidas de o

  $\forall$ j, $0 \leq j \leq k-1$
    - Toda condición $x_i = v_i$ en pre(a) se transforma a una restricción de la forma:

      $(act(j) = a(v_u,..v_m), x_i(j) = v_i)$

    - Toda condición $(x_i \in D'_i)$ en pre(a) se transforma a una restricción de la forma:

      $(act(j) = a(v_u,..v_m), x_i(j) = v_i \mid v_i \in D'_i)$

Paso 3: Las acciones

$\forall$ j, $0 \leq j \leq k-1$

- Toda asignación $x_i \leftarrow v_i$ en eff(a) se transforma a una restricción de la forma:

$$(act(j) = a(v_u,..v_m), x_i(j + 1) = v_i)$$

Paso 3: Ejemplo

- move (R, L, M)
  - **Pre**: rloc(R) = L, adyacente(L, M)
  - **Eff**: rloc(R) ←M

Restricciones:

{(act(j) = move(R,L,M), rloc(j, R)=L |
adyacente(L, M) 0≤ j≤ 3)}

{(act(j) = move(R,L,M), rloc(j+1,R)=M |
adyacente(L, M) 0≤ j≤ 3)}

Paso 3: Ejemplo

- load(C, R, L)
  - **Pre**: rloc(R) = L, cpos(C)=L, rload(R)=nil
  - **Eff**: rload(R)←C, cpos(C)←R

Restricciones:

{(act(j) = load(C,R,L), rloc(j, R)=L | 0≤ j≤ 3)}

{(act(j) = load(C,R,L), rload(j, R)=nil | 0≤ j≤ 3)}

{(act(j) = load(C,R,L), cpos(j, C)=L | 0≤ j≤ 3)}

{(act(j) = load(C,R,L), rload(j+1, R)=C | 0≤ j≤ 3)}

{(act(j) = load(C,R,L), cpos(j+1, C)=R | 0≤ j≤ 3)}

Paso 4: Axiomas

- toda variable que es invariante para una acción a permanece sin cambios en el siguiente paso
  - $\forall\ x_i \in inva(a)$

$\{ (act(j) = a(vu,..vm),\ x_i(j)=v_i\ ,\ x_i(j+1)=v_i \mid v_i \in Di\ \}$

Caso especial la acción no-op

Paso 4: Ejemplo

- Dado el ejemplo para la acción move las variables invariantes son rload y cpos

{(act(j)=move(R,L,M), rload(j,R)=v, rload(j+1,R)=v | adyacente(L, M) $v \in D_{rload}$)}

{(act(j)=move(R,L,M), cpos(j,C)=v, cpos(j+1,C)=v | adyacente(L, M) $v \in D_{cpos}$)}

D_{rload} = {c1, c2, c3, nil}

D_{cpos} = {l1, l2, l3, r1}

Paso 4: Axiomas

- Condición de exclusión mutua

No es necesaria, debido a que la variable act(j) solo puede tener un único valor para cada j

Extracción de la solución

- La solución al problema es una tupla σ la cual asigna un valor a cada variable del problema (si es que la solución existe)
- Los valores que tenga la variable act(j) para cada j determinan la secuencia de acciones

$$\pi = \langle a_1, a_2, \ldots a_k \rangle \mid a_i = act(i-1)$$

CSP Vs PSP

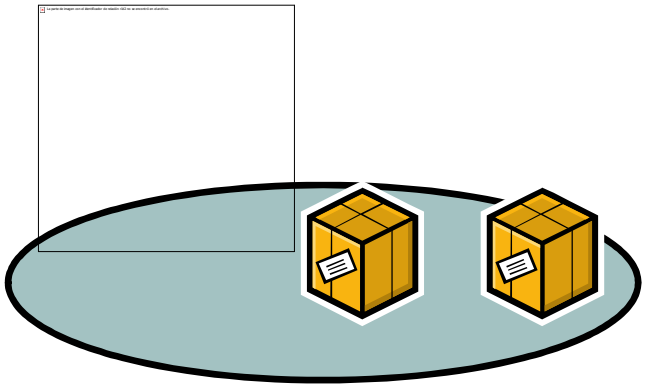
- Codificaciones similares
- Cambia la forma de representar el problema de planning (State-Variable vs relaciones)
- Axioma de exclusión mutua
- **Ambos son NP-Completos**

CSP

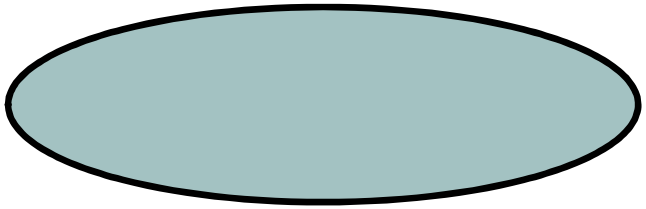
- La representación es muy similar al problema de planning lo cual permite incorporar conocimiento de control a los solvers.
- Se utiliza dentro de otras formas de resolver el problema de planning
  - Relaciones de Binding de las variables
  - Relaciones Mutex (GP)

Preguntas

Robots

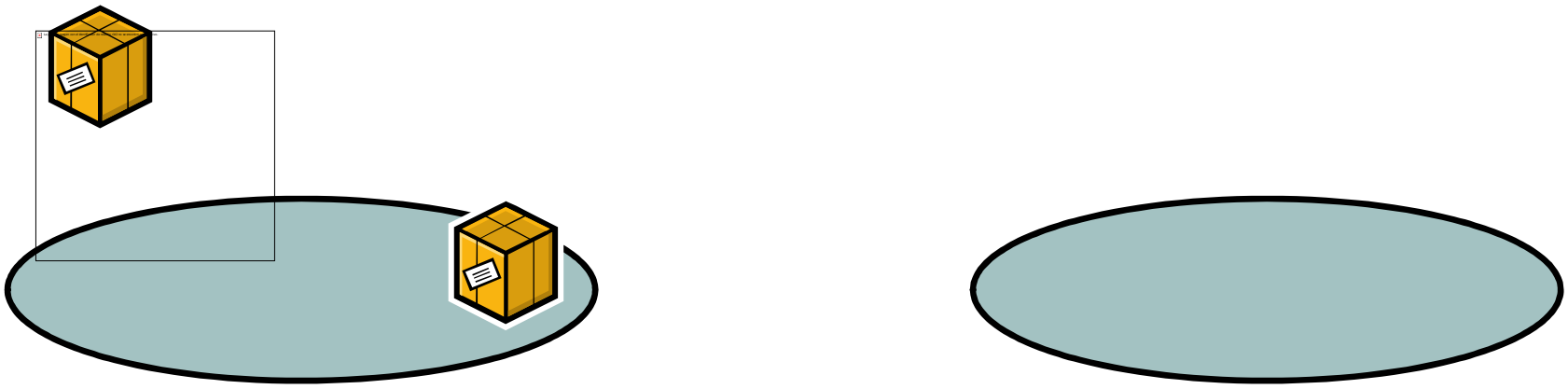


Lugar 1

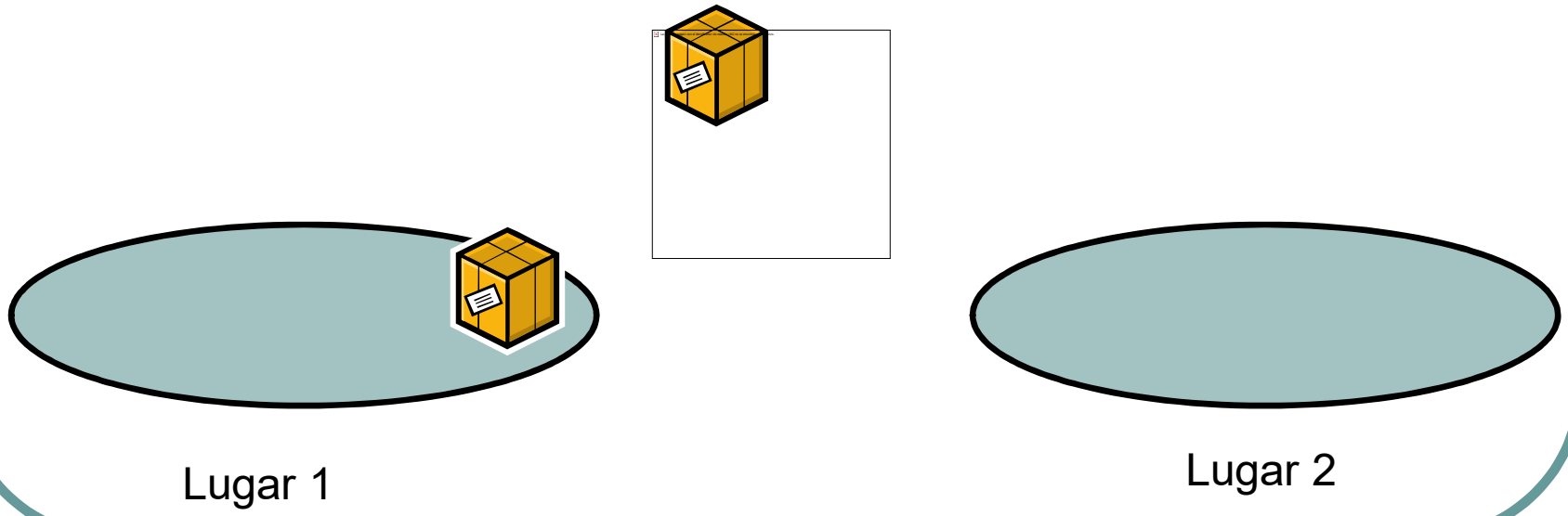


Lugar 2

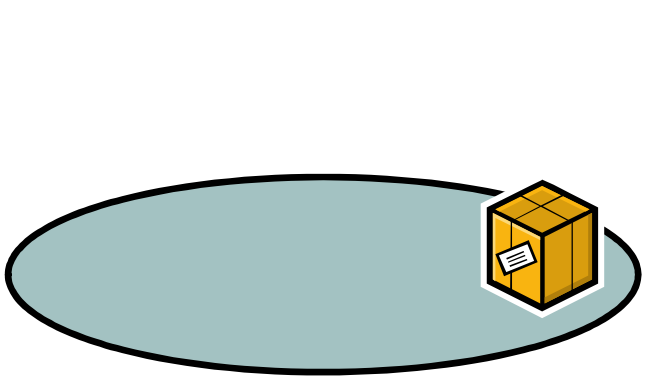
Robots



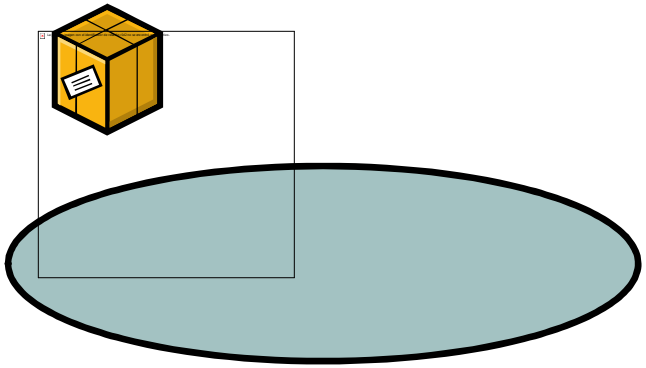
Robots



Robots

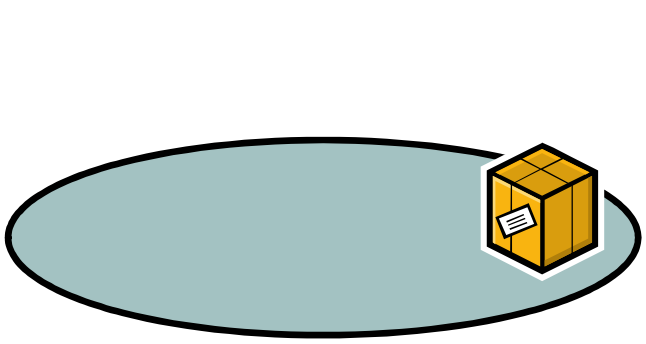


Lugar 1

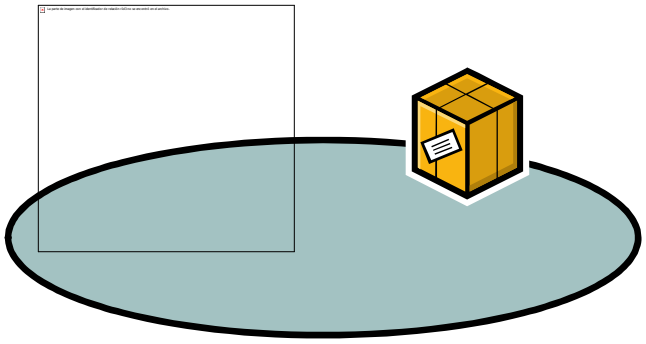


Lugar 2

Robots



Lugar 1



Lugar 2

Robos