Representación del conocimiento

Representación clásica

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

24 de junio de 2021

Problema clásico de planeación

Temas

Problema clásico de planeación

El *problema clásico de planeación* es una versión simplificada de ambientes donde se desean resolver problemas de planeación:

- Se asume que el mundo se encuentra en un estado estático S.
- Sólo se transforma a otro estado estático cuando un solo agente realiza alguna acción de entre un conjunto preestablecido $a \in A$.
- El problema de planeación $\mathcal{P}$ consiste en encontrar una secuencia de acciones $\pi =< a_1, ..., a_k >$ que transformen al estado inicial del mundo s_i en alguno de un conjunto dado de estados meta $s \vDash g$.

Temas

Elementos para la descripción del estado

Definición (Lenguaje)

Sea $\mathcal{L}$ un lenguaje de primer orden con:

- un número finito de símbolos de predicados,
- símbolos constantes y
- **sin** símbolos funcionales ($\Rightarrow$) los estados posibles son finitos.

Estado

Definición (Estado)

Un *estado* es un conjunto de átomos aterrizados p de $\mathcal{L}$[a], bajo la *hipótesis del mundo cerrado*.

[a] Conjunción de átomos que son verdaderos en ese estado $\mathrm{p} \in \mathrm{s}$.

Relaciones flexibles y rígidas

Flexible Un predicado aterrizado cuyo valor de verdad puede variar entre estados es una *relación flexible* o *fluída*.

Rígido Un predicado aterrizado invariante al estado es una *relación rígida*.

Temas

Operadores y Acciones

- *Operador* $o = (\text{nombre}(o), \text{precondiciones}(o), \text{efectos}(o))$.

  $\text{nombre}(o)$ $n(x_1, ..., x_k)$, con n el símbolo del operador y x_i variables.

  $\text{precondiciones}(o)$ $\text{precondiciones}^+(o)$ y $\text{precondiciones}^-(o)$. Relaciones rígidas y/o flexibles.

  $\text{efectos}(o)$ $\text{efectos}^+(o)$ y $\text{efectos}^-(o)$. Sólo relaciones flexibles.

- *Acción* Una acción es una instancia aterrizada de un operador de planeación.

La acción a es *aplicable* en el estado s sí $\text{precondiciones}^+(a) \subseteq s$ y $\text{precondiciones}^-(a) \cap s = \emptyset$.

Temas

Función objetivo

La meta g está dada por una lista de *literales (átomos y átomos negados)*, de modo que un estado s satisface g ($s \vDash g$) si existe una sustitución σ tal que:

1. Cada literal positiva de $\sigma(g)$ está en s.
2. Ninguna literal negada de $\sigma(g)$ se encuentra en s.

Problema de planeación

- El *dominio* para el sistema de planeación clásica es un sistema de transiciones restringido $\Sigma = (S, A, \gamma)$ con
  - S atomos aterrizados en $\mathcal{L}$.
  - A todas las instancias aterrizadas de los operadores en el conjunto O.
  - $\gamma(s, a)$ la función de transición.
- Un *problema de planeación clásica* es $\mathcal{P} = (\Sigma, s_0, g)$, en el *dominio* Σ, con estado inicial s_0 y meta g.

  Obsérvese que la especificación de $\mathcal{L}$ define las constantes y átomos en el dominio.
- Las *declaración* de $\mathcal{P}$ es la **especificación sintáctica** utilizada para describir $\mathcal{P}$ a un programa de computadora:

$$P = (O, s_0, g) \tag{1}$$

  donde O es el conjunto de operadores y s_0 y g son conjuntos de literales aterrizadas.

Ejemplo

Robots en el almacén

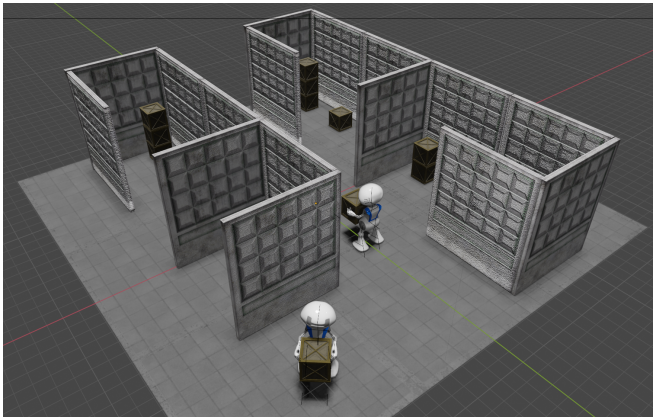

Figura: Robots acomodan cajas en pilas en un almacén.

Dominio “Robots en el almacén”

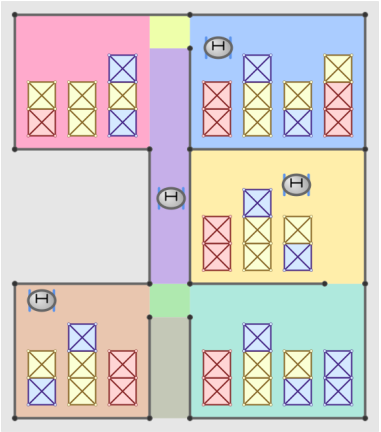

Figura: Robots trabajadores en un almacén, versión abstracta.

Elementos:

- Robots $\mathtt{r}$
- Colores $\mathtt{a}$
- Cajas de colores $\mathtt{c}$
- Pilas de cajas $\mathtt{p}$
- Bodegas $\mathtt{b}$
- Regiones $\mathtt{l}$

Operadores/Acciones

mover(r,l,l') al robot r de la región l a alguna región desocupada adyacente l'.

cargar(c,r,p,l) la caja c con el robot desocupado r de la cima de la pila p, estando todos en la región l.

poner(c,k,p,l) la caja c, sostenida por el robot r, en la cima de la pila p, estando todos en la región l.

PDDL

Temas

PDDL

- *PDDL* son las siglas de *Planning Domain Description Language*.
- Es un lenguaje para describir dominios de planeación e instancias concretas de problemas.
- No indica cómo es que los problemas serán resueltos, para ello se necesita una *máquina de inferencias*.

Temas

PDDL y la declaración de dominios

```
1  ;; Especificación en PDDL1 del dominio "robots en el almacén"
2
3  (define (domain almacen-robotizado)
4    (:requirements :strips :typing)
5    (:types
6      robot    ;sostiene máximo una caja, sólo uno por región.
7      color    ;las cajas pueden ser de diferentes colores.
8      caja     ;contiene los materiales almacenables.
9      pila     ;en una bodega, tiene una base y una pila de cajas.
10     bodega   ;región dónde se pueden acomodar pilas de cajas.
11     región   ;posiciones conectadas en las pueden estar los robots.
12   )
13 ...)
```

Predicados

```
...
;; Los predicados describen el estado

(:predicates
  (adyacente ?l1  ?l2 - región)        ; la región ?l1 es adyacente a ?l2
  (adjunta ?p - pila ?b - bodega)      ; la pila ?p está en ?b
  (ubicada ?b - bodega ?l - región)    ; la bodega ?b está en la región ?l
  (tipo ?c - caja ?a - color)          ; la caja ?c es color ?a

  (en ?r - robot ?l - región)          ; el robot ?r está en la región ?l
  (ocupada ?l - región)                ; hay un robot en la región ?l
  (cargando ?r - robot ?c - caja )     ; el robot ?r está cargando la caja ?c
  (descargado ?r - robot)              ; el robot ?r no lleva carga

  (almacenada ?c - caja ?p - pila); la caja ?c está almacenada en la pila ?p
  (tope ?c - caja ?p - pila)     ; la caja ?c está hasta arriba de la pila ?p
  (sobre ?c1 - caja ?c2 - caja)        ; la caja ?c1 está sobre la caja ?c2
)
...
```

Operadores

```
1   ...
2   ;; los operadores del dominio son plantillas que indican lo que
3   ;; pueden hacer los agentes:
4
5   ;; mueve al robot entre dos regiones adyacentes
6   (:action mover
7     :parameters (?r - robot ?desde ?hasta - región)
8     :precondition (and (adyacente ?desde ?hasta) (en ?r ?desde)
9                        (not (ocupada ?hasta)))
10    :effect (and (en ?r ?hasta)
11                 (not (en ?r ?desde))
12                 (ocupada ?hasta)
13                 (not (ocupada ?desde))  ))
```

Operadores (cont)

```
;; el robot carga una caja de una pila
(:action cargar
  :parameters (?r - robot ?c - caja ?p - pila)
  :vars (?l - región ?b - bodega ?otra - caja)
  :precondition (and (desocupado ?r) (en ?r ?l)
                     (ubicada ?b ?l) (adjunta ?p ?b)
                     (almacenada ?c ?p) (tope ?c ?p) (sobre ?c ?otra))
  :effect (and (cargando ?r ?c) (not (desocupado ?r))
               (tope ?otra ?p) (not (tope ?c ?p))
               (not (almacenada ?c ?p)) (not (sobre ?c ?otra)) ))

;; pone una caja en una pila cercana
(:action poner
  :parameters (?r - robot ?c - caja ?p - pila)
  :vars (?l - región ?b - bodega ?otra - caja)
  :precondition (and (cargando ?r ?c) (en ?r ?l)
                     (ubicada ?b ?l) (adjunta ?p ?l) (tope ?otra ?p))
  :effect (and (almacenada ?c ?p) (tope ?c ?p) (sobre ?c ?otra)
               (not (tope ?otra ?p)) (not (cargando ?r ?c)) (desocupado ?k))))
```

Temas

PDDL y la declaración de problemas I

```
;; un problema sencillo con 1 robot y 2 bodegas
(define (problem almacen1)
  (:domain almacen-robotizado)

  (:objects
   r1 - robot
   amarilla azul roja - color
   l1 l2 - región
   b1 b2 - bodega
   p1 q1 p2 q2 - pila
   ca cb cc cd ce cf base - caja)

  (:init
   ; Descripción del almacén
   (adyacente l1 l2) (adyacente l2 l1)
   (ubicada b1 l1)  (ubicada b2 l2)
   (adjunta p1 b1)  (adjunta q1 b1)
   (adjunta p2 b2)  (adjunta q2 b2)
```

PDDL y la declaración de problemas II

```
19
20      ; Robot
21      (en r1 l1)
22      (descargado r1)
23      (ocupada l1)
24
25      ; Cajas
26      (tipo ca amarilla)  (tipo cb amarilla)  (tipo cc amarilla)
27      (tipo cd azul)  (tipo ce azul)
28      (tipo cf roja)
29
30      ; Descripción del acomodo de las cajas en las pilas
31      ;; Pila p1
32      (almacenada ca p1)  (almacenada cb p1)  (almacenada cc p1)
33      (sobre ca base)  (sobre cb ca)  (sobre cc cb)
34      (top cc p1)
35
36      ;; Pila p2
37      (almacenada cd q1)  (almacenada ce q1)  (almacenada cf q1)
```

PDDL y la declaración de problemas III

```
38      (sobre cd base)  (sobre ce cd)  (sobre cf ce)
39      (top cf q1)
40
41      ;; Pilas en la bodega b2
42      (top base p2)
43      (top base q2)
44    )
```

PDDL y la descripción de la meta

```
1    ...
2    ;; el objetivo es mover a todas las cajas a la bodega b2
3    ;; ca and cc a la pila p2, el resto a q2
4    (:goal
5      (and (almacenada ca p2)
6                   (almacenada cb q2)
7                   (almacenada cc p2)
8                   (almacenada cd q2)
9                   (almacenada ce q2)
10                  (almacenada cf q2)))
11 )
```

STRIPS

Temas

STanford Research Institute Problem Solver (STRIPS)

- STRIPS es un **generador** automático de planes y un **monitor** para la ejecución del plan desarrollado en los 1960-70.
- Asume que sólamente ocurre una acción a la vez y que sus efectos son instantáneos.
- Utiliza un lenguaje de primer orden para representar el dominio de un problema: sus estados posibles y operadores.
- Utiliza un demostrador de teoremas por resolución para determinar si una meta es verdadera (i.e. se cumple) en un estado dado.
- Fue utilizado por el robot Shakey, el cual navegaba en un ambiente con cajas como obstáculos y múltiples habitaciones interconectadas.

(Fikes y Nilsson s.f.)

Lenguaje para la especificación del dominio

Definición

Sea $\mathcal{L}$ un lenguaje de primer orden con un número finito de símbolos de predicados, símbolos constantes y sin símbolos funcionales.

- Un *estado en un dominio de planeación STRIPS* es un conjunto de átomos de $\mathcal{L}$[a].
- Un *operador* es una regla de la forma:

```
if <precondiciones>
then (retract <eliminaciones>) (assert  <adiciones>)
```

donde las *precondiciones*, *eliminaciones* y *adiciones* se especifican con listas de símbolos en $\mathcal{L}$.

[a] Conjunción de átomos que son verdaderos en ese estado.

Hipótesis STRIPS

La especificación de los estados cumple que:

- Un átomo aterrizado p *es verdadero* en el *estado* $s \Leftrightarrow p \in s$[1].
- *Metas:* s *satisface* un conjunto de literales[2] aterrizadas g ($s \vDash g$) si:
  1. Cada literal positiva en g se encuentra en s.
  2. Cada literal negativa en g no se encuentra en s.

Para afrontar el **problema del marco**, en adición a la *hipótesis del mundo cerrado*, este sistema de planeación trabaja bajo la hipótesis STRIPS:

> *Un operador de planeación sólo afecta aquellos aspectos del mundo mencionados explícitamente en sus listas de adiciones y remociones de hechos.*
> *(Fikes y Nilsson s.f.)*

[1] Hipótesis del mundo cerrado.
[2] Átomos y átomos negados

Resolución de problemas mediante búsqueda hacia atrás

El mecanismo para la búsqueda de planes usado por STRIPS es predecesor del *encadenamiento hacia atrás*, una técnica de búsqueda que va de la meta hacia el estado inicial.

- Considera que un operador es *relevante* a la satisfacción de la meta cuando reduce la diferencia entre las descripciones del estado actual y los requisitos para un estado meta.
- Usa técnicas de reconocimiento de patrones para encontrar operadores cuyas listas de adiciones sean relevantes para reducir la diferencia entre demostraciones exitosas de que se alcanzó un estado meta y demostraciones fallidas; ignorando los efectos de las listas de eliminaciones.
- Por ello sólo elige un operador que satisfaga las precondiciones pendientes.
- Sólo se busca satisfacer las precondiciones del último operador agregado.
- Es incompleto por ambas razones.
- No siempre encuentra la solución óptima.

Temas

Monitor de la ejecución

- Cuando un robot como Shakey ejecutaba el plan en el ambiente real, el estado obtenido frecuentemente difería del estado esperado por el planeador.
- El *monitor* verificaba, después de la ejecución de cada paso, si era posible continuar o si se necesitaba *replanear*.
- En cada paso, se tenía un *núcleo* de sentencias que debían ser verdaderas para que el resto del plan aún fuera exitoso.
- Ya que las consecuencias inesperadas podían ser benéficas también, el monitor verificaba, después de cada paso, si algunas de las precondiciones pendientes no habían sido satisfechas ya, incluyendo a la meta misma.
- Dado que este monitor evitaba replanear lo más posible, si era posible generar un mejor plan desde un estado inesperado, su política de continuar con el mismo plan no era la más eficiente.
- Ahora otros sistemas de planeación incluyen *acciones condicionales* que revisan las acciones del núcleo e indican qué hacer según el resultado de la prueba.

Referencias I

- Fikes, Richard E. y Nils J. Nilsson (s.f.). «STRIPS, a retrospective». En: *Artificial Intelligence* 59 (), págs. 227-232.
- Ghallab, Malik, Dana Nau y Paolo Traverso (2004). *Automated Planning, Theory and Practice*. Morgan Kaufmann Publishers.

Licencia