Planificación

Inteligencia Artificial

UNIVERSIDAD EAFIT®

Planificación clasica

La planificación clásica se define como la tarea de encontrar una secuencia de acciones para lograr un objetivo en un entorno discreto, determinista, estático y completamente observable. Hemos visto un enfoque para esta tarea: el agente de resolución de problemas.

Planificación clasica

En respuesta a estas limitaciones, los investigadores de planificación han invertido en una representación factorizada utilizando una familia de lenguajes llamada PDDL, el Lenguaje de Definición de Dominios de Planificación (Ghallab et al. 1998), que nos permite expresar todas las acciones con un solo esquema de acción y no necesita conocimientos específicos del dominio.

Planificación clasica

En PDDL, un estado se representa como una conjunción de fluentes atómicos ground.

"**Ground**": significa sin variables
"**fluent**": significa un aspecto del mundo que cambia con el tiempo
"ground atomic": significa que hay un solo predicado y, si hay argumentos, deben ser constantes.

EJEMPLO

Fluentes individuales:
At(Truck1, Melbourne) → El camión está en Melbourne.
At(Package1, Sydney) → El paquete está en Sídney.
Loaded(Package1, Truck1) → El paquete está en el camión.

Estado:
At(Truck1, Melbourne) ∧ At(Package1, Sydney) ∧ Loaded(Package1, Truck1)

Planificación clasica

Los siguientes fluentes **no están permitidos** en un estado:

- **At(x, y)** (porque tiene variables),
- **¬Poor** (porque es una negación),
- **At(Spouse(Ali), Sydney)** (porque usa un símbolo de función, **Spouse**).

Planificación clasica-Esquema de Acción

Un esquema de acción representa una familia de acciones ground. Por ejemplo, aquí hay un esquema de acción para volar un avión de un lugar a otro:

$$\text{Action}(\text{Fly}(p, from, to)),$$

$$\text{Precond} : At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$$

$$\text{Effect} : \neg At(p, from) \land At(p, to)$$

Planificación clasica-Esquema de Acción

Si en nuestro dominio tenemos dos aviones (P1, P2) y tres aeropuertos (SFO, JFK, LAX), entonces podríamos generar múltiples acciones ground a partir del esquema de acción:

1. **Fly(P1, SFO, JFK)**
2. **Fly(P1, SFO, LAX)**
3. **Fly(P1, JFK, LAX)**
4. **Fly(P2, SFO, JFK)**
5. **Fly(P2, LAX, JFK)**
6. **Fly(P2, JFK, SFO)**

... y así sucesivamente.

Cada una de estas es una acción ground específica, y todas forman parte de la familia de acciones descrita por el esquema Fly(p, from, to).

Planificación clasica-Esquema de Acción

Una acción *alpha* ground **es aplicable** en el estado s si s satisface la precondición de *alpha* ; es decir, si cada literal positivo en la precondición está en s y cada literal negado no lo está.

$$\text{Action}(\text{Fly}(p, from, to)),$$

$$\text{Precond} : At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$$

$$\text{Effect} : \neg At(p, from) \land At(p, to)$$

El **resultado** de ejecutar la acción aplicable aaa en el estado sss se define como un estado s′, que está representado por el conjunto de fluentes formados comenzando con s.

$$s = \{At(P_1, SFO), Plane(P_1), Airport(SFO), Airport(JFK)\}$$

$$\text{RESULT}(s, a) = (s - \text{DEL}(a)) \cup \text{ADD}(a).$$

Planificación clasica-Problemas

Un **problema específico** dentro del dominio se define con la adición de un estado inicial y un objetivo. El **estado inicial** es una conjunción de fluentes ground:

$$Init\{\ \}$$

El **objetivo** (introducido con *Goal*) es similar a una precondición: una conjunción de literales (positivos o negativos) que pueden contener variables

$$At(C_1, SFO) \land \neg At(C_2, SFO) \land At(p, SFO)$$

Se refiere a cualquier estado en el que la carga $\mathbf{C_1}$ esté en **SFO**, pero $\mathbf{C_2}$ no lo esté, y en el que haya un avión en **SFO**..

Planificación clasica-Ejemplos

Imagina un problema de transporte de carga aérea que implica cargar y descargar carga y volar de un lugar a otro.

El problema se puede definir con tres acciones: **Load** (Cargar), **Unload** (Descargar) y **Fly** (Volar).

Las acciones afectan dos predicados:
•**In(c, p)** significa que la carga **c** está dentro del avión **p**.
•**At(x, a)** significa que el objeto **x** (ya sea un avión o una carga) está en el aeropuerto **a**.

El siguiente plan es una solución para el problema:

$$[Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK), Unload(C_1, P_1, JFK),$$
$$Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO), Unload(C_2, P_2, SFO)].$$

Planificación clasica-Ejemplos

$$
\begin{aligned}
&Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK) \\
&\quad \land Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2) \\
&\quad \land Airport(JFK) \land Airport(SFO)) \\
&Goal(At(C_1, JFK) \land At(C_2, SFO)) \\
&Action(Load(c, p, a), \\
&\quad \text{PRECOND}: At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
&\quad \text{EFFECT}: \neg At(c, a) \land In(c, p)) \\
&Action(Unload(c, p, a), \\
&\quad \text{PRECOND}: In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
&\quad \text{EFFECT}: At(c, a) \land \neg In(c, p)) \\
&Action(Fly(p, from, to), \\
&\quad \text{PRECOND}: At(p, from) \land Plane(p) \land Airport(from) \land Airport(to) \\
&\quad \text{EFFECT}: \neg At(p, from) \land At(p, to))
\end{aligned}
$$

Planificación clasica-Ejemplos

Imagina un problema de cambiar una llanta pinchada

El objetivo es tener una llanta de repuesto en buen estado montada correctamente en el eje del automóvil, donde el estado inicial es que hay una llanta pinchada en el eje y una llanta de repuesto en el maletero.

Solo hay cuatro acciones:
1.Sacar la llanta de repuesto del maletero.
2.Retirar la llanta pinchada del eje.
3.Colocar la llanta de repuesto en el eje.
4.Dejar el automóvil desatendido durante la noche.

La solución al problema es la siguiente:

$$[Remove(Flat, Axle), Remove(Spare, Trunk), PutOn(Spare, Axle)]$$

Planificación clasica-Ejemplos

$$
\begin{aligned}
&Init(Tire(Flat) \land Tire(Spare) \land At(Flat, Axle) \land At(Spare, Trunk)) \\
&Goal(At(Spare, Axle)) \\
&Action(Remove(obj, loc), \\
&\quad \text{PRECOND: } At(obj, loc) \\
&\quad \text{EFFECT: } \neg At(obj, loc) \land At(obj, Ground)) \\
&Action(PutOn(t, Axle), \\
&\quad \text{PRECOND: } Tire(t) \land At(t, Ground) \land \neg At(Flat, Axle) \land \neg At(Spare, Axle) \\
&\quad \text{EFFECT: } \neg At(t, Ground) \land At(t, Axle)) \\
&Action(LeaveOvernight, \\
&\quad \text{PRECOND:} \\
&\quad \text{EFFECT: } \neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk) \\
&\qquad \land \neg At(Flat, Ground) \land \neg At(Flat, Axle) \land \neg At(Flat, Trunk))
\end{aligned}
$$

Inteligencia Artificial

https://stability.ai/blog/stable-diffusion-public-release

https://openai.com/blog/chatgpt/

https://ai.googleblog.com/2022/06/minerva-solving-quantitative-reasoning.html

https://www.youtube.com/watch?v=jMvLCZBXbtc

https://www.deepmind.com/blog/tackling-multiple-tasks-with-a-single-visual-language-model

https://www.deepmind.com/blog/building-safer-dialogue-agents

https://www.deepmind.com/publications/a-generalist-agent

https://www.deepmind.com/research/highlighted-research/alphago

https://ai.facebook.com/research/cicero/

https://openai.com/blog/whisper/

https://valle-demo.github.io/