

# Planejamento Automático

Jomi F. Hübner

Universidade Federal de Santa Catarina  
Departamento de Automação e Sistemas  
<http://jomi.das.ufsc.br/ia>



# Contexto

- ▶ único agente racional, que deve **escolher** ações que levam o estado atual para um estado objetivo
- ▶ um ambiente completamente observável e determinístico

# Planejamento

- ▶ **estado** atual e objetivo na forma de conjunção de literais
- ▶ **ações** com pré-condições e efeitos
- ▶ ambos descritos com uma **linguagem** específica (STRIPS, PDDL, ...)
- ▶ É mais fácil definir o problema em linguagem específica e declarativa (que definir estados e transições em linguagens de propósito geral)
- ▶ Possui algoritmos e heurísticas **gerais** e **eficientes**

# Especificação dos **Estados**

- ▶ conjunção de literais positivos sem variáveis

- ▶ exemplos:

*Poor*  $\wedge$  *Unknown*

*At(Truck<sub>1</sub>, Melbourne)*  $\wedge$  *At(Truck<sub>2</sub>, Sydney)*

- ▶ mundo fechado: o que não é dito é falso

# Especificação dos **Ações**

Exemplo:

$Fly(p, from, to)$

Pre-condições:

$At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$

Efeitos:

$\neg At(p, from) \wedge At(p, to)$

- ▶ Pre-condições: em que estados a ação se aplica
- ▶ Efeitos: o que muda no estado (adições e remoções)

# Especificação de um **Problema** de Planejamento

**Init**  $At(C_1, SFO) \wedge At(C_2, JFK) \wedge At(P_1, SFO) \wedge$   
 $At(P_2, JFK) \wedge Cargo(C_1) \wedge Cargo(C_2) \wedge$   
 $Plane(P_1) \wedge Plane(P_2) \wedge Airport(SFO) \wedge Airport(JFK)$

**Goal**  $At(C_1, JFK) \wedge At(C_2, SFO)$

**Load**( $c, p, a$ ) **if**  $At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$   
 $\rightsquigarrow \neg At(c, a) \wedge In(c, p)$

**Unload**( $c, p, a$ ) **if**  $In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$   
 $\rightsquigarrow: \neg In(c, p) \wedge At(c, a)$

**Fly**( $p, f, t$ ) **if**  $At(p, f) \wedge Plane(p) \wedge Airport(f) \wedge Airport(t) \wedge f \neq t$   
 $\rightsquigarrow: \neg At(p, f) \wedge At(p, t)$

## └ Especificação de um **Problema** de Planejamento

```

init At(C1, SFO) ∧ At(C2, JFK) ∧ At(P1, SFO) ∧
    At(P2, JFK) ∧ Cargo(C1) ∧ Cargo(C2) ∧
    Plane(P1) ∧ Plane(P2) ∧ Airport(SFO) ∧ Airport(JFK)

Goal At(C1, JFK) ∧ At(C2, SFO)

Load(c, p, a) iff At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    ⇔ ¬At(c, a) ∧ ¬At(p, a)

Unload(c, p, a) iff ¬At(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    ⇔ ¬At(c, p) ∧ At(c, a)

Fly(p, f, t) iff At(p, f) ∧ Plane(p) ∧ Airport(f) ∧ Airport(t) ∧ f ≠ t
    ⇔ ¬At(p, f) ∧ At(p, t)
  
```

essa é a “programação” da “aplicação”, não precisa codificar nada em java, python, ....

qual a solução, quantas, qual melhor....  
como?

# Resolução de um Problema de Planejamento

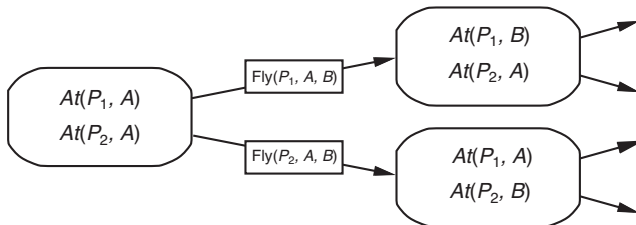
Dado um problema de planejamento (Init, Goal, Actions), encontrar uma sequencia de ações que leva do estado atual ao estado objetivo.



# Solução com busca – progressão

do estado atual para o objetivo

- ▶ para cada estado  $s$  a explorar, tentar todas as ações  $a$  que tem suas pre-condições implicadas pelo estado  $s$   
 $s \models PRE(a)$
- ▶ o novo estado é o antigo mais a aplicação dos efeitos da ação  
 $s' = s + ADD(a) - DEL(a)$
- ▶ termina quando o estado objetivo é alcançado

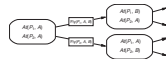


## └ Solução com busca – **progressão**

### Solução com busca – **progressão**

do estado atual para o objetivo

- para cada estado  $s$  a explorar, tentar todas as ações  $a$  que tem suas pre-condições implicadas pelo estado  $s$   
 $s \models PRE(a)$
- o novo estado é o antigo mais a aplicação dos efeitos da ação  
 $s' = s \div ADD(a) \div DEL(a)$
- termina quando o estado objetivo é alcançado



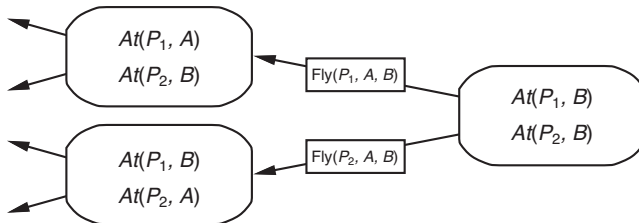
apesar de poucas ações, o problema são os argumentos. Tem que fazer todas as combinações parâmetro/valor!

$O(v^k)$ ,  $v$  é nro de args da ação,  $k$  o número de valores.

# Solução com busca – regressão

do estado objetivo para o atual

- ▶ o estado atual é o objetivo da busca  
o estado objetivo é o início da busca
- ▶ para cada estado  $s$  a explorar, tentar todas as ações  $a$  que tenham efeitos que contribuem para o estado  
$$\exists_{e \in ADD(a)} e \in s \wedge \neg \exists_{e \in DEL(a)} e \in s$$
- ▶ o novo estado é o antigo acrescido das pre-condições da ação e sem as adições da ação  
$$s' = s + PRE(a) - ADD(a)$$
- ▶ termina quando o estado atual é alcançado pela busca

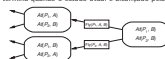


## └ Solução com busca – regressão

### Solução com busca – regressão

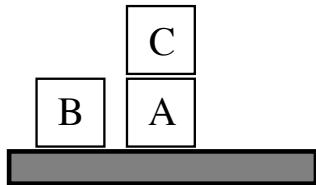
do estado objetivo para o atual

- o estado atual é o objetivo da busca
- o estado objetivo é o início da busca
- para cada estado  $s$  a explorar, tentar todas as ações  $a$  que tenham efeitos que contribuem para o estado
- $\exists_{e \in ADD(a)} e \in s \wedge \neg \exists_{e \in DEL(a)} e \in s$
- o novo estado é o antigo acrescido das pre-condições da ação e sem as adições da ação  
 $s' = s + PRE(a) - ADD(a)$
- termina quando o estado atual é alcançado pela busca

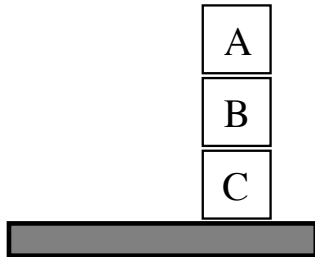


diminui os número de antecessores por causa da unificação entre objetivo/efeito  $b$

## Exercício: mundo dos blocos

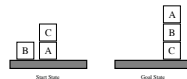


Start State



Goal State

# └ Exercício: mundo dos blocos



não dá pra usar quantificadores,  $\neg \exists x On(x, b)$ , então se usa  $Clear(b)$ . distinguir move de moveToTable para lidar com efeitos diferentes.

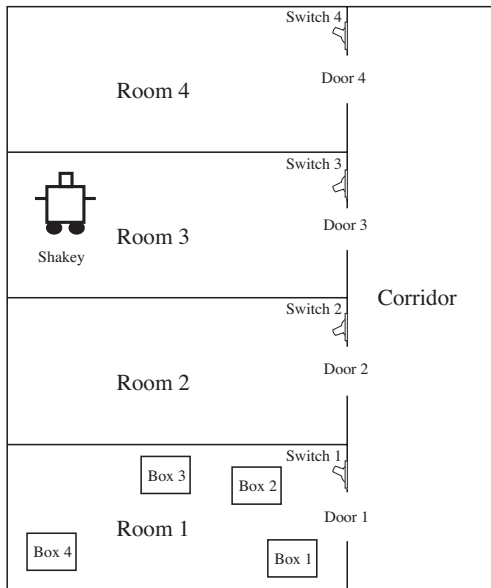
Init  $On(A, Table), On(B, Table), On(C, A),$   
 $Block(A), Block(B), Block(C), Clear(B), Clear(C)$

Goal  $On(A, B), On(B, C)$

$Move(b, x, y)$  if  $On(b, x), Clear(b), Clear(y), Block(b), Block(y),$   
 $b \neq x, b \neq y, x \neq y$   
 $\rightsquigarrow On(b, y), Clear(x), \neg On(b, x), \neg Clear(y)$

$MoveToTable(b, x)$  if  $On(b, x), Clear(b), Block(b), b \neq x$   
 $\rightsquigarrow On(b, Table), Clear(x), \neg On(b, x)$

# Exercício: Shakey Robot



Objetivo:  $Box_2$  na sala  $Room_2$

Ações: robô ir de um lugar para outro, robô empurrar uma caixa de um lugar para outro, subir ou descer de uma caixa, ligar ou desligar uma lâmpada.

# Material de **consulta**

- ▶ RUSSEL & NORVIG. **Artificial Intelligence: a modern approach**. 3rd Ed. 2010. Ch. 10–11.  
<http://aima.cs.berkeley.edu>
- ▶ GALLAB & NAU & TRAVERSO. **Automated Planning and Acting**. 2016.  
<http://projects.laas.fr/planning>