Planejamento Automático

Jomi F. Hübner

Universidade Federal de Santa Catarina
Departamento de Automação e Sistemas
`http://jomifred.github.io/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:
  $Poor \wedge Unknown$
  $At(Truck_1, Melbourne) \wedge At(Truck_2, Sydney)$
- mundo fechado: o que não é dito é falso

Especificação dos Ações

Exemplo:

$$Fly(p, from, to)$$

Pre-condições:

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

Efeitos:

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

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

Especificação de um Problema de Planejamento

$$
\begin{aligned}
\textbf{Init} \quad & At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land \\
& At(P_2, JFK) \land Cargo(C_1) \land Cargo(C_2) \land \\
& Plane(P_1) \land Plane(P_2) \land \\
& Airport(SFO) \land Airport(JFK)
\end{aligned}
$$

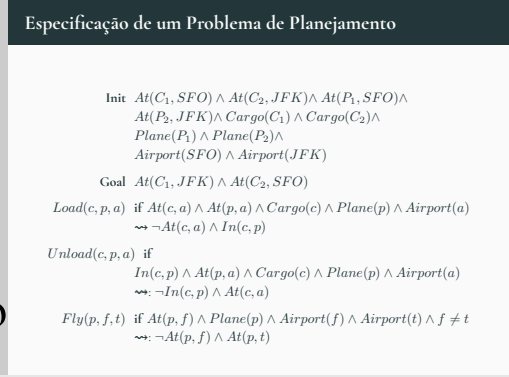
$$
\textbf{Goal} \quad At(C_1, JFK) \land At(C_2, SFO)
$$

$$
\begin{aligned}
Load(c, p, a) \quad & \textbf{if } At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
& \rightsquigarrow \neg At(c, a) \land In(c, p)
\end{aligned}
$$

$$
\begin{aligned}
Unload(c, p, a) \quad & \textbf{if} \\
& In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a) \\
& \rightsquigarrow: \neg In(c, p) \land At(c, a)
\end{aligned}
$$

$$
\begin{aligned}
Fly(p, f, t) \quad & \textbf{if } At(p, f) \land Plane(p) \land Airport(f) \land Airport(t) \land f \neq t \\
& \rightsquigarrow: \neg At(p, f) \land At(p, t)
\end{aligned}
$$

Especificação de um **Problema** de Planejamento



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

qual a solução, quantas, qual melhor....

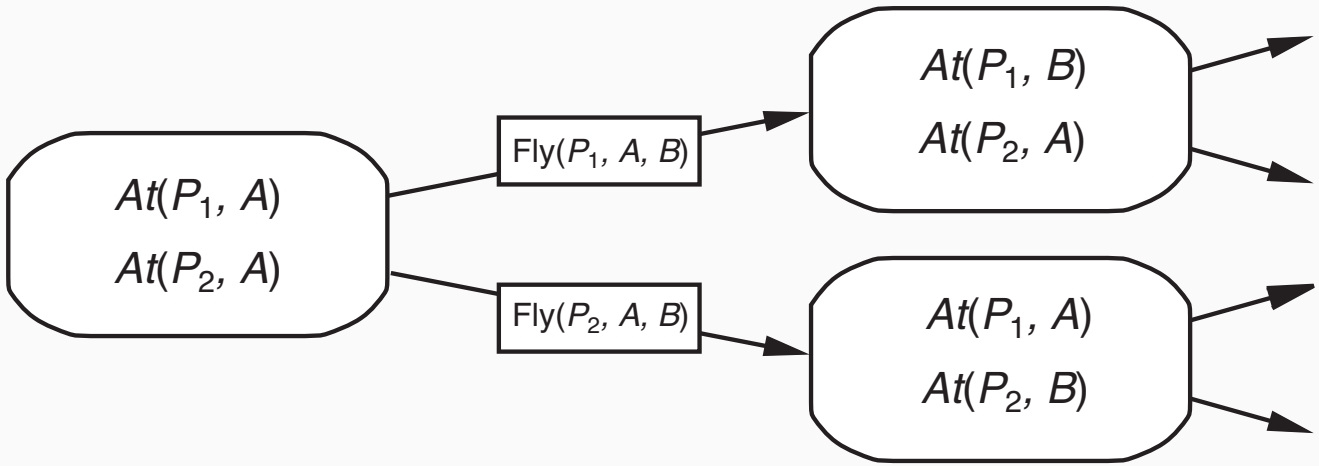
como?

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.

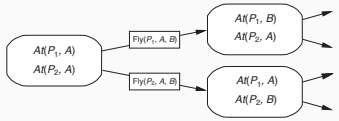
Solução com busca – progressão

- 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

- 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

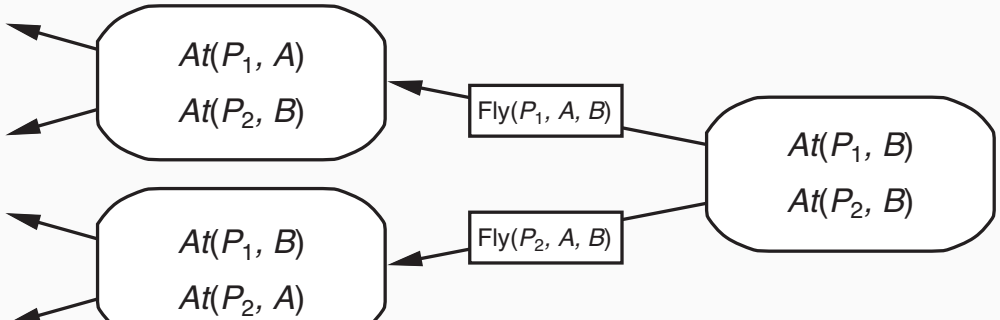


apesar de poucas ações, o problema são os argumentos. Tem que fazer todas as combinações parêmetro/valor!

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

Solução com busca – regressão

- o estado atual é o objetivo da busca
  o estado objetivo é o inicio da busca
- para cada estado s a explorar, tentar todas as ações a que tenham efeitos que contribuem para o estado
  $\exists_{e \in ADD(a)}\ e\theta \in s \wedge \neg \exists_{e \in DEL(a)}\ e \in s$
- 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



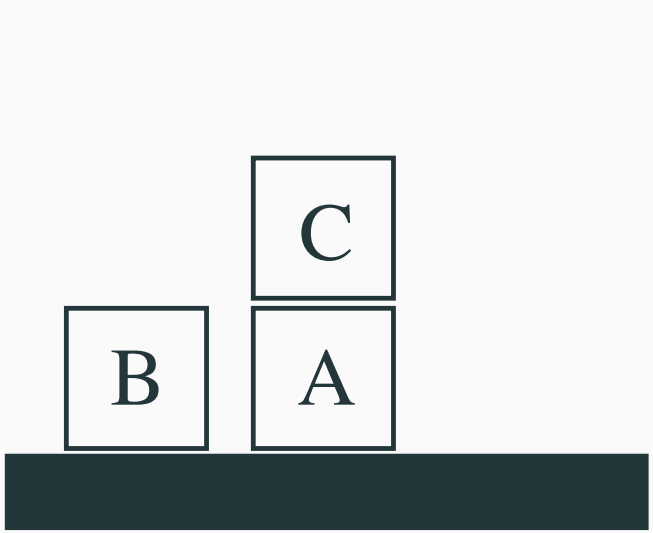
Solução com busca – regressão

Solução com busca – regressão

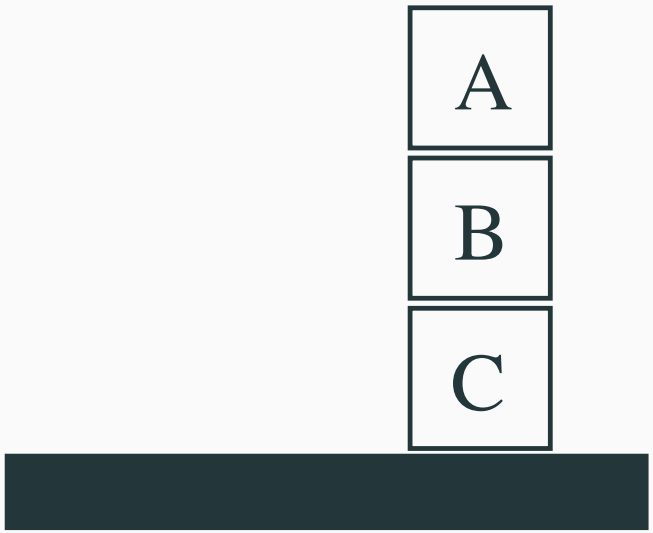
- o estado atual é o objetivo da busca
  o estado objetivo é o inicio da busca
- para cada estado s a explorar, tentar todas as ações a que tenham efeitos que contribuem para o estado
  $\exists_{e \in ADD(a)}\ e\theta \in s \wedge \neg\exists_{e \in DEL(a)}\ e \in s$
- o novo estado é o antigo acrescido 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

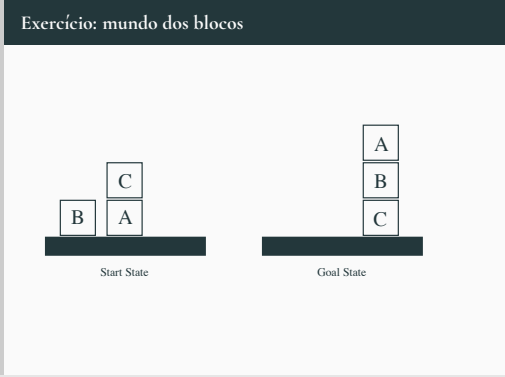


Start State



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.

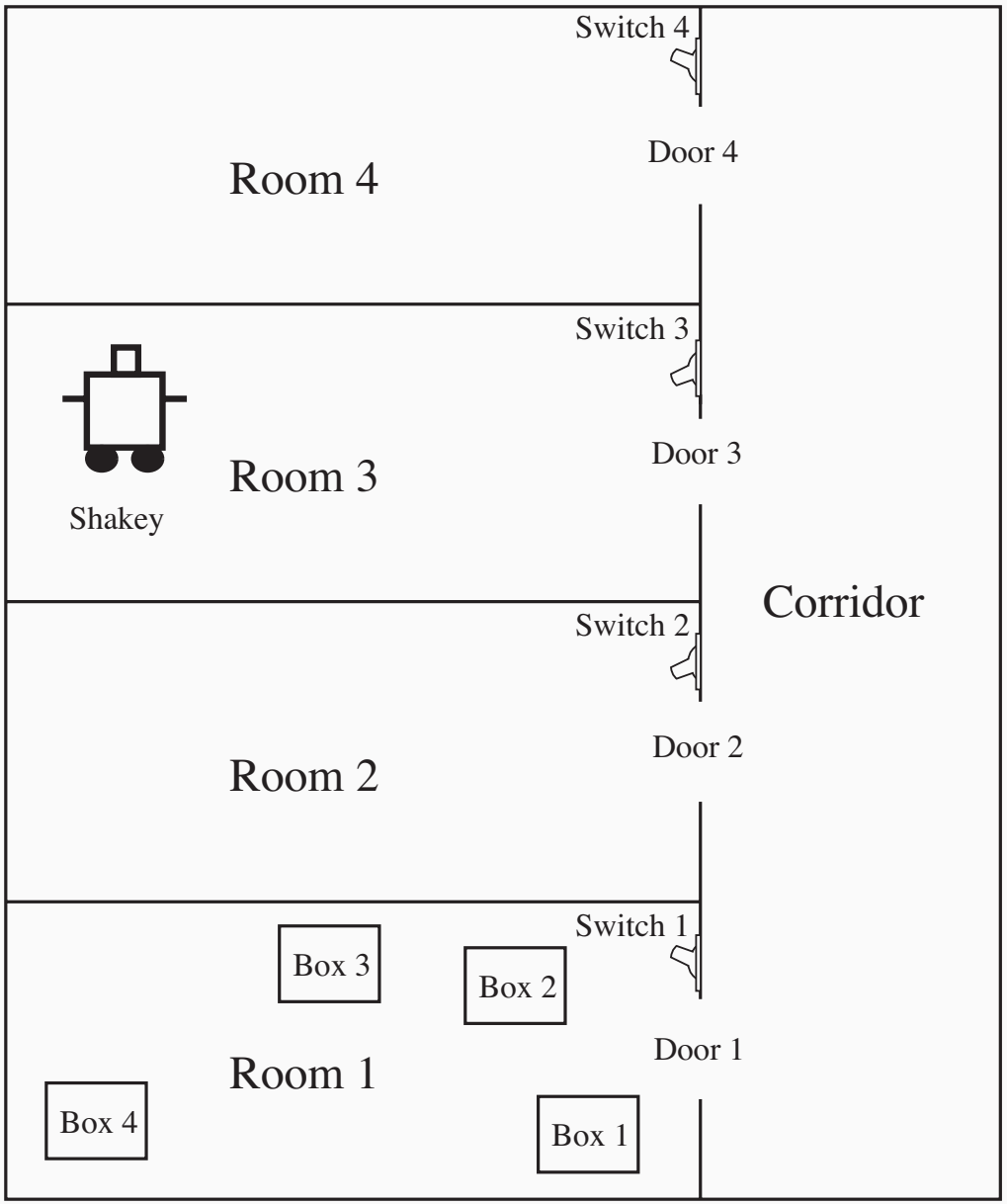
Init $On(A, Table), On(B, Table), On(C, A),$
$Block(A), Block(B), Block(C), Clear(B), Clear(C)$

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`