Planejamento Clássico

Sergio Queiroz *

Sistemas Inteligentes – 2017.1

*Adaptado dos slides de Alan Fern (Oregon State University), que por sua vez foram parcialmente baseados em slides por Daniel Weld.

Alguns problemas de planejamento em IA

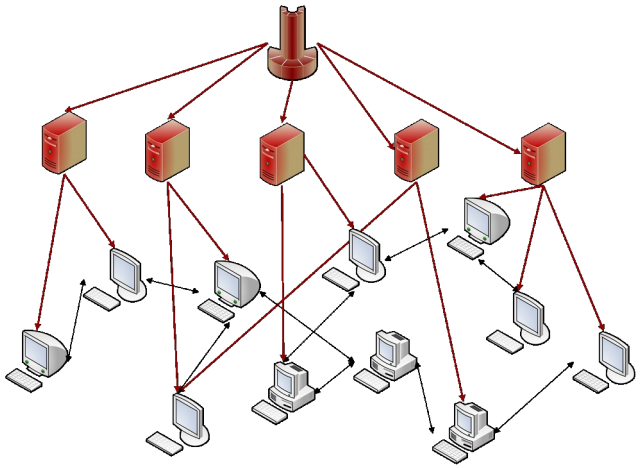
Planejamento para resposta a Incêndio, Resgate

Paciência

Jogos de Estratégia

Controle de Helicóptero

Controle de robôs

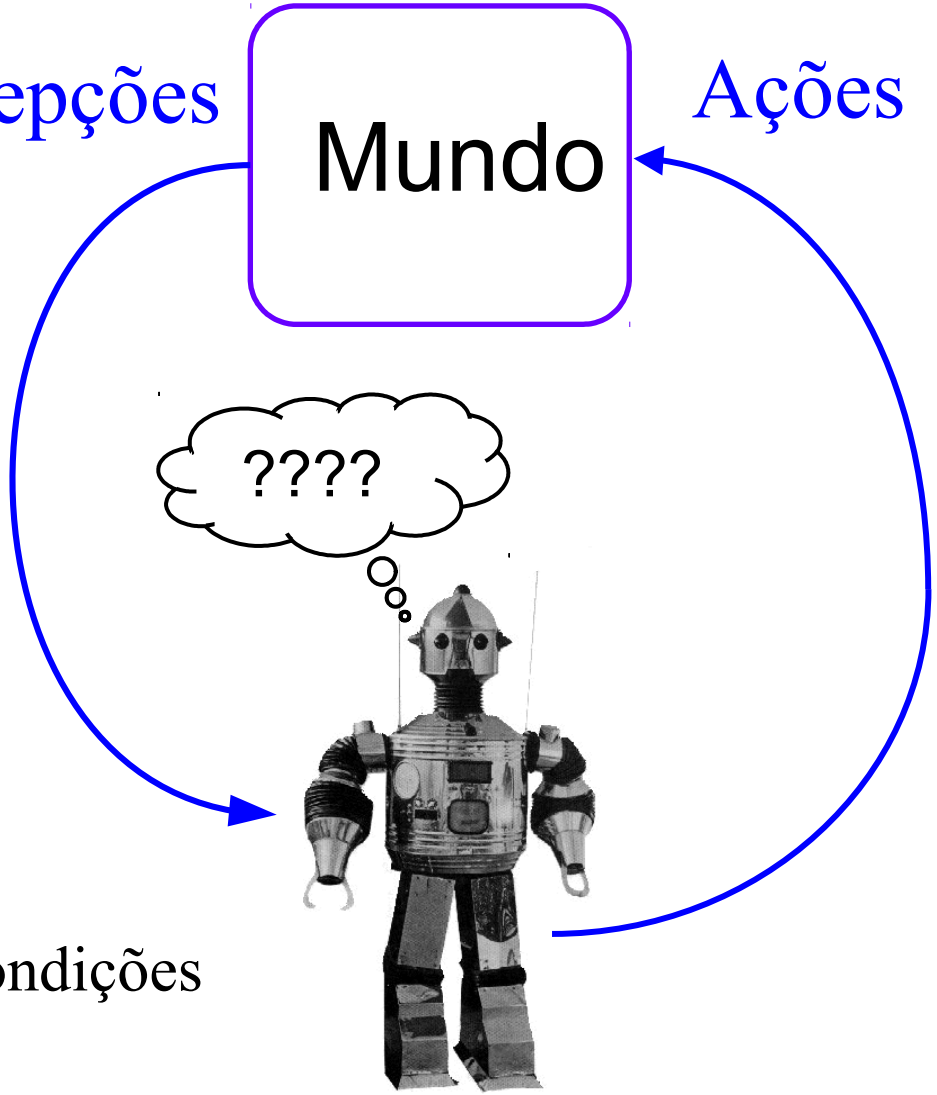


Segurança/Controle de Redes

Elementos comuns

- Nós temos um sistema controlável que muda de estado com o tempo (de alguma forma previsível)
  - O estado descreve informações essenciais sobre o sistema (as informações visíveis sobre as cartas no jogo de paciência…)
- Nós temos um objetivo que especifica quais estados, ou sequências de estados, são mais/menos desejados
- Pode-se (parcialmente) controlar a transição dos estados do sistema através da tomada de ações
- **Problema:** A cada momento deve-se selecionar uma ação que otimiza o objetivo geral
  - Produzir as sequências de estados mais desejáveis

Premissas do Planejamento Clássico



perfeito

totalmente observável

única fonte de mudança

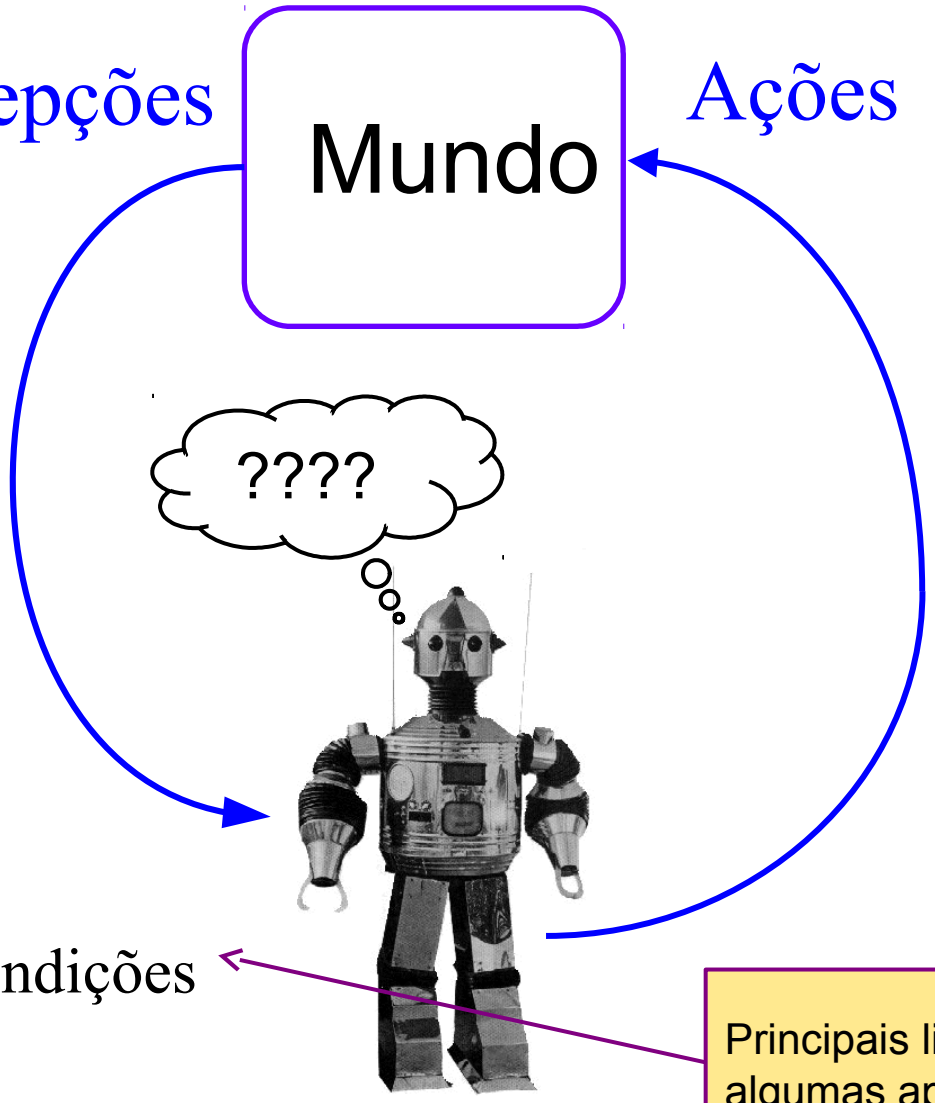
determinístico

instantâneo

Objetivo

alcançar as condições do objetivo

Premissas do Planejamento Clássico



perfeito

totalmente observável

única fonte de mudança

determinístico

instantâneo

Objetivo

alcançar as condições do objetivo

Principais limitantes para algumas aplicações práticas

Por que planejamento clássico é relevante?

- Ênfase em analisar a estrutura combinatória de problemas
  - Desenvolveu muitas ideias poderosas nesta direção
  - Pesquisas “modernas” em planejamento, como Markov Decision Processes (MDP), praticamente ignoraram este tipo de análise (embora também enfrentem este tipo de problema)
- Planejadores clássicos costumam ser capazes de escalar melhor para espaços de estados grandes graças à utilização dessas ideias.
- **Replanejamento:** muitos ambiente estáveis satisfazem aproximadamente essas premissas (e.g. robô para mover caixas)
  - É possível tratar violações pequenas das premissas através de replanejamento e monitoramento da execução
  - O mundo frequentemente não é tão aleatório e pode ser pensado de forma efetivamente de forma determinística

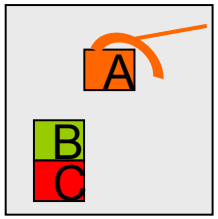
Por que planejamento clássico é relevante?

- Ideias oriundas de técnicas de planejamento clássico frequentemente são a base para desenvolver técnicas de planejamento não-clássicas
  - Uso de planejadores clássicos como um componente de planejamento probabilístico [Yoon et. al. 2008] (i.e. redução de planejamento probabilístico para planejamento clássico)
  - Técnicas poderosas de análise de domínio do planejamento clássico foram integradas em planejadores MDP

Representando Estados

Estados do mundo são representados como conjuntos de fatos

Fatos também serão chamados de proposições.

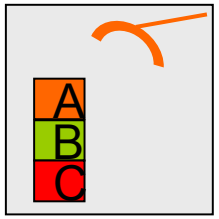


holding(A)
clear(B)
on(B,C)
onTable(C)

State 1

handEmpty
clear(A)
on(A,B)
on(B,C)
onTable(C)

State 2



Closed World Assumption (CWA):

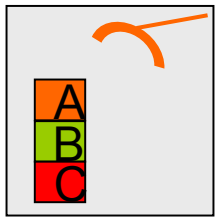
Fatos não listados em um estado são considerados falsos. Sob CWA estamos assumindo que o mundo é totalmente observável.

Representando Estados

Objetivos também são representados como conjuntos de fatos.

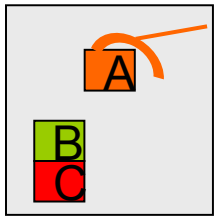
Por exemplo, { on(A,B) }, é um objetivo no mundo dos blocos.

Um **estado objetivo** é qualquer estado que contenha todos os estados objetivo



handEmpty
clear(A)
on(A,B)
on(B,C)
onTable(C)

Estado 1

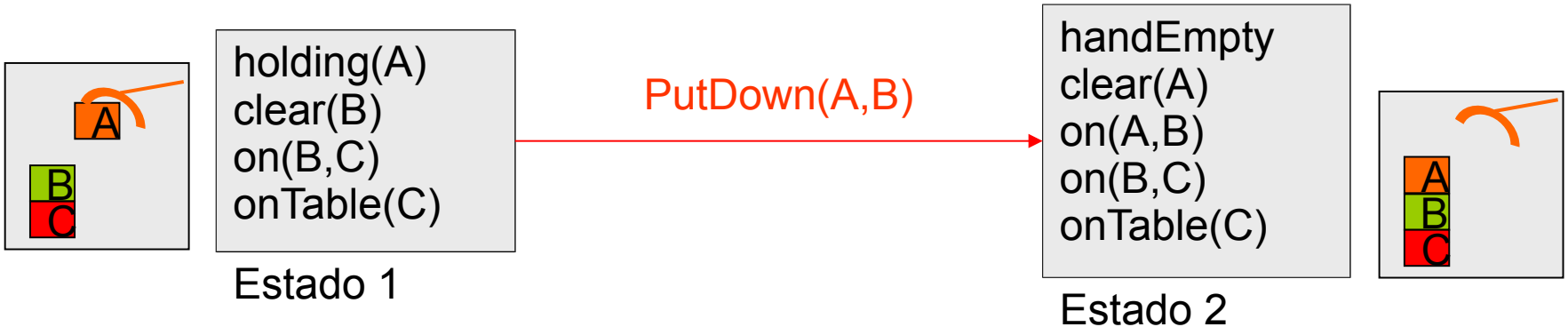


holding(A)
clear(B)
on(B,C)
onTable(C)

Estado 2

Estado 1 é um estado objetivo para o objetivo { on(A,B) }.
Estado 2 não é um estado objetivo para { on(A,B) }.

Representando uma Ação em STRIPS



Uma definição STRIPS de ação especifica:

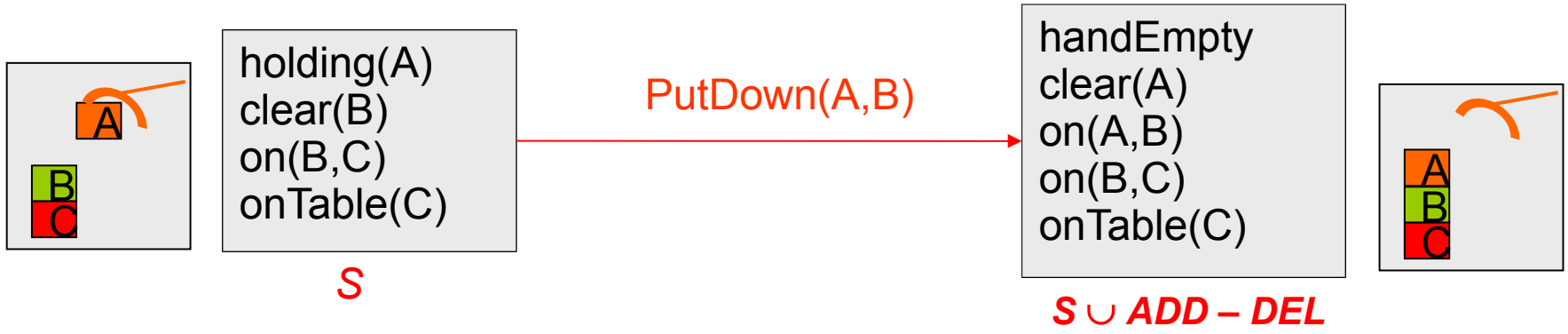
1) um conjunto PRE de fatos que são pré-condições
2) um conjunto ADD de fatos-efeito que são adicionados
3) um conjunto DEL de fatos-efeito que são apagados

PutDown(A,B):

PRE: { holding(A), clear(B) }
ADD: { on(A,B), handEmpty, clear(A) }
DEL: { holding(A), clear(B) }

STRIPS = Stanford Research Institute Problem Solver

Semântica de Ações STRIPS



- Uma ação STRIPS é **aplicável** (ou permitida) em um estado quando
  suas pré-condições estão contidas no estado.
- Tomar uma ação em um estado ***S*** resulta em um novo estado:
  - ***S ∪ ADD – DEL***
    (i.e. adicionar os efeitos ADD e remover os efeitos DEL)

PutDown(A,B):

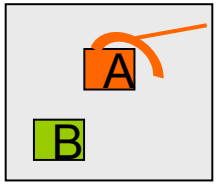
PRE: { holding(A), clear(B) }
ADD: { on(A,B), handEmpty, clear(A)}
DEL: { holding(A), clear(B) }

Problema de Planejamento STRIPS

Um **problema de planejamento STRIPS** especifica:
1) um estado inicial *S*
2) um objetivo *G*
3) um conjunto de ações STRIPS

Objetivo: encontrar uma sequência de ações que chegue a um estado objetivo, ou retornar que o objetivo é inalcançável.

Problema-Exemplo :



holding(A)
clear(B)
onTable(B)

Estado Inicial

Solução: (PutDown(A,B))

on(A,B)

Objetivo

PutDown(A,B):

PRE: { holding(A), clear(B) }
ADD: { on(A,B), handEmpty, clear(A)}
DEL: { holding(A), clear(B) }

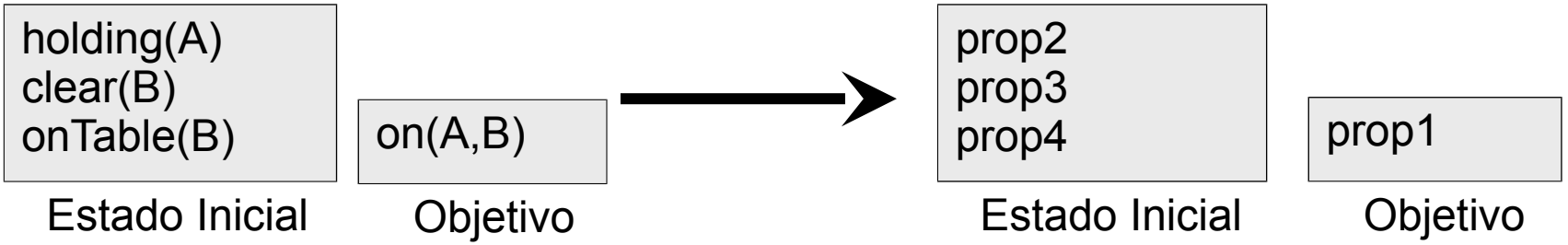
PutDown(B,A):

PRE: { holding(B), clear(A) }
ADD: { on(B,A), handEmpty, clear(B) }
DEL: { holding(B), clear(A) }

Ações STRIPS

Planejadores Proposicionais

- Por razões de clareza escrevemos proposições tais como on(A,B) em função de objetos (e.g. A e B) e predicados (e.g. on).
- No entanto, os planejadores que iremos considerar ignoram a estrutura interna de proposições tais como on(A,B).
- Tais planejadores são chamados **planejadores proposicionais** em oposição a planejadores de primeira ordem, ou relacionais
- Portanto não fará diferença para o planejador se nós substituíssemos toda ocorrência de “on(A,B)” em um problema por “prop1” (e assim por diante para outras proposições)
- Parece errado ignorar a existência de objetos. Mas atualmente planejadores proposicionais são o estado-da-arte.



Esquemas de ações STRIPS

Por conveniência nós tipicamente especificamos problemas via esquemas de ações ao invés de escrever ações STRIPS individuais.

Esquema de Ações: (x e y são variáveis)

PutDown(x,y):

PRE: { holding(x), clear(y) }
ADD: { on(x,y), handEmpty, clear(x) }
DEL: { holding(x), clear(y) }

PutDown(B,A):

PRE: { holding(B), clear(A) }
ADD: { on(B,A), handEmpty, clear(B) }
DEL: { holding(B), clear(A) }

. . . .

PutDown(A,B):

PRE: { holding(A), clear(B) }
ADD: { on(A,B), handEmpty, clear(A) }
DEL: { holding(A), clear(B) }

- Cada forma de substituir variáveis por objetos a partir do estado inicial e objetivo gera uma ação STRIPS "ground" (instanciada – sem variáveis livres).
- Dado um conjunto de esquemas, um estado inicial, e um objetivo, planejadores proposicionais compilam esquemas em ações ground e então ignoram a existência de objetos deste ponto em diante.

STRIPS Versus PDDL

- O AIMA faz referência à PDDL para definição de problemas de planejamento (≠ STRIPS)
- A Planning Domain Description Language (PDDL) foi definida por pesquisadores em planejamento como uma linguagem padrão para definir problemas de planejamento
  - Inclui STRIPS como caso especial juntamente com características mais avançadas
  - Algumas características adicionais simples incluem: especificação de tipo para objetos, pré-condições negadas, efeitos add/del condicionais
  - Características avançadas incluem variáveis numéricas e ações duráveis
- A maioria dos planejadores que você pode baixar recebe PDDL como entrada (veja http://fai.cs.uni-saarland.de/hoffmann/ff.html)
  - A maioria suporta apenas as características mais simples de PDDL (≅ STRIPS)
  - A sintaxe PDDL é fácil de aprender a partir dos exemplos que acompanham os planejadores, mas uma definição formal pode ser encontrada:

http://www.informatik.uni-ulm.de/ki/Edu/Vorlesungen/GdKI/WS0203/pddl.pdf

Propriedades de Planejadores

- Um planejador é correto (**sound**) se toda sequência de ações que ele retorna é de fato uma solução verdadeira
- Um planejador é completo (**complete**) se ele retorna uma sequência de ações ou “não existe solução” para qualquer problema de entrada
- Um planejador é ótimo (**optimal**) se ele sempre retorna a solução menor possível

O ótimo é um requisito importante?

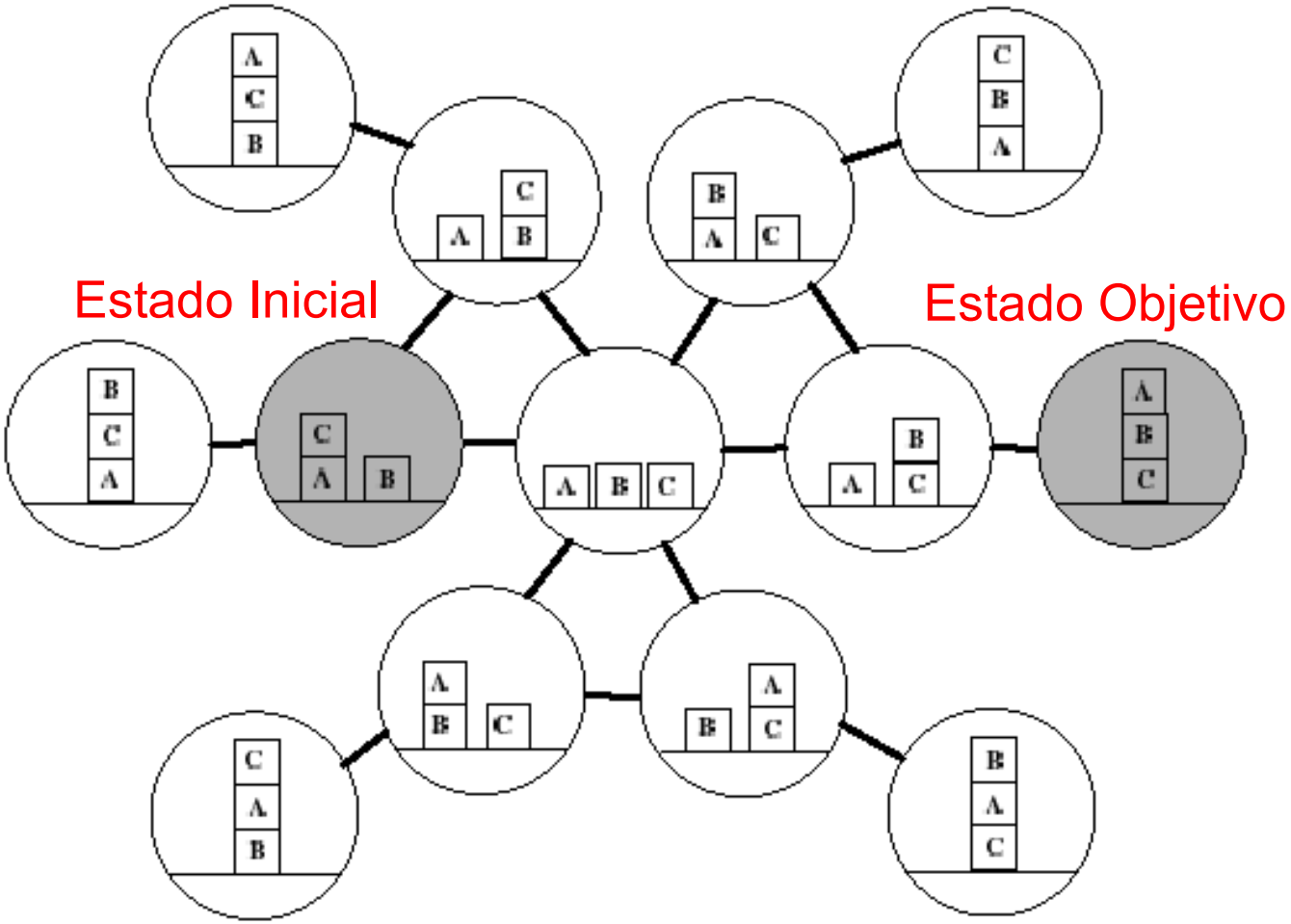
É um requisito razoável?

Planejamento como Busca em Grafos

- É fácil ver planejamento como um problema de busca em grafos
- Nós/vértices = estados possíveis
- Arestas direcionadas = ações STRIPS
- Solução: caminho de um estado inicial (i.e. vértice) para um estado/vértice que satisfaz o objetivo

Espaço de Busca: Mundo dos Blocos

Grafo é finito



Planejamento como Busca em Grafos

- Planejamento é simplesmente encontrar um caminho em um grafo
  - Por que não simplesmente usar algoritmos padrão de grafos para encontrar caminhos?
- **Resposta:** grafos são exponencialmente grandes no tamanho de codificação dos problemas (i.e. tamanho dos problemas STRIPS).
  - Mas, algoritmos padrão são polinomiais no tamanho do grafo
  - Então algoritmos padrão necessitariam de tempo exponencial
- É possível fazer melhor do que isso?

Complexidade de Planejamento STRIPS

PlanSAT

Entrada: um problema de planejamento STRIPS

Saída: "sim" se o problema tem solução, caso contrário "não"

- PlanSAT é decidível.
  - Por que? (o número de estados é finito)
- Em geral PlanSAT é PSPACE-completo!
  Simplesmente encontrar um plano é difícil no pior caso.
  - Mesmo com ações limitadas para 2 pré-condições e 2 efeitos

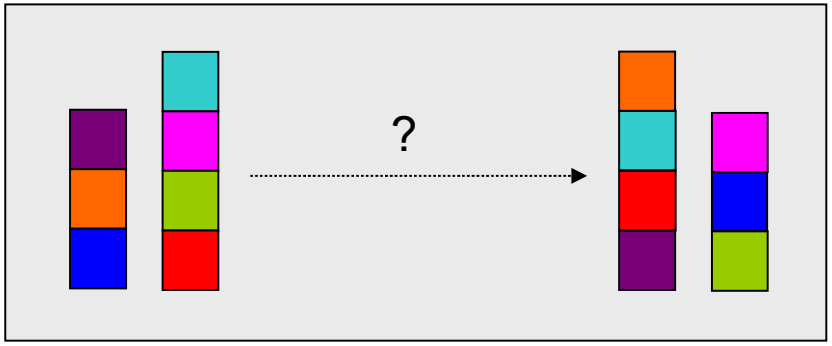
Isso significa que deve-se desistir de planejamento em IA?

NOTA: PSPACE é o conjunto de todos os problemas que são decidíveis em espaço polinomial.

Acredita-se que PSPACE-completo contém estritamente NP.

Satisficing vs. Optimality

- Enquanto simplesmente encontrar um plano é difícil no pior caso, para muitos domínios de planejamento, encontrar um plano é fácil.
- No entanto, encontrar soluções ótimas pode ainda ser difícil nesses domínios.
  - Por exemplo, planejamento ótimo no mundo dos blocos é NP-completo.
- Na prática, é frequentemente suficiente encontrar “boas” soluções “rapidamente”, mesmo que elas não sejam ótimas.
  - Isso é frequentemente chamado de um objetivo do tipo “satisficing”.
  - Por exemplo, no mundo dos blocos é possível produzir soluções aproximadamente ótimas em tempo linear. Como?



Satisficing

- Mesmo assim, encontrar planos satisficing para problemas STRIPS arbitrários não é fácil.
  - Ainda é preciso lidar com o tamanho exponencial do espaço de estados
  - Por que é possível fazer melhor do que algoritmos genéricos de grafos?
- **Resposta:** nós temos uma descrição STRIPS compacta e estruturada dos problemas
  - Tentar tomar proveito da estrutura presente nessas descrições para procurar por soluções de forma inteligente
  - Isso que faz um dos algoritmos de bastante sucesso em planejamento, GraphPlan.