Busca sob Incerteza/Não-determinismo

Disciplina if684 2017.1

Busca até então...

- Ações com resultados determinísticos
- Ambientes observáveis
- Agora vamos considerar situações em que isso não é o caso

Importância das percepções

- Ambiente não determinístico
  - Percepções informam ao agente quais dos resultados possíveis de suas ações de fato ocorreram
  - Percepções futuras não podem ser determinadas com antecedência, no entanto ações futuras dependem dessas percepções
    - Solução para um problema não é uma sequência de ações, mas um **plano de contigência** (estratégia).
      - Especifica o que fazer dependendo das percepções recebidas

Mundo do aspirador de pó

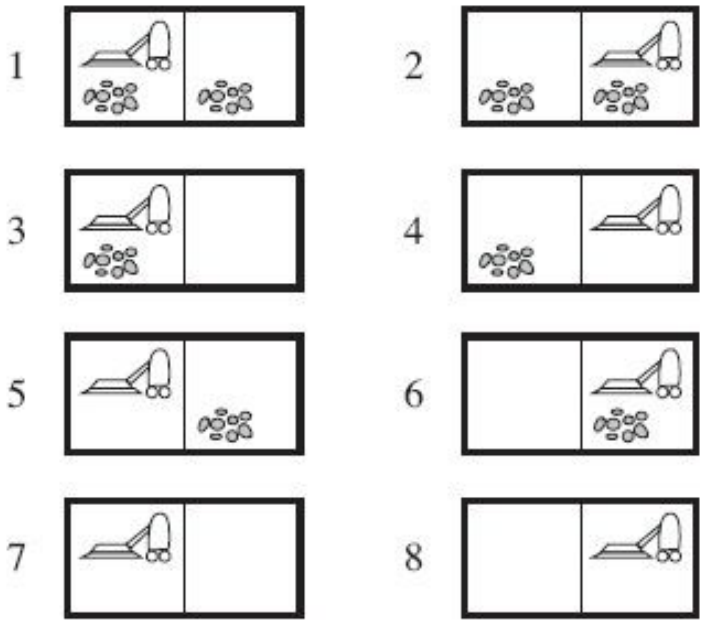


Figura 4.9 Os oito estados possíveis do mundo

Três ações: Esquerda (E), Direita (D), Aspirar (A)
Objetivo: 7 ou 8

Mundo do aspirador de pó

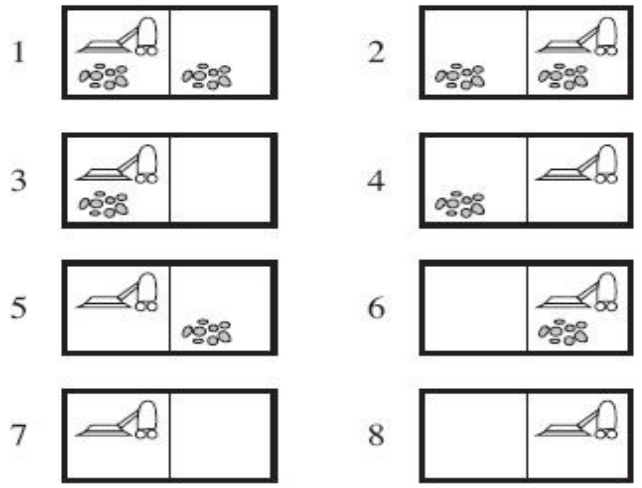


Figura 4.9 Os oito estados possíveis do mundo

Três ações: Esquerda (E), Direita (D), Aspirar (A)

Objetivo: 7 ou 8

- Ambiente observável, determinista, completamente conhecido
- Trivialmente solucionável por qualquer algoritmo de busca
- A solução é uma sequência de ações

Não determinismo

- Mundo do aspirador de pó defeituoso. Comportamento da ação “aspirar”:

  - Quando aplicada a um quadrado sujo, a ação limpa o quadrado e, por vezes, limpa a sujeira do quadrado adjacente, também.
  - Quando aplicado a um quadrado limpo, a ação por vezes deposita sujeira no carpete.[9]

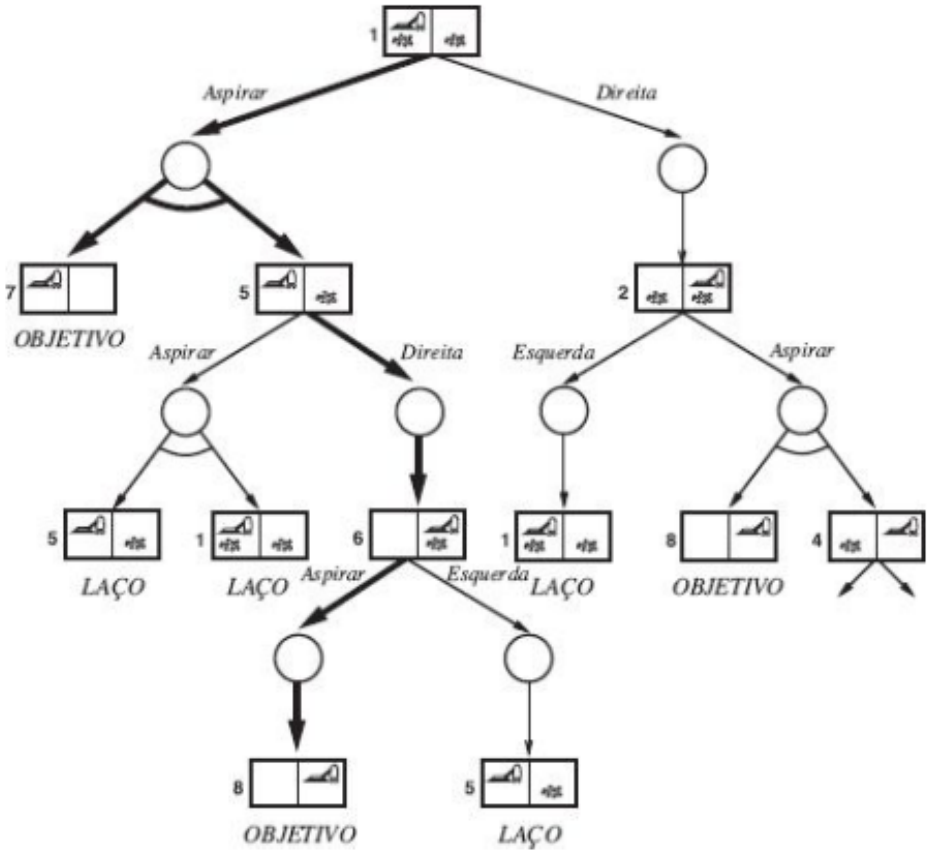
Modelo de transição

- Nesse caso, o RESULTADO de uma ação é um conjunto de estados possíveis.
  - função de transição
    - (Estado, Ação) → Conjunto de estados possíveis
      - Antes era, (Estado, Ação) → Estado

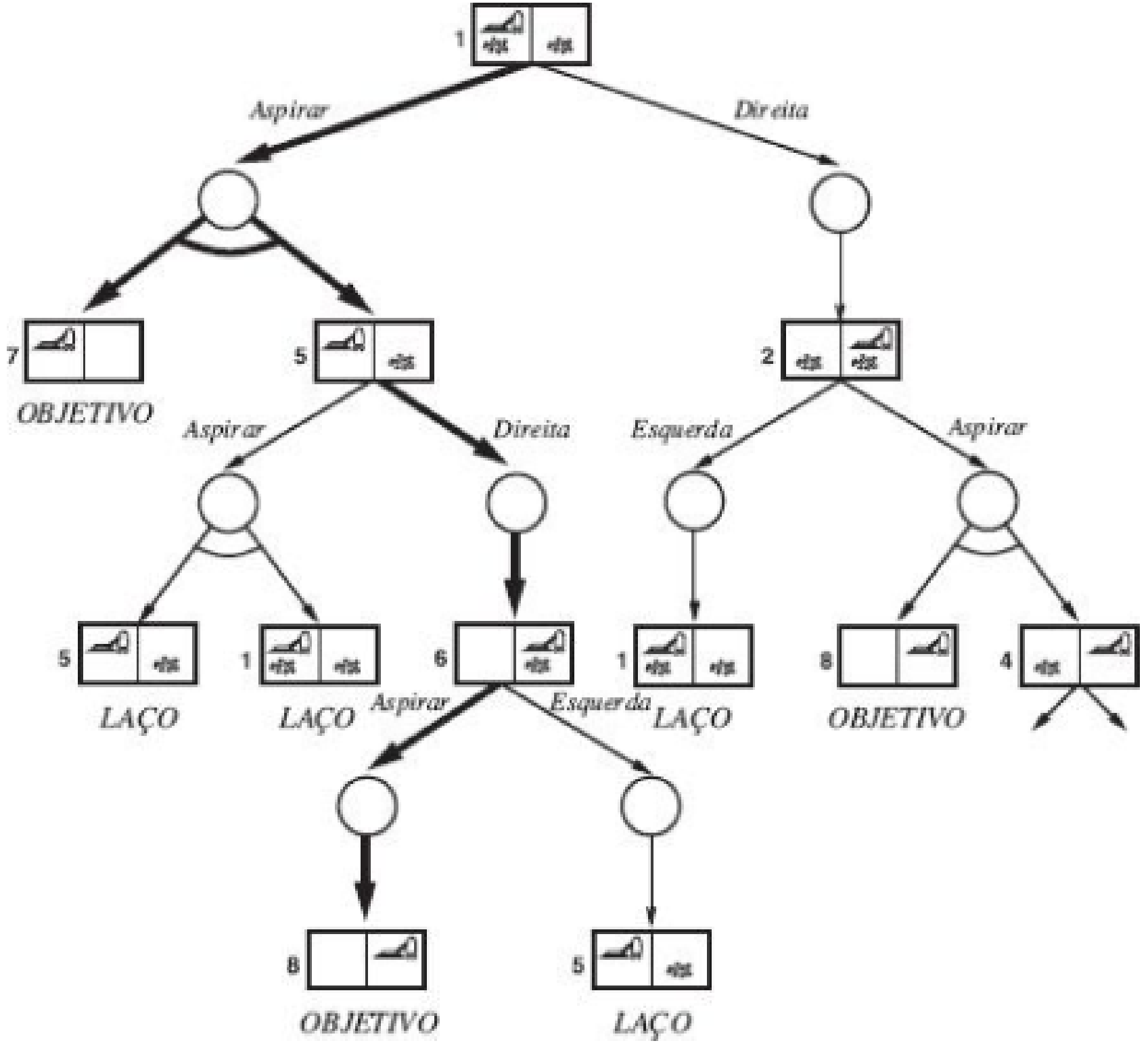
Solução para o problema

- Não existe uma única sequência de ações que resolva o problema.
- Precisaremos de um **plano de contigência**, tal como:
  - [Aspirar, se Estado = 5, então [Direita, Aspirar] senão [ ]].
- Soluções para problemas não determinísticos podem conter comandos **se-então-senão** aninhados.
  - Ou seja, são **árvores**, e não sequências.
  - É preciso recorrer às percepções para saber “onde está” (qual o estado atual).

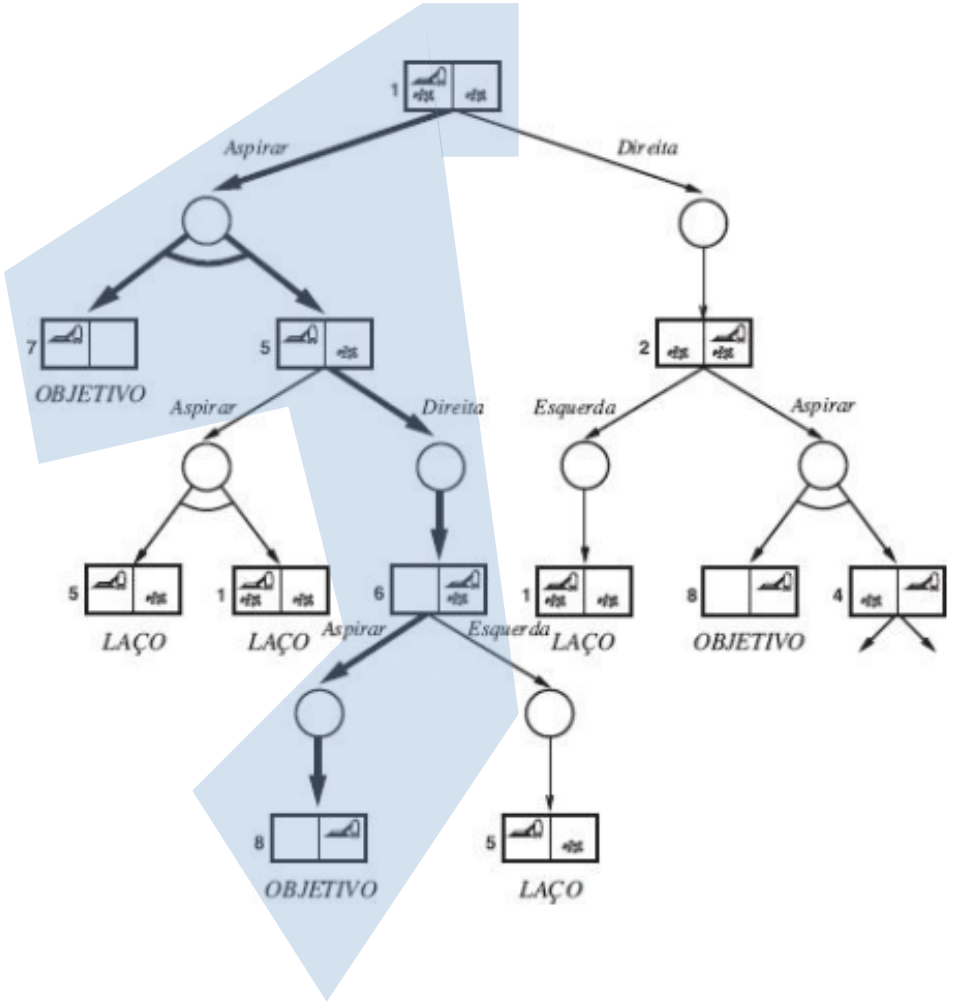
Árvores E-OU



- Nós OU (quadrado): escolha do agente
- Nós E (círculo): resultado do ambiente para cada ação
- Nós OU e E se alternam numa árvore E-OU

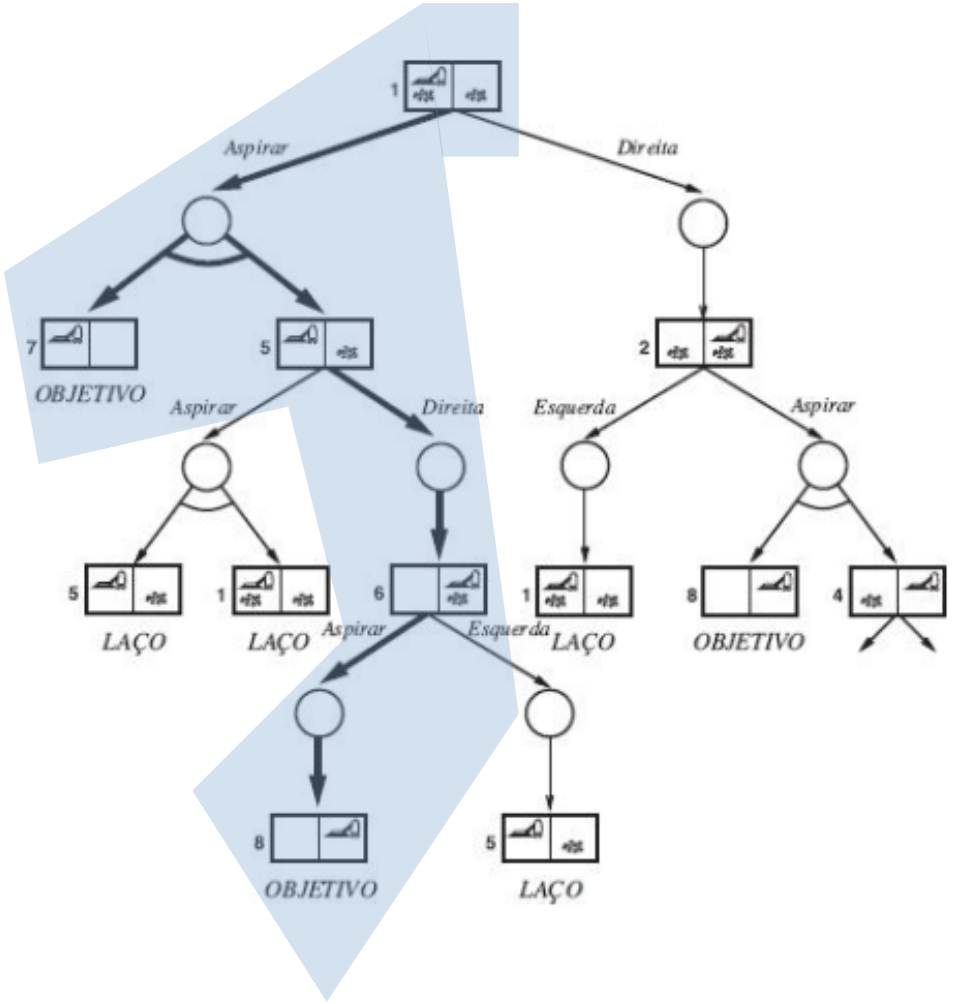

1
Aspirar
Direita
7
OBJETIVO
5
2
Aspirar
Direita
Esquerda
Aspirar
5
1
6
1
8
4
LAÇO
LAÇO
Aspirar
Esquerda
LAÇO
OBJETIVO
8
5
OBJETIVO
LAÇO

Árvores E-OU



- Solução: encontrar um plano de contigência.
- Subárvore que:
  - Tem um nó objetivo em cada folha
  - Especifique uma ação em cada um de seus nós OU
  - Inclua todos os nós resultantes em cada um de seus nós E.

Árvores E-OU



- Plano:
  - [Aspirar, se Estado=5, então [Direita, Aspirar] senão []]
- Note que a percepção durante a execução agora é importante

Algoritmo recursivo para Busca em Grafos E-OU

- Note que é necessário tratar ciclos, que podem surgir frequentemente.
  - Ex.: uma ação por vezes não muda o estado (sem efeito)
- O algoritmo a seguir trata estados repetidos que já ocorreram no caminho da raiz.
  - Se o estado atual for idêntico a um estado no caminho da raiz, retorna FALHA.
  - Se existir uma solução não cíclica ela deve ser acessível a partir da recursão anterior do estado atual, por isso a nova recursão poderá ser descartada

Algoritmo

função BUSCA-EM-GRAFOS-E-OU(*problema*) **retorna** *um plano condicional ou falha*
    BUSCA-OU(*problema*.Estado-Inicial, *problema*, [ ])

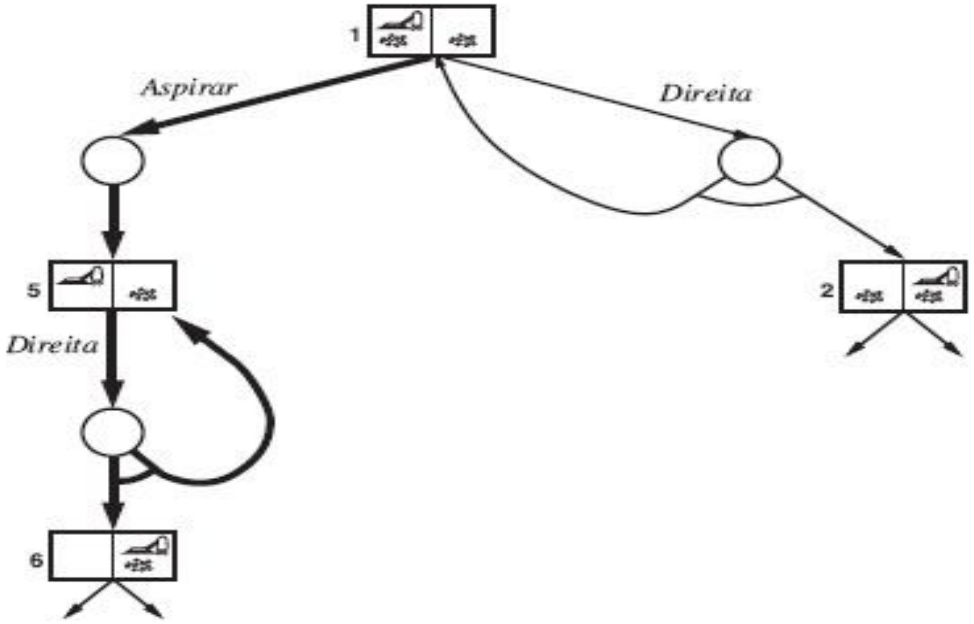
função BUSCA-OU(*estado, problema, caminho*) **retorna** *um plano condicional ou falha*
    se *problema*.TESTE-OBJETIVO(*estado*) **então retorna** o plano vazio
    se *estado* estiver no *caminho* **então retorna** *falha*
    para cada *ação* **em** problema.AÇÕES(*estado*) **faça**
        plano <- BUSCA-E(RESULTADO(*estado*, *ação*), *problema*, [*estado* | *caminho*])
        se *plano* <> *falha* **então retorna** [*AÇÃO* | *plano*]
    retorna *falha*

função BUSCA-E(*estados, problema, caminho*) **retorna** *um plano condicional ou falha*
    para cada s_i **em** *estados* **faça**
        $plano_i$ <- BUSCA-OU(s_i, *problema, caminho*)
        se $plano_i$ = *falha* **então retorna** *falha*
    retorna [**se** s_1 **então** $plano_1$ **senão se** s_2 **então** $plano_2$ **senão** ... **se** s_{n-1} **então** $plano_{n-1}$ **senão** $plano_n$

Aspirador em pó com incerteza

- Como o mundo comum (sem defeitos)
- MAS, as ações de movimento por vezes falham, deixando o agente no mesmo local.
  - Ex. Mover para a Direita no estado 1, leva para o conjunto de estados {1, 2}

Aspirador em pó com incerteza



- Não há nenhuma solução acíclica a partir do estado 1.
- Há solução cíclica: continuar tentando para direita até que funcione.
- Todas as soluções para esse problema são planos cíclicos pois não há nenhuma maneira de se mover de forma confiável.

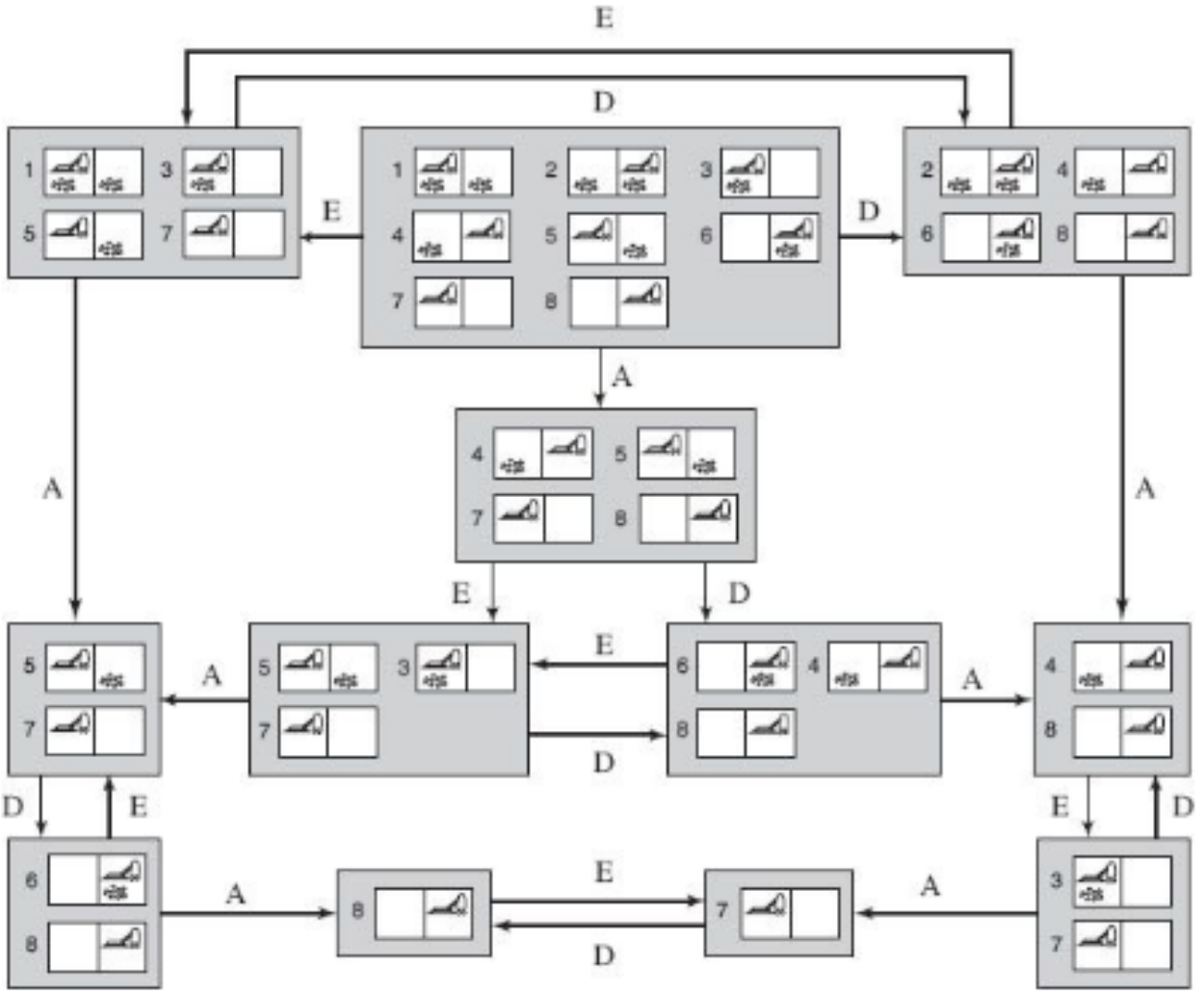
Planos cíclicos

- Em geral, um plano cíclico pode ser considerado uma solução, desde que:
  - cada folha seja um estado objetivo e;
  - uma folha seja acessível de cada ponto no plano.
- Dada a definição de uma solução cíclica, um agente que executa tal solução eventualmente alcançará o objetivo, desde que:
  - cada resultado de uma ação não determinística ocorra eventualmente.
- É razoável?
  - Depende do problema

Mundo parcialmente observável

- Imaginemos agora o mundo determinístico do aspirador de pó, mas o aspirador não tem sensores para saber onde está nem se está sujo
- Estados de crença: onde o agente “pode estar”

Aspirador de pó determinístico inobservável



Aspirador de pó determinístico inobservável