Agentes Inteligentes

exemplos

Inteligência Artificial

PCS3438

Escola Politécnica da USP
Engenharia de Computação (PCS)

Estrutura do Agente

Agente = arquitetura de HW

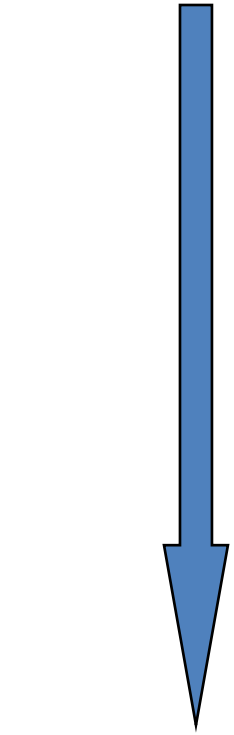
+

arquitetura de SW

- Arquitetura de HW:
  - onde o agente vai ser implementado (dispositivo computacional, sensores e atuadores)
- Arquitetura de SW:
  - "arquitetura do agente": módulos básicos do programa e suas inter-relações

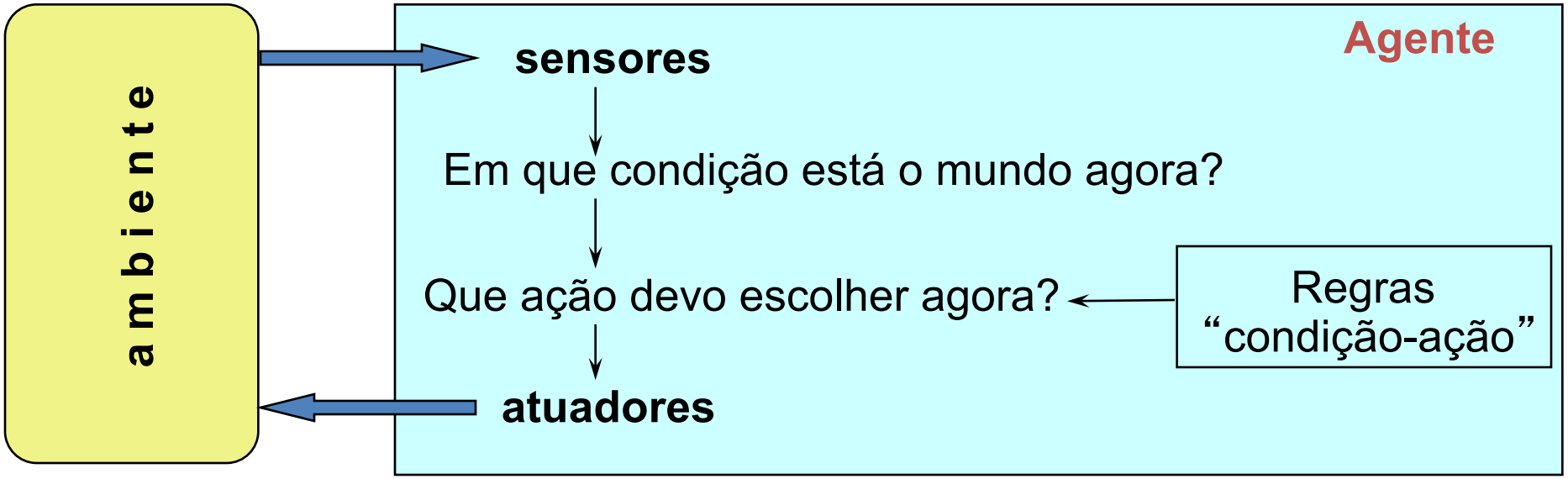
Arquiteturas

- Agente tabela
- **Agente reativo**
- Agente baseado em modelo
- Agente baseado em objetivos
- Agente baseado em utilidade
- Agente aprendiz



autonomia
complexidade

Agente reativo



- Vantagens e desvantagens
  - Regras condição-ação: representação inteligível, modular e eficiente
    - ex. **Se** velocidade > 60 **então** multar
  - Não pode armazenar uma sequência perceptiva, pouca autonomia
- Ambientes:
  - Reflexo imprescindível em ambientes dinâmicos
  - Observável, episódico, pequeno

function Agente-Reflexo-Simples (*percept*) return uma ação

static: *regras* – um conjunto de regras condição-ação

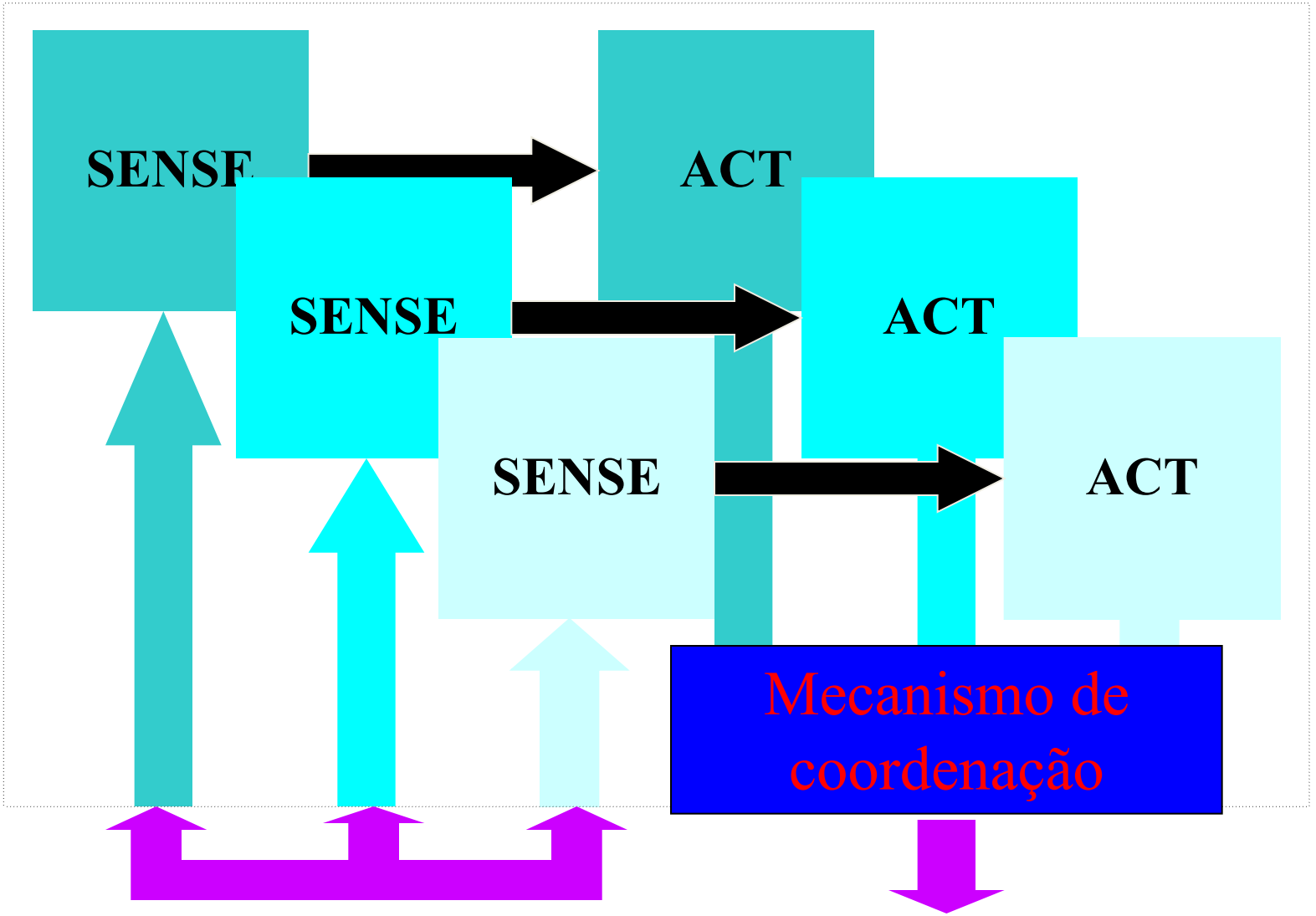
estado ← Interpreta-Entrada(*percept*)
regra ← Acha-Regra(estado, *regras*)
ação ← Regra-Ação [regra]
return ação

Uso limitado: o ambiente tem que ser totalmente observável, pois o agente só funciona apropriadamente se a regra correta for disparada, o que depende da percepção atual realizada.

Arquiteturas Reativas para Robôs

- Surgidas no final dos anos 80.
- Fundamentadas em estudos do comportamento animal (Etologia) → baseadas em comportamentos.
- Baseadas em processamento paralelo (vários comportamentos simultaneamente ativos).

Arquiteturas Reativas



Mecanismo de coordenação

- **Coordenação Competitiva**: a ação resultante num dado instante é selecionada a partir de uma competição entre os comportamentos ativos (um vence).
- **Coordenação Cooperativa**: a função de coordenação produz uma ação resultante para a qual contribuem todos os comportamentos ativos.

Estudo de Caso: REACT
Comportamentos Reativos para Robôs Móveis

HW: plataforma Pioneer 2DX

LTI
Laboratório de
Técnicas Inteligentes

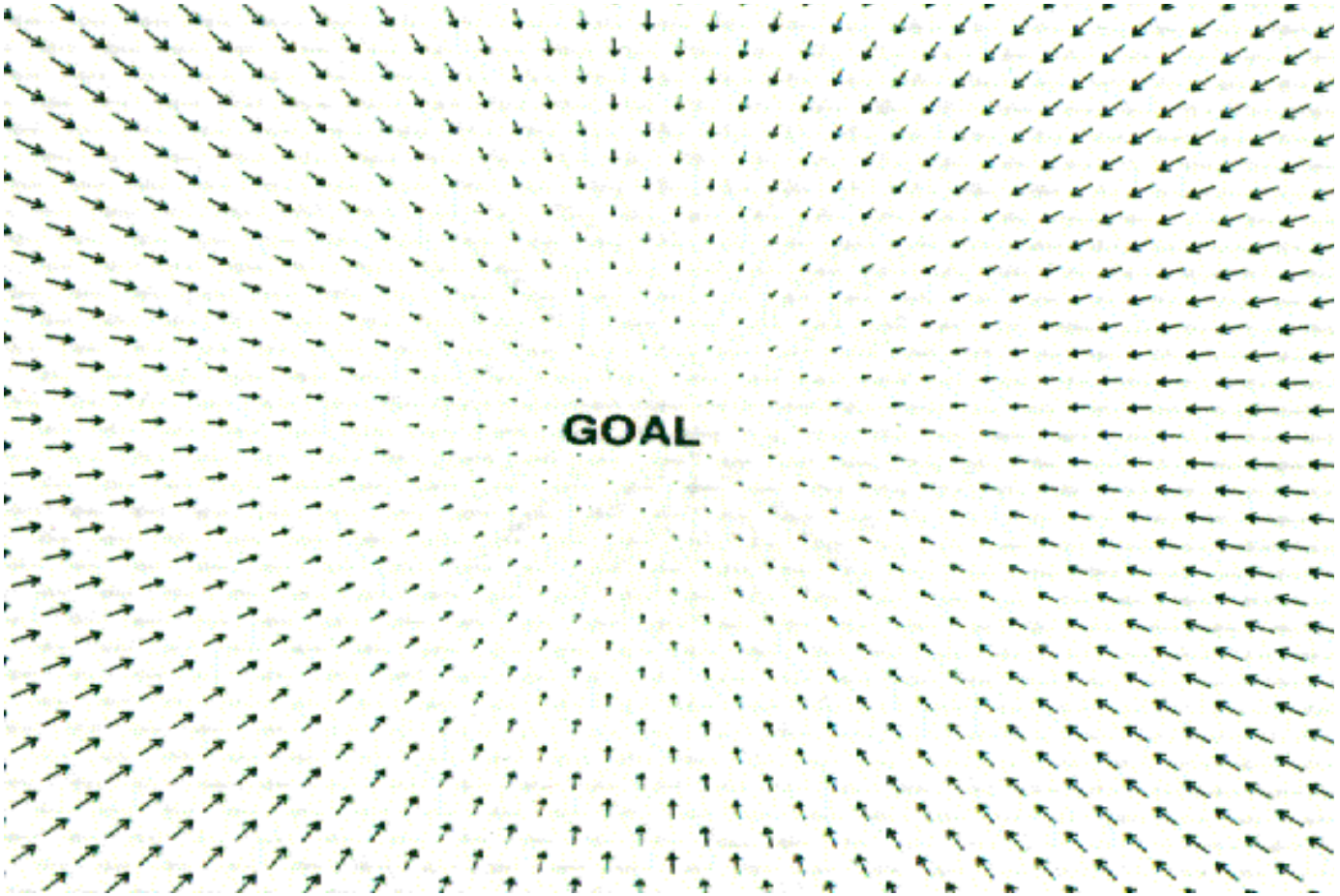
REACT

- Arquitetura baseada em *Motor Schemas*
- *Motor Schema* ≡ comportamento
  - Comportamentos são divididos em:
    - módulo de percepção
    - módulo de codificação
- Saída do *Motor Schema* → *Vetor*
  - representa a ação a ser executada
    - Magnitude ≡ velocidade
    - direção ≡ rotação

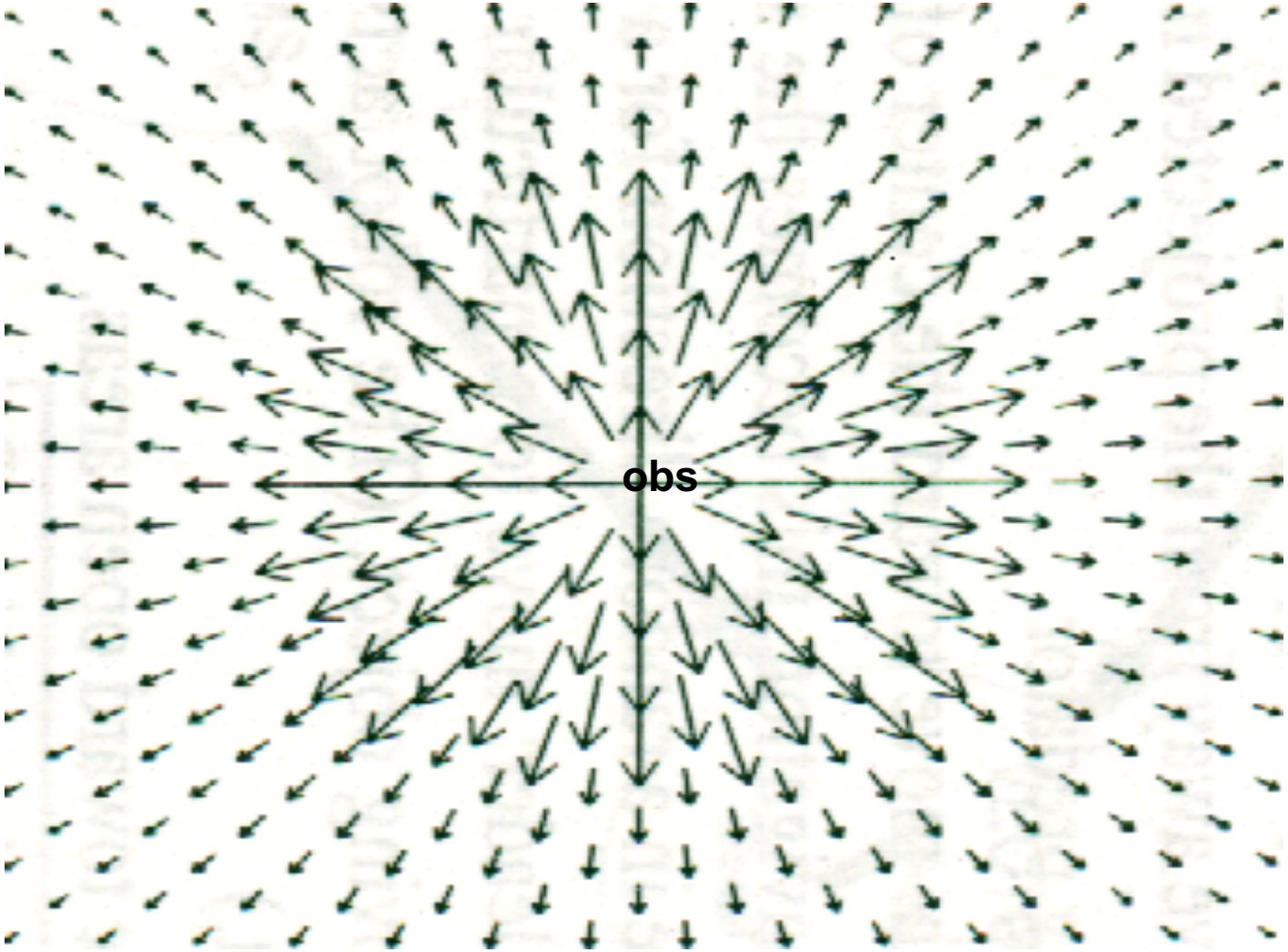
REACT

- Módulo de codificação:
  - Mapeamento contínuo: percepções ➔ ações
  - Usa o Método de Campos Potenciais
- Coordenação dos comportamentos:
  - Abordagem cooperativa ➔ Superposição dos campos de força
- Alguns comportamentos na REACT:
  - avoidCollision
  - moveToGoal

REACT – moveToGoal

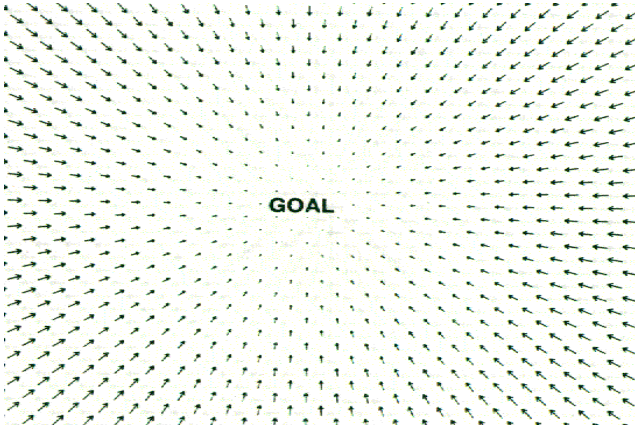


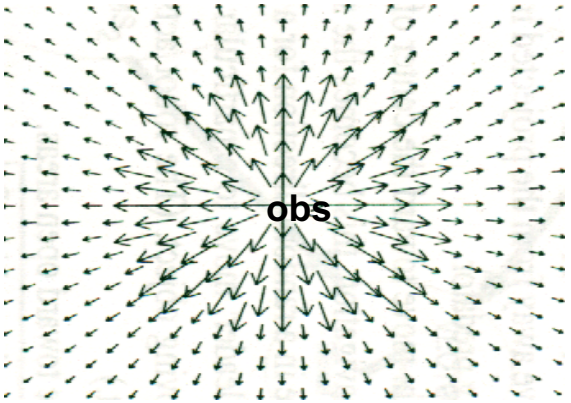
REACT - avoidCollision



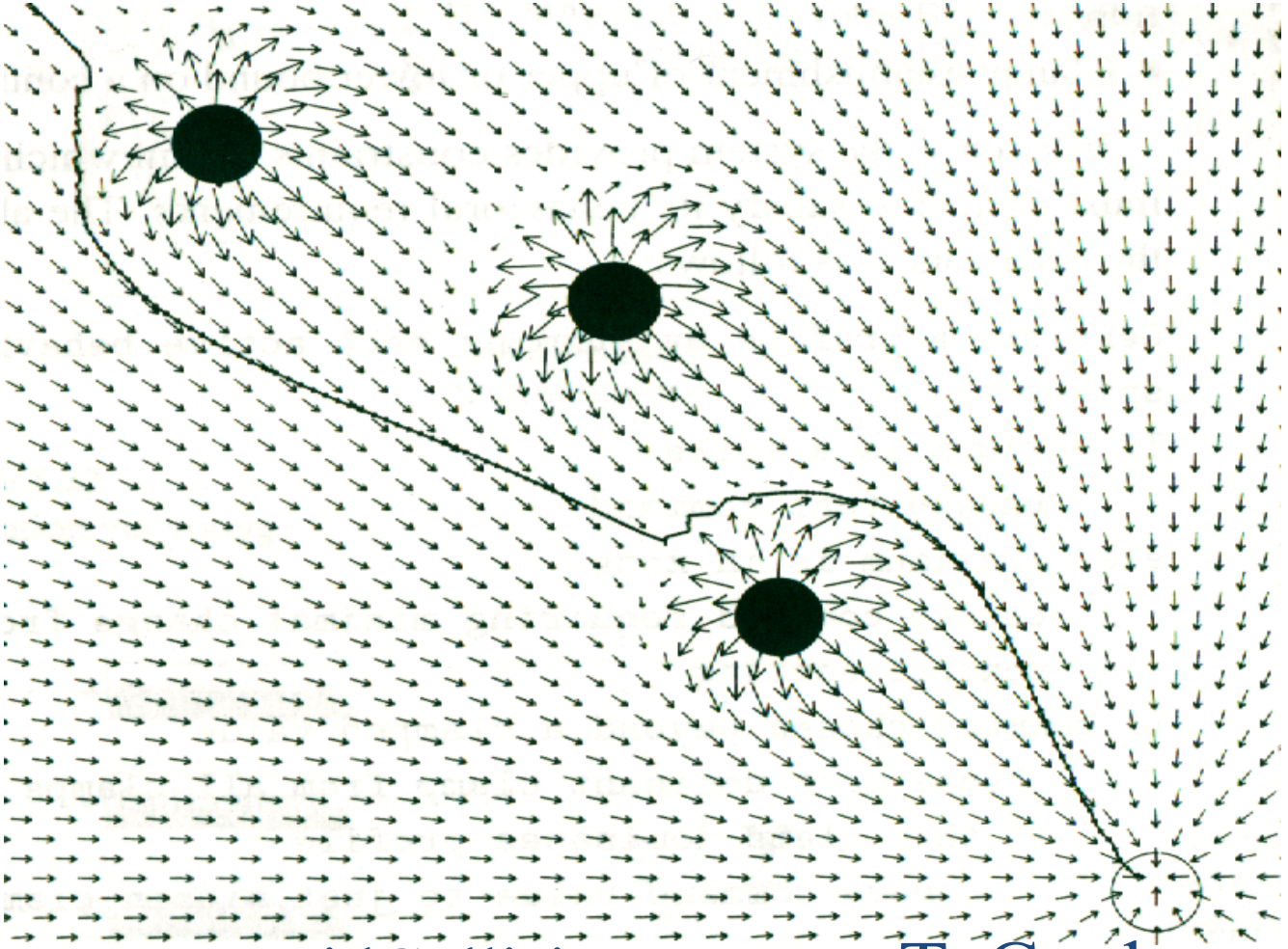
REACT

moveToGoal



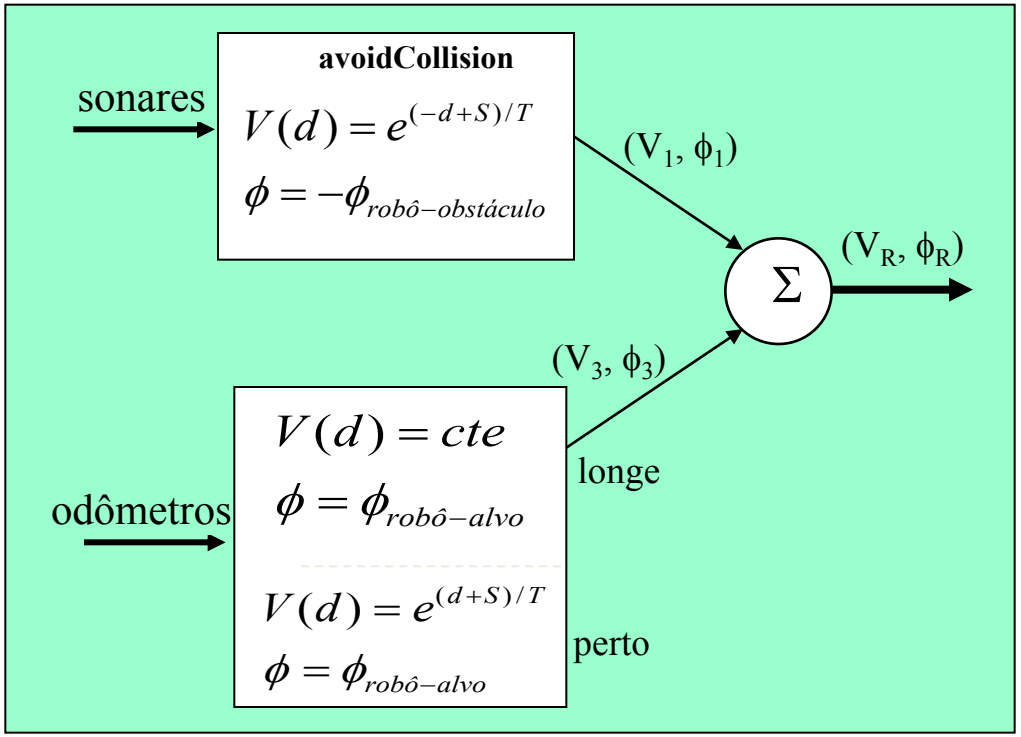


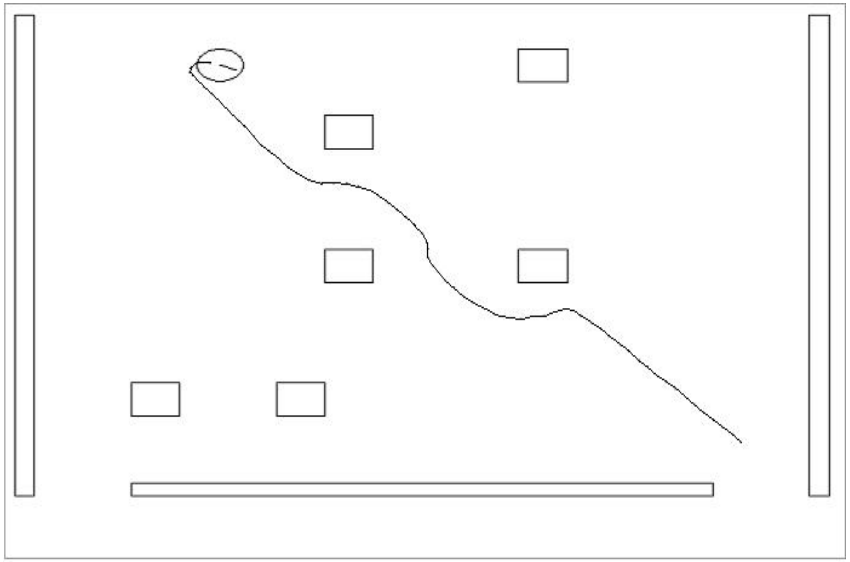
avoidCollision

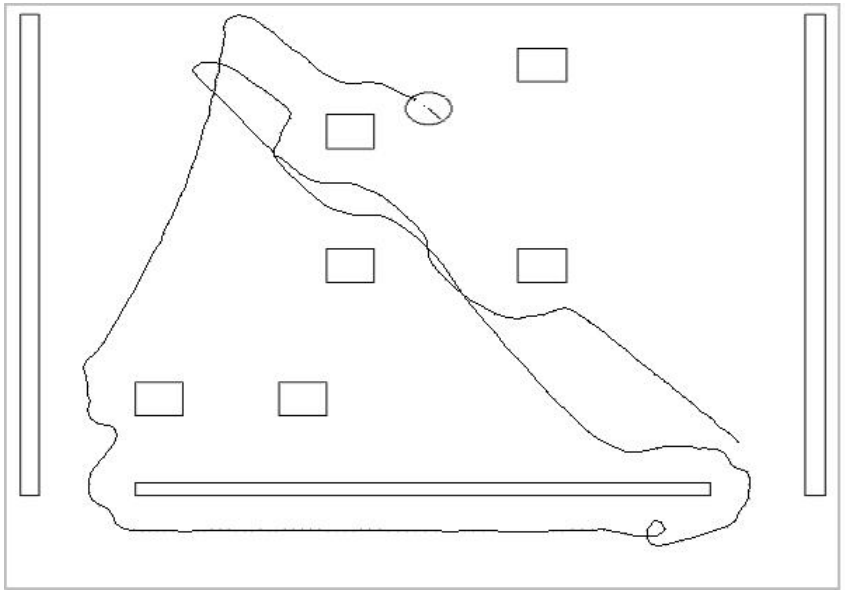


avoidCollision + moveToGoal

REACT


avoidCollision
sonares
$V(d) = e^{(-d+S)/T}$
$\phi = -\phi_{robô-obstáculo}$
(V_1, ϕ_1)
(V_R, ϕ_R)
Σ
(V_3, ϕ_3)
odômetros
$V(d) = cte$
$\phi = \phi_{robô-alvo}$
longe
$V(d) = e^{(d+S)/T}$
$\phi = \phi_{robô-alvo}$
perto





Arquiteturas Reativas: Vantagens

- Comportamentos reativos são normalmente simples de projetar.
- Paralelismo, simplicidade de comportamentos individuais e ligação direta *SENSE-ACT* permitem operação em tempo real.
- Processamento local da informação sensorial.
- Prototipação rápida para poucos comportamentos.

Arquiteturas Reativas: Problemas

- Implementação de um grande conjunto de comportamentos é uma tarefa difícil.
- Combinação de comportamentos reativos não garante sucesso na execução da tarefa (defensores da abordagem falam em *inteligência emergente*).
  - Ex: na REACT, pode haver pontos de campo nulo.
- Difícil definição de um conjunto mínimo de comportamentos reativos no caso geral.

Arquiteturas

- Agente tabela
- Agente reativo
- Agente baseado em modelo
- Agente baseado em objetivos
- Agente baseado em utilidade
- Agente aprendiz

Necessidade de metas/objetivos

- Além do estado interno, um agente precisa de alguma informação a respeito de metas, indicando situações desejáveis, para decidir a melhor ação a executar.
- Assim, pode combinar as informações do impacto de suas ações com seus objetivos, de modo a fazer considerações acerca do futuro (predições) e decidir melhor suas ações.
  - O agente poderá ter que considerar longas seqüências de ações encadeadas para poder atingir sua meta ➔ **busca** e **planejamento** são subáreas de IA que visam determinar a **sequência de ações** que leva o agente ao objetivo.

Resolvendo Problemas através de Busca

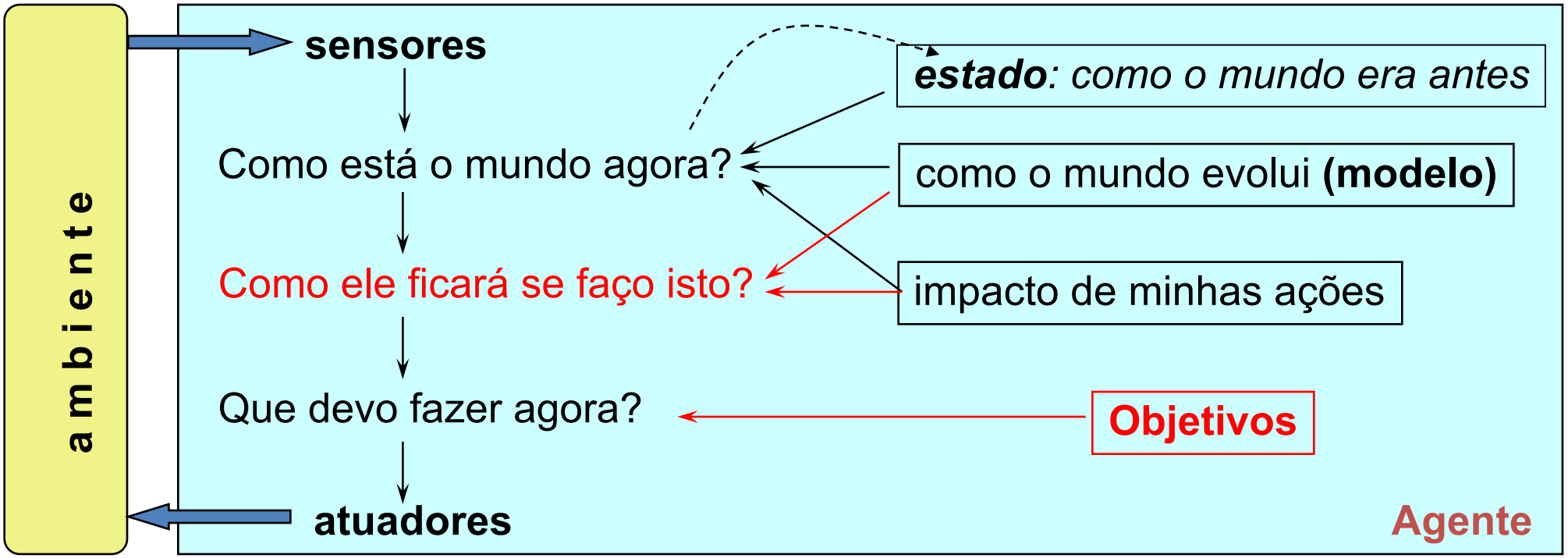
Inteligência Artificial

PCS3438

Escola Politécnica da USP
Engenharia de Computação (PCS)

Agente solucionador de problemas (guiado por objetivo – deliberativo)

- Busca uma *sequência de ações* que o leve a estados desejáveis (*objetivos*).



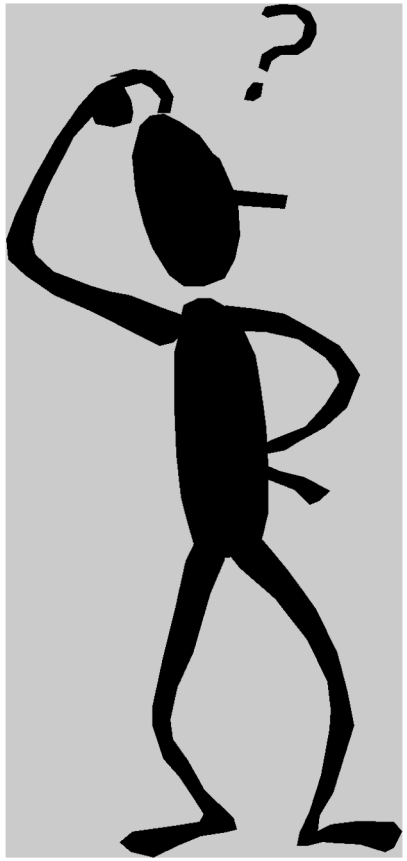
Agente solucionador de problemas

- Propriedades do ambiente para este agente:
  - **Estático**: não muda enquanto o agente delibera
  - **Discreto**: enumera sequências de estados e alternativas de ações
  - **Determinístico**: solução é uma sequência definida de ações
    - não lida com eventos inesperados ou incertezas
    - executa a sequência definida sem considerar percepções ➔ sistema de controle em malha aberta
  - **Observável:** observa completamente os estados e sabe seu estado inicial

- *Algumas flexibilizações serão feitas em relação às propriedades de determinismo e observabilidade.*

Agentes solucionadores de problemas

- O que é um problema em I.A.?
- Como formulá-lo?
- Como buscar a solução do problema?
- Como avaliar a solução e o processo de encontrá-la?

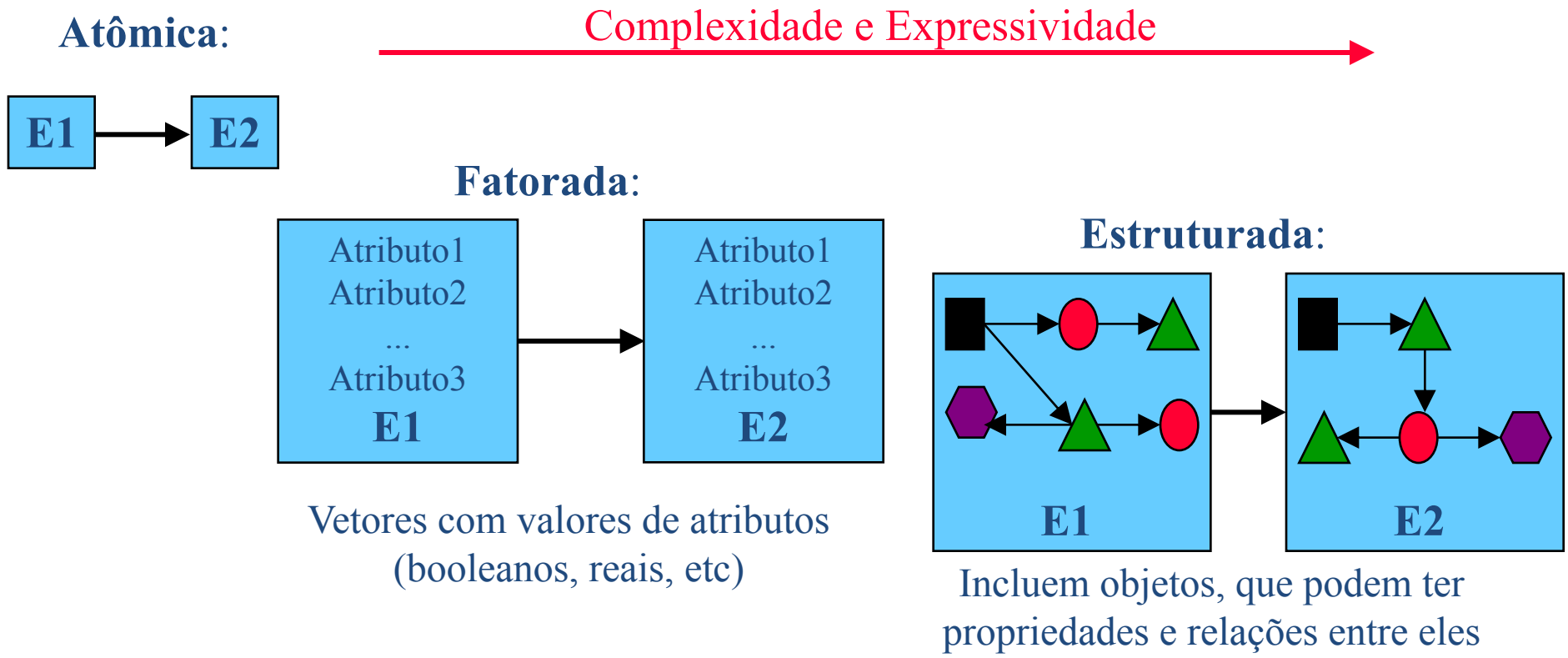


Definição de Problema: Quatro componentes

- **Estado inicial** do problema (onde o agente inicia)
- Descrição das possíveis **ações** do agente:
  - Pela função sucessor: dado um estado *x*, ***suc(x)*** retorna um conjunto de pares ordenados ***(a,y),*** onde ***a*** indica cada ação válida em ***x*** e ***y*** é o estado sucessor.
  - Pelo conjunto de operadores que podem ser aplicados em um estado para gerar os sucessores.
- Um teste de **término**:
  - Pode ser um conjunto de estados-objetivos ou
  - Propriedade mais abstrata (ex. cheque-mate em xadrez)
- Uma **função de custo** da solução
  - avalia numericamente cada solução (medida de desempenho)

Representação de Estados

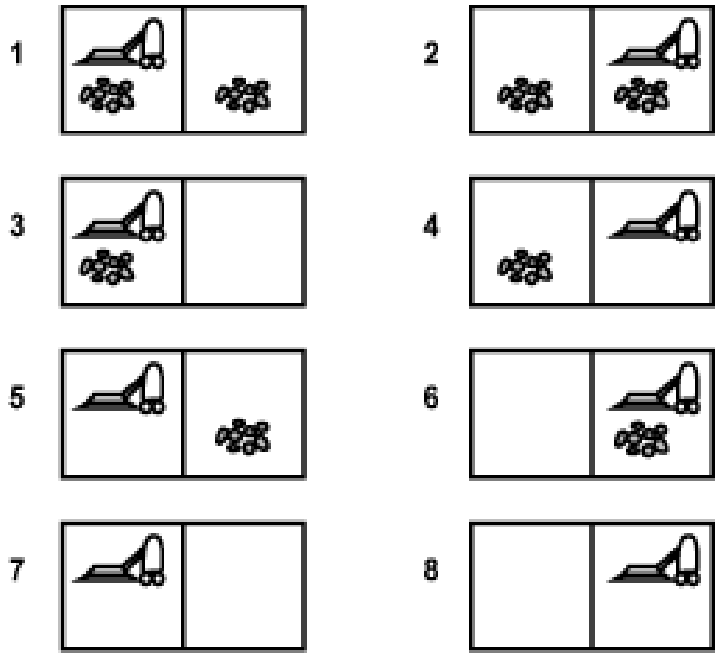
Descrição: Atômica ➔ Fatorada ➔ Estruturada



Definição de Solução

- O estado inicial e a função sucessor implicitamente definem o **espaço de estados** do problema
- O espaço de estados é descrito por um **grafo** onde os vértices representam estados e as arestas, ações.
- Um **caminho** no espaço de estados é uma sequência de estados conectada por uma sequência de ações.
- Uma **solução** para um problema é um caminho do estado inicial para um estado meta (objetivo).
- A **qualidade** da solução é medida pela função de custo da solução.

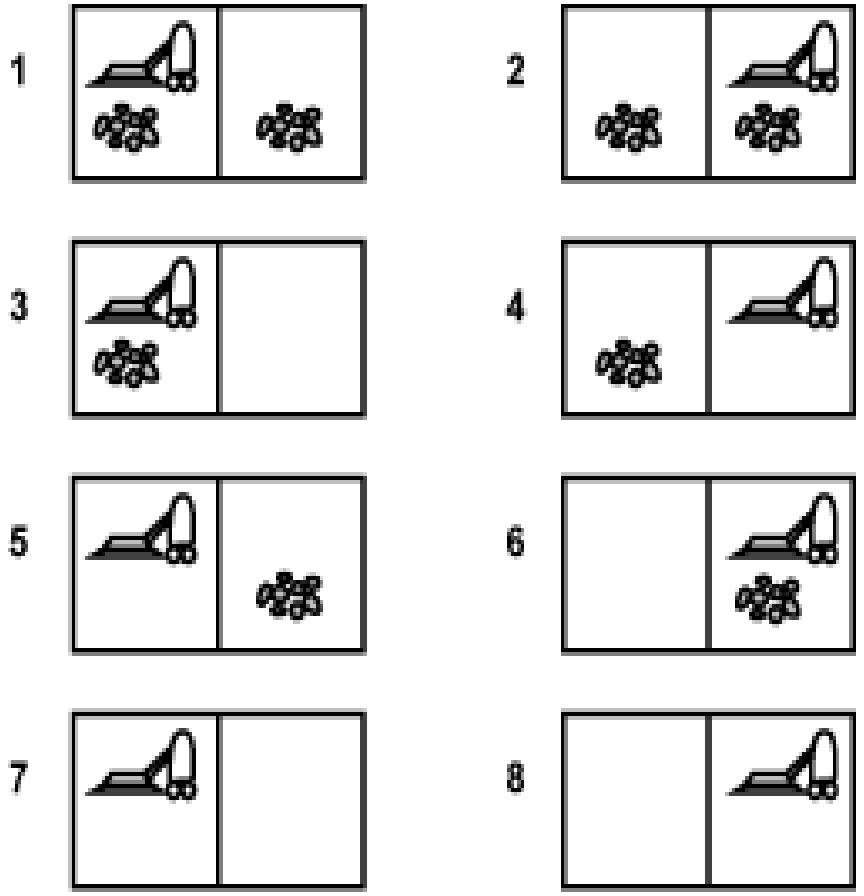
Exemplo 1: Agente Aspirador de Pó



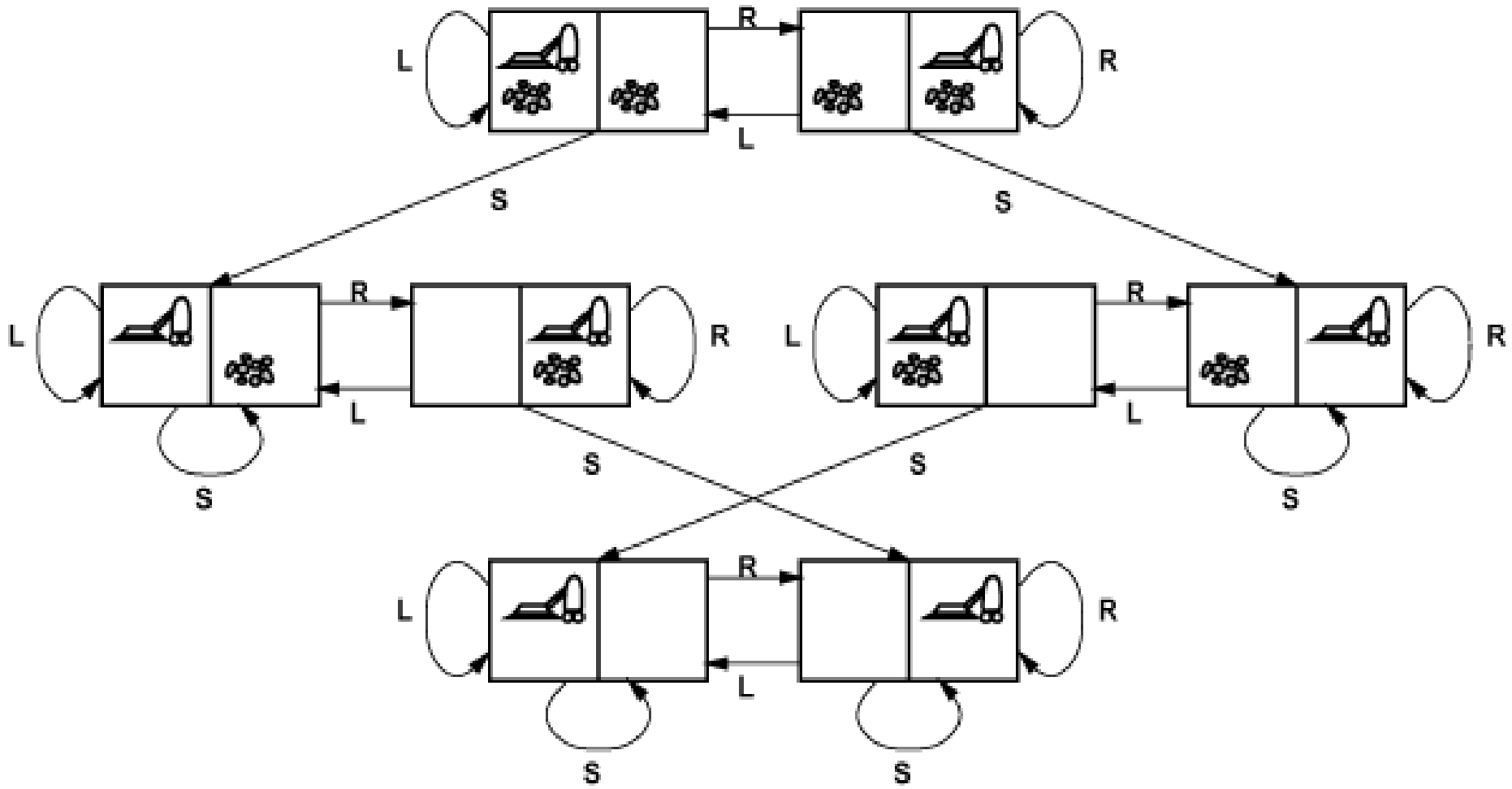
- Formulação do problema:
  - estado inicial = qualquer um dos 8 estados acima
  - função sucessor: operadores = mover direita (R), mover esquerda (L), aspirar (S); suc(1) = {(R,2), (L,1), (S,5)}, etc..
  - teste de término = os dois quartos limpos (estado 7 ou 8)
  - custo do caminho = quantidade de ações realizadas (custo 1 para cada ação)

Exercício 1:

- Considerando os oito estados possíveis (ao lado), representar o espaço de estados como um grafo, com os estados como vértices e as ações como arestas.
- Ações: R, L e S



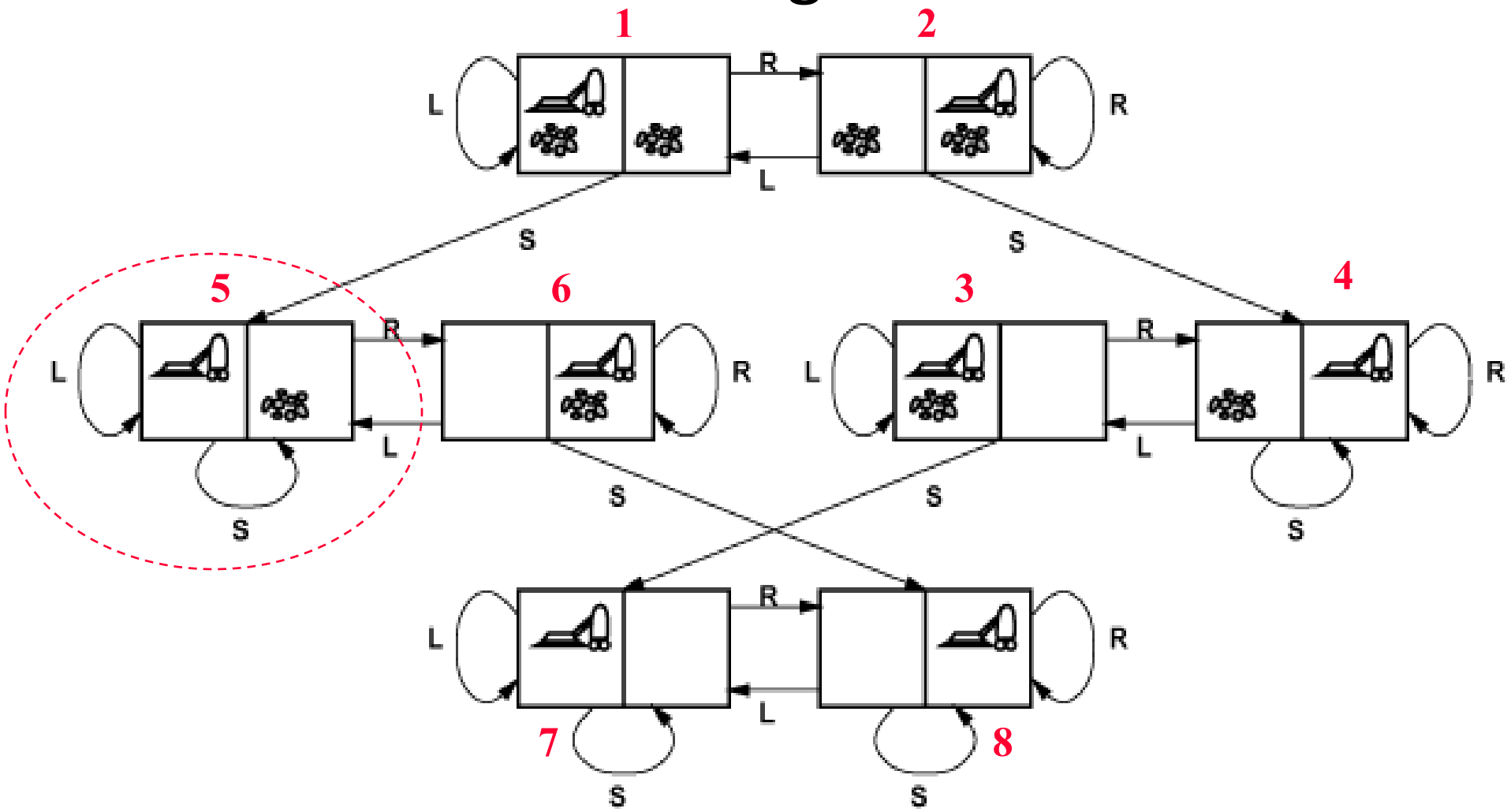
Espaço de estados do agente aspirador



Exercício 2

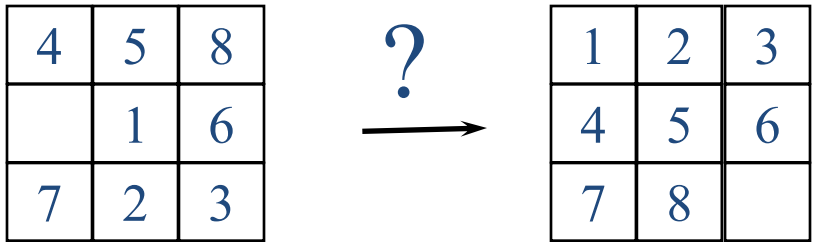
- Considere que o agente inicie no estado 5.
- Utilizando o espaço de estados do problema, encontre pelo menos três soluções para o problema.

Problema: se agente em 5?



Solução 1: LLSRRS ➔ CUSTO=6
Solução 2: SRRS ➔ CUSTO=4
Solução 3: RS ➔ CUSTO=2 .....

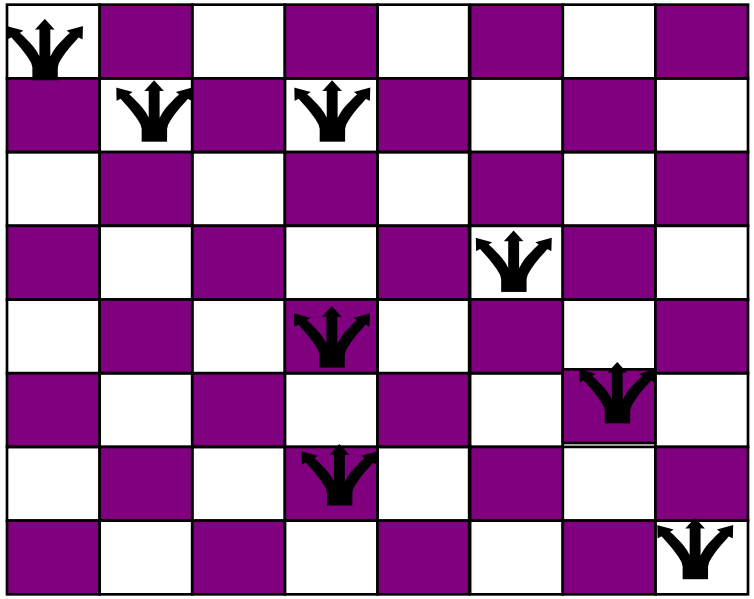
Exemplo 2: Jogo dos 8 Números



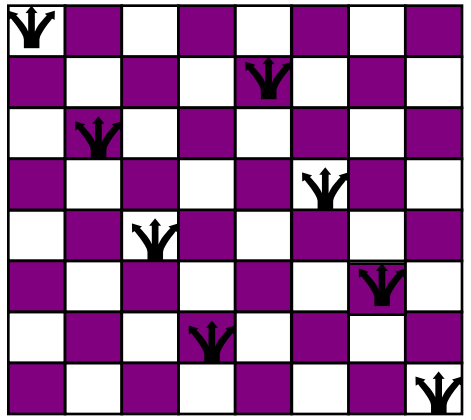
- Estado inicial:
  - cada estado especifica a posição de cada uma das 8 peças e do branco no tabuleiro de 9 posições
  - O estado inicial pode ser qualquer estado
- Função sucessor:
  - Gera os estados possíveis que resultam ao aplicar cada uma de 4 ações: branco para esquerda (L), para a direita (R), para cima (U), para baixo (D)
- Teste de término
  - Números ordenados, branco em [3,3].
- Custo do caminho
  - quantidade de ações realizadas (custo 1 para cada ação)

Importância da formulação (1)

- Jogo das 8 Rainhas
  - Teste de término: dispor 8 rainhas de forma que não possam se "atacar"
  - Custo da solução: ignorado aqui.



Importância da formulação (2)



- **Formulações:**
  - Incremental: envolve operadores que "crescem" a descrição de estado, iniciando com um tabuleiro vazio
  - Estado completo:
    - ***Estado inicial****: qualquer estado com disposição das 8 rainhas, uma em cada coluna*
    - ***Função sucessor****: retorna todos os possíveis estados ao mover uma rainha para outra casa na mesma coluna (cada estado tem 8x7=56 sucessores).*

Importância da formulação (3)

- Formulação incremental (1)
  - **Estado inicial**: tabuleiro sem rainhas; estado: qualquer disposição de 0 a 8 rainhas no tabuleiro
  - **Função sucessor**: adicionar uma rainha a qualquer célula vazia
  - **Teste de término**: 8 rainhas sem ataque mútuo

  Nesta formulação o espaço de estados é:

  = 64 x 63 x ... X 57 $\approx$ 3 x 10^{14}

Importância da formulação (4)

- Formulação incremental (2)
  - **Estado inicial**: tabuleiro sem rainhas; estado: tabuleiro com n ($0 \leq n \leq 8$) rainhas dispostas na n-ézima coluna mais à esquerda sem ataque mútuo
  - **Função sucessor**: adicionar uma rainha em qualquer casa na coluna vazia mais à esquerda de forma que não possa ser atacada (**teste gradual**)

Formulação melhor (espaço de estados bem menor!)

Como encontrar a solução?

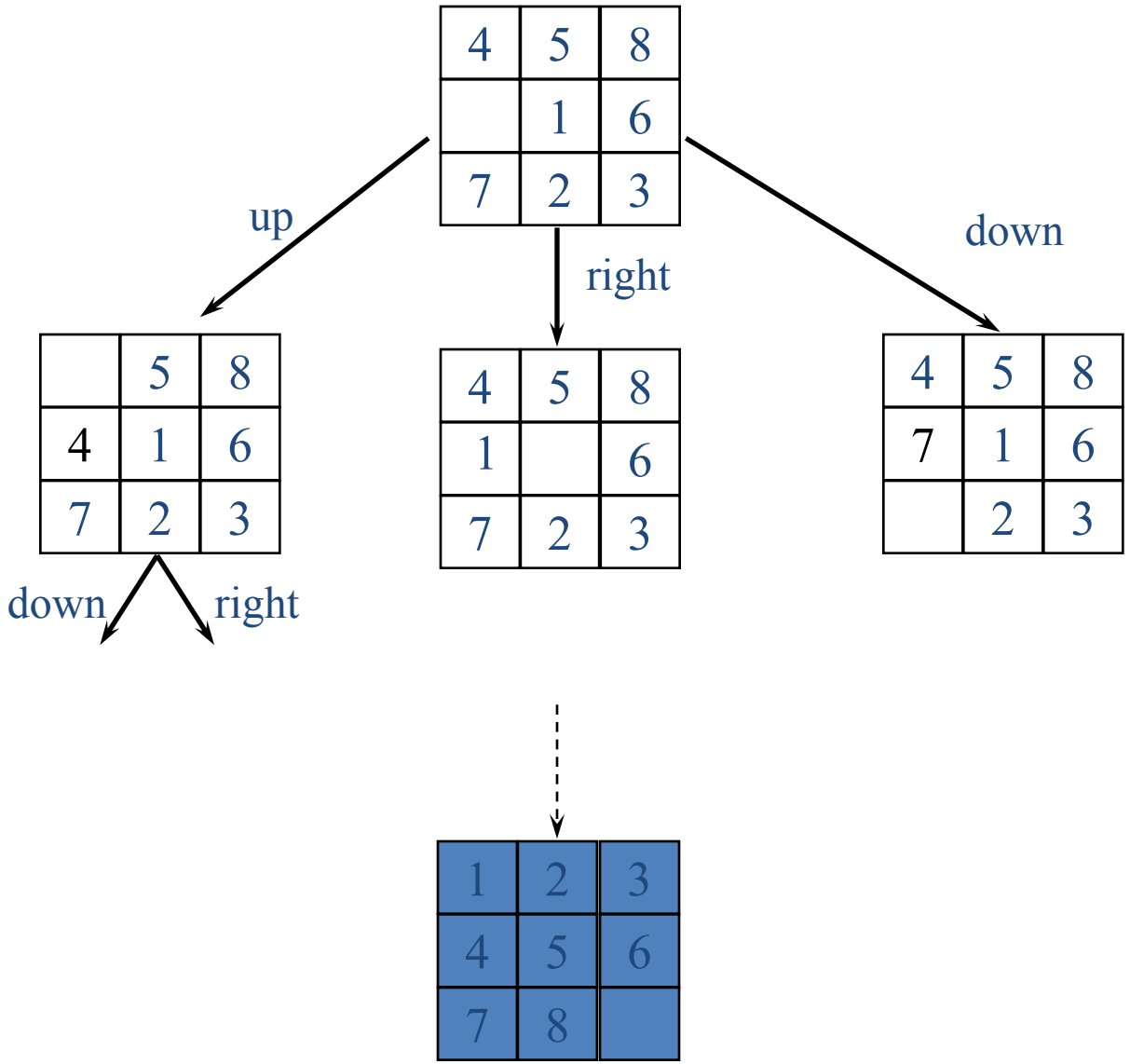
- Uma vez o problema bem formulado, o estado meta deve ser **buscado** no espaço de estados
- A busca é representada em uma **árvore de busca**:
  1. **Raiz**: corresponde ao estado inicial
  2. **Expande-se o estado corrente**: aplica-se a função sucessor ao estado corrente, gerando um novo conjunto de sucessores
  3. **Escolhe-se o próximo estado** a expandir seguindo uma **estratégia de busca**
  4. **Prossegue-se** até sucesso (atingir estado meta – retorna solução) ou falha

Como encontrar a solução?

- Espaço de estados ≠ árvore de busca

- Ex.: TSP com 20 cidades
  - espaço de estados = 20 vértices (=cidades)
  - árvore de busca com infinitos vértices (na prática, a busca evita repetir estados)

Árvore de busca para o "jogo dos 8 números"



Medida de Desempenho na Busca (1)

- Desempenho de um algoritmo de busca:
  - **Completo**: se existir uma solução, ela certamente é encontrada
  - **Ótimo**: a busca encontra a solução de menor custo
  - **Complexidade temporal**: quanto tempo demora para encontrar a solução
  - **Complexidade espacial**: quanta memória é usada para realizar a busca

Medida de Desempenho na Busca (2)

- Em IA a árvore de busca é tipicamente infinita ➔ complexidade é expressa por:
  - **b** – fator de ramificação (*branching*) ou número máximo de sucessores de um nó;
  - **d** – profundidade (*depth*) do nó-meta mais próximo da raiz;
  - **m** – comprimento máximo de um caminho no espaço de estados.

Medida de Desempenho na Busca (3)

- Custo total = custo da solução + custo da busca
  - custo da solução (ex. TSP: caminho a percorrer, em km)
  - custo da busca (tipicamente depende da complexidade em tempo)

  Problema: relacionar custo da solução (km) com o da busca (seg)
- Espaço de estados grande:
  - compromisso (conflito) entre a melhor solução (menor custo da solução) e a solução mais barata (menor custo da busca)

Métodos de Busca

- **Busca cega** (ou busca não informada)
  - Não tem informação sobre qual sucessor é mais promissor para atingir a meta.
  - Estratégias de Busca (ordem de expansão dos nós):
    - busca em largura
    - busca de custo uniforme
    - busca em profundidade
    - busca em profundidade limitada
    - busca em profundidade com aprofundamento iterativo
    - busca bidirecional
- **Busca heurística** (ou busca informada)
  - Possui informação (estimativa) de qual sucessor é mais promissor para atingir a meta.