Inteligência Artificial
Ciência da Computação

Prof. Aline Paes / alinepaes@ic.uff.br

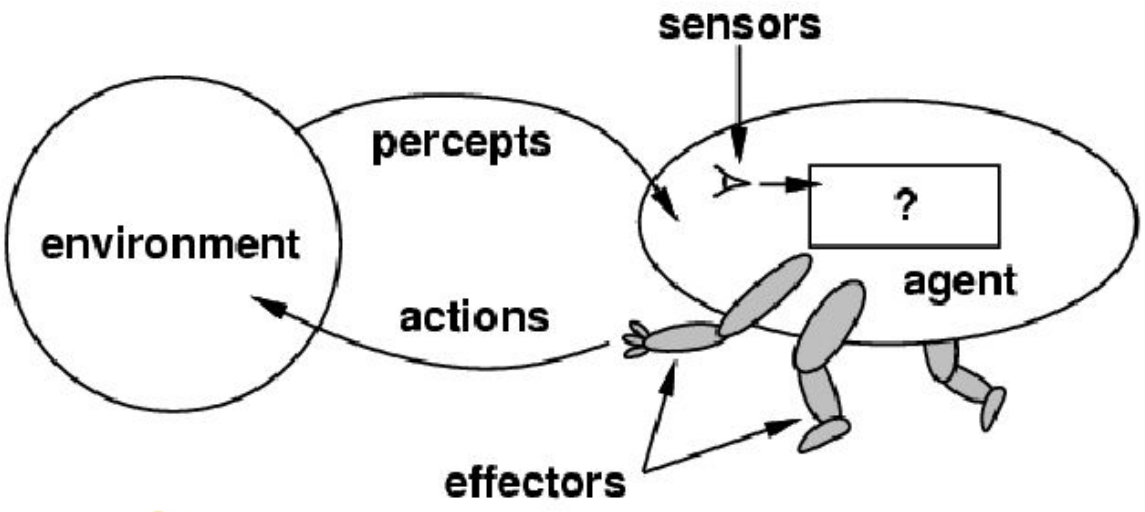
Formulação de problemas e ambientes
cap 2a, 3a, RN; Caps. 1,2 ER

Universidade Federal Fluminense

Instituto de Computação

Agentes e ambiente

- Agente é algum artefato capaz de perceber seu ambiente por meio de sensores e agir no ambiente por meio de atuadores
- Programas que operam de forma autônoma



Agentes

- Agente humano
  - sensores?
  - atuadores?
- Agente robótico
  - sensores?
  - atuadores?
- Agente de software
  - sensores?
  - atuadores?

Percepções e ações

- Percepção: entrada para o agente
- Sequência de percepções: histórico de entradas
- Comportamento do agente é definido pela função do agente
  - mapeia uma sequência de percepções para uma ação

$$f = P^* \rightarrow A$$

Percepções e Ações

- Robótica
  - Percepção: medidas de sensores (câmeras, microfones, laser range finders, sonar, GPS)
  - Ações: mover, dar a volta, pegar um objeto, etc.

- Visão computacional
  - Percepções: pixels de uma imagem
  - Ações: produzir descrições de objetos em uma imagem

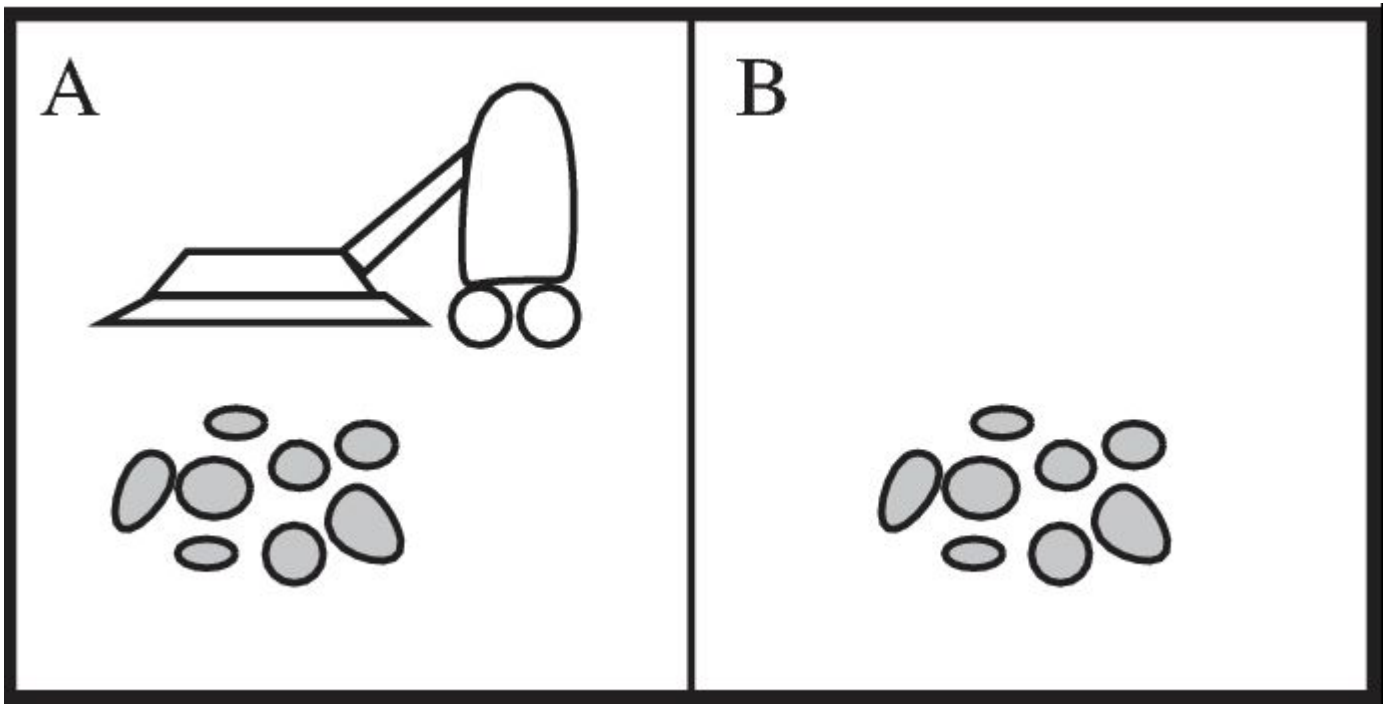
Percepções e Ações

- Linguagem natural
  - Percepções: requisições em um contexto (*Onde é o aeroporto mais próximo?*)
  - Ação: resposta (*Santos Dumont*)

- Jogos
  - Percepções: estado de um tabuleiro de xadrez
  - Ação: executar um movimento legal

O aspirador de pó

- Percepção: local e conteúdo



O aspirador de pó: tabulação de ações

Sequência de Percepções	Ação
[A, Limpo]	Direita
[A, Sujo]	Aspirar
[B, Limpo]	Esquerda
[B, Sujo]	Aspirar
[A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Sujo]	Aspirar
...	...
[A, Limpo], [A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Limpo], [A, Sujo]	Aspirar
...	...

Agente racional

- Agente que faz a coisa certa, baseado em suas percepções, com o objetivo de ter sucesso

- Como medir o sucesso?
  - medida de desempenho
    - quantidade de energia gasta, sujeira coletada, tempo gasto

Agente racional

- "*Para cada possível sequência de **percepções**, um agente racional deve selecionar a **ação** que espera-se **maximizar a medida de performance**, dada evidência fornecida pela **sequência de percepções** e qualquer outro **conhecimento que o agente tenha***"

Modelagem de um agente

- Desempenho
- Ambiente
- Atuadores
- Sensores

Agente: Motorista de taxi automatizado

- Desempenho:
  - seguro, rápido, cumpre as leis, conforto, lucro
- Ambiente:
  - Ruas/estradas, outros carros, pedestres, clientes
- Atuadores:
  - direção, acelerador, freios, buzina, embreagem, setas, alguma forma de comunicação
- Sensores:
  - câmeras, sonar, velocímetro, GPS, sensores do motor, teclado

Agente: Sistema de diagnóstico médico

- Desempenho
  - paciente saudável, custos reduzidos
- Ambiente
  - paciente, hospital, equipe
- Atuadores
  - tela de questões, testes, diagnósticos, tratamentos
- Sensores
  - teclado, respostas do paciente

Propriedades do ambiente

- **Completamente observável**
  - sensores do agente conseguem capturar todos os aspectos relevantes do ambiente

- **Parcialmente observável**
  - sensores do agente não conseguem capturar todas as informações sobre o estado do ambiente
  - pode resultar de ruído ou sensores com ruído

Propriedades do ambiente

- **Agente único**
  - apenas um agente opera no ambiente
- **Multi-agentes**
  - vários agentes interagindo no ambiente
    - cooperando
    - competindo
  - desempenho do agente também depende das ações dos outros agentes

Propriedades do ambiente

- **Determinístico**
  - O próximo estado do ambiente é completamente determinado pelo estado atual e pela ação executada pelo agente.

- **Não determinístico**
  - Não se tem certeza do que pode acontecer com o ambiente ao executar uma ação.
  - **Estocástico**: ações dependem de probabilidades

Propriedades do ambiente

- **Episódico**
  - Próximos episódios (percepção-ação) não dependem de ações anteriores
- **Sequencial**
  - decisão corrente pode afetar decisões futuras
  - acontece frequentemente em jogos

Propriedades do ambiente

- **Estático**
  - O ambiente não muda entre o tempo de percepção e ação
- **Dinâmico**
  - O ambiente pode mudar enquanto o agente pensa ou está executando uma ação
- **Semi-dinâmico**
  - o ambiente não muda, mas o agente sim

Propriedades do ambiente

- **Discreto**
  - Um número limitado e definido de percepções, ações e estados
- **Contínuo**
  - Um número possivelmente infinito de percepções, ações e estados.

Carro robótico: propriedades

- Informação do carro: posição, orientação, velocidade, aceleração, ...
- Informação do ambiente: mapas das estradas, ruas, rodovias, sinais (semáforo e paradas), outros veículos, pedestres, ciclistas, ...
- Ações: velocidade, frear, mudança de direção, ...
- Sensores: câmeras, GPS, …

Carro robótico: propriedades do ambiente

- Parcialmente observável
- Estocástico
- Dinâmico
- Contínuo
- Multi-agente
- Sequencial

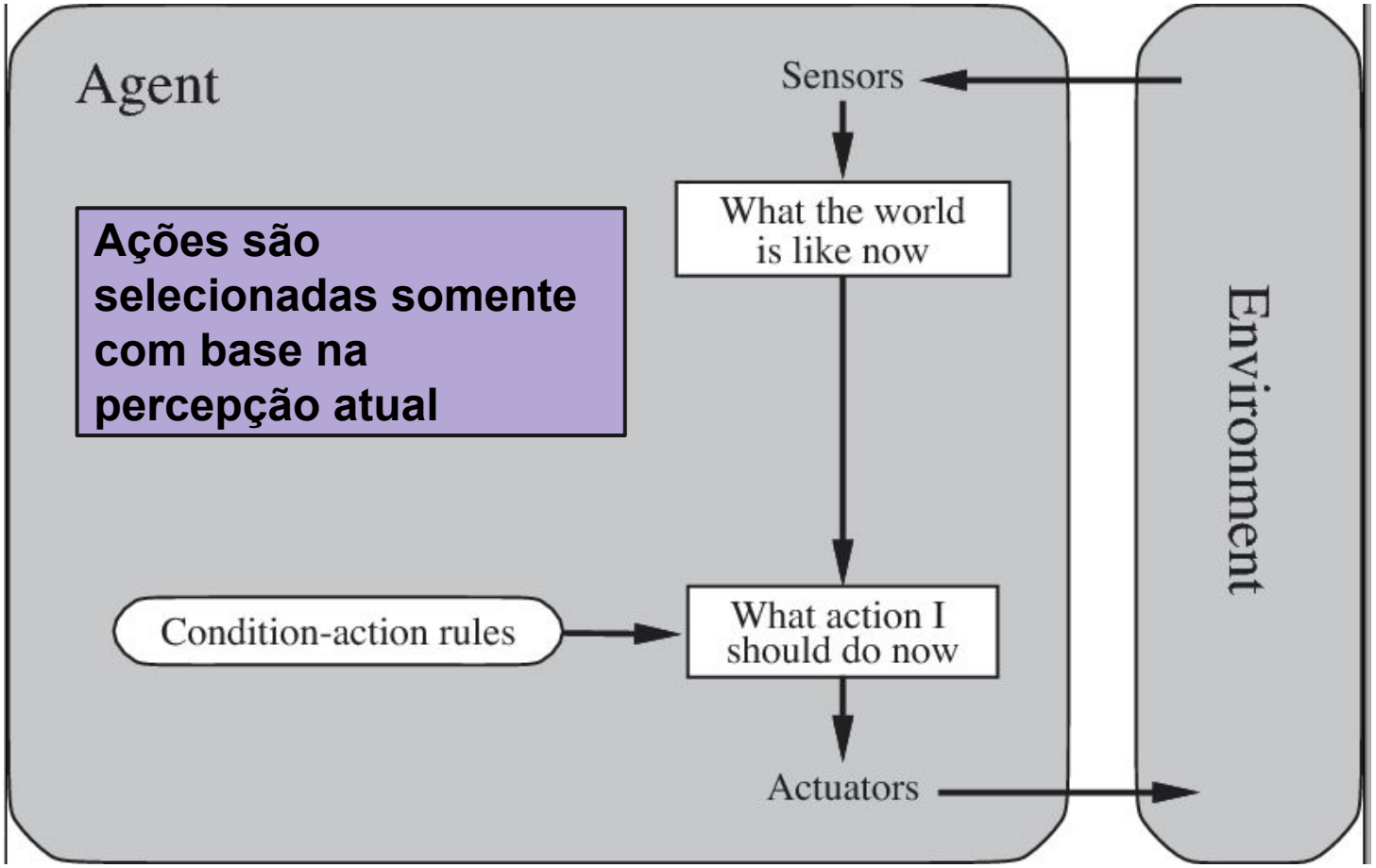
Exemplo

Tarefa no Ambiente	**Observável**	**Agente**	**Determinístico**	**Episódico**	**Estático**	**Discreto**
Análise de imagens						
Palavras cruzadas						
Diagnóstico Médico						
Poker						
Xadrez com relógio						
Backgamão						

Exemplo

Tarefa no Ambiente	Observável	Agente	Determinístico	Episódico	Estático	Discreto
Análise de imagens	Sim	único	Sim	Sim	Sim	Não
Palavras cruzadas	Sim	único	Sim	Não	Sim	Sim
Diagnóstico Médico	Parcialmente	único	Não	Não	Não	Não
Poker	Parcialmente	Multi	Não	Não	Sim	Sim
Xadrez com relógio	Sim	Multi	Sim	Não	Semi	Não
Backgamão	Sim	Multi	Não	Não	Sim	Sim

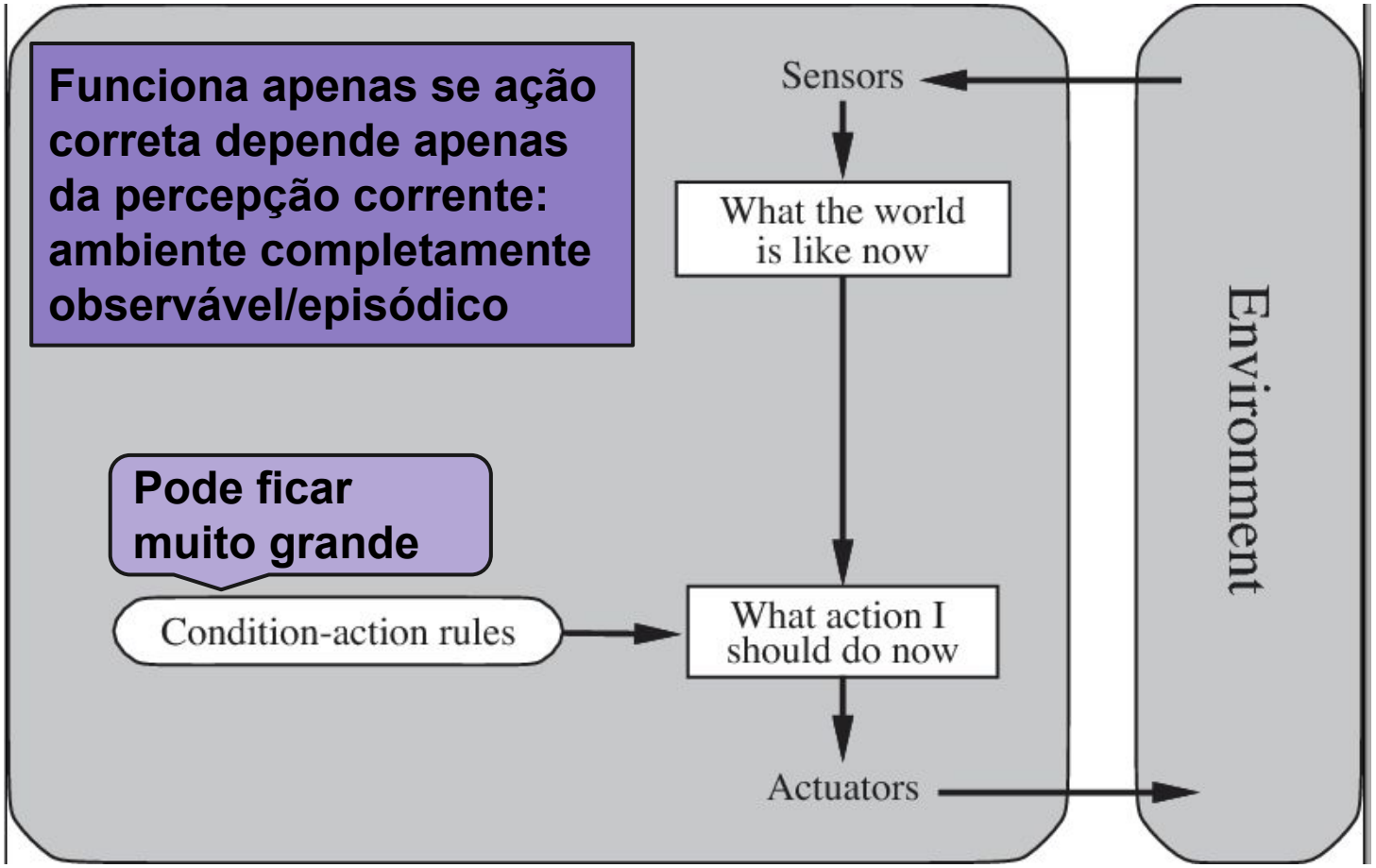
Tipos de agentes: reativo simples



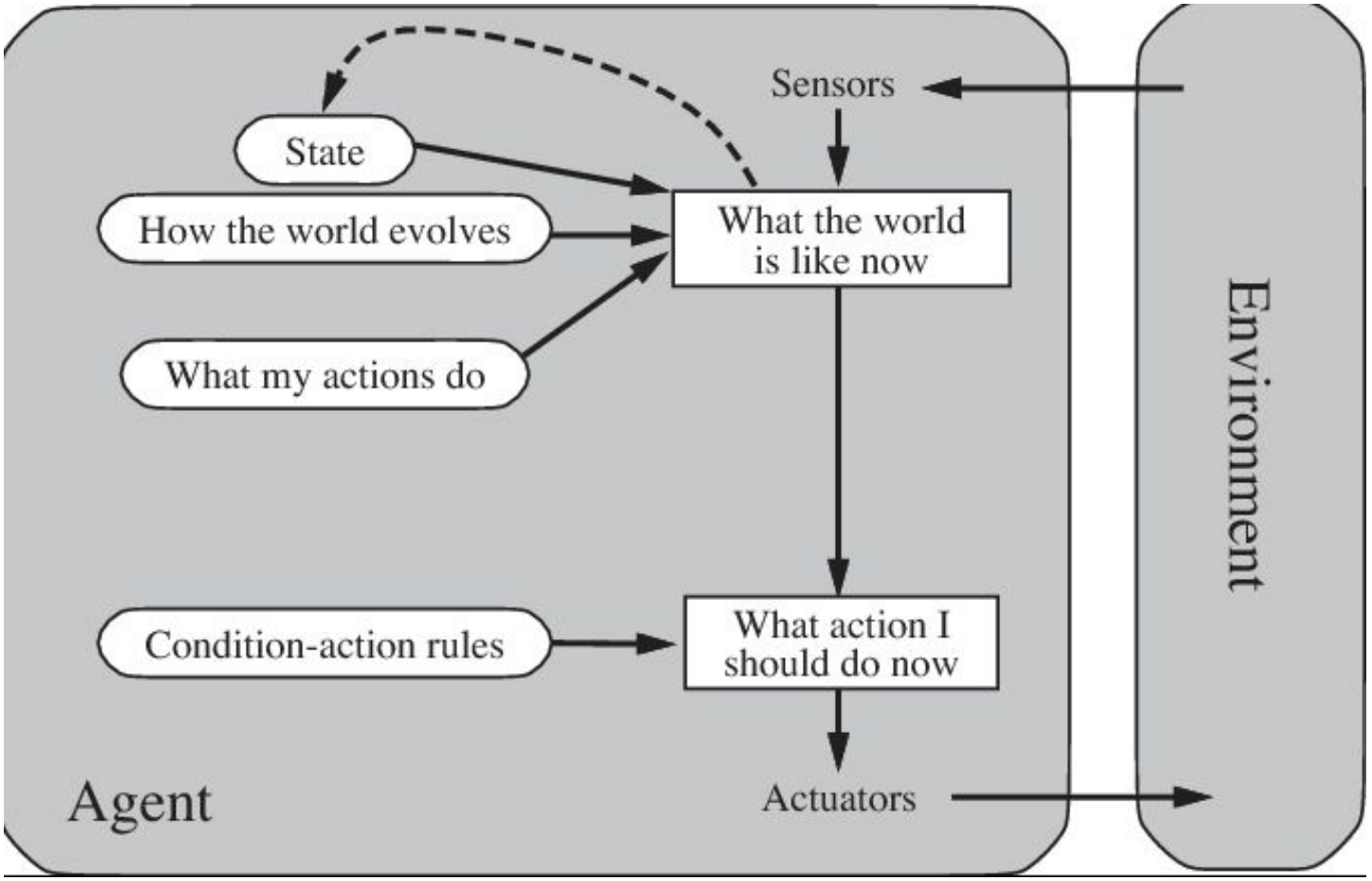
Agente reativo simples

function SIMPLE-REFLEX-AGENT(*percept*) **returns** *action*
    static:*rules*, a set of condition action rules
    state ← INTERPRET-INPUT(*percept*)
    rule ← RULE-MATCH(*state, rules*)
    action ← RULE-ACTION(*rule*)
    return *action*

Tipos de agentes: reativo simples



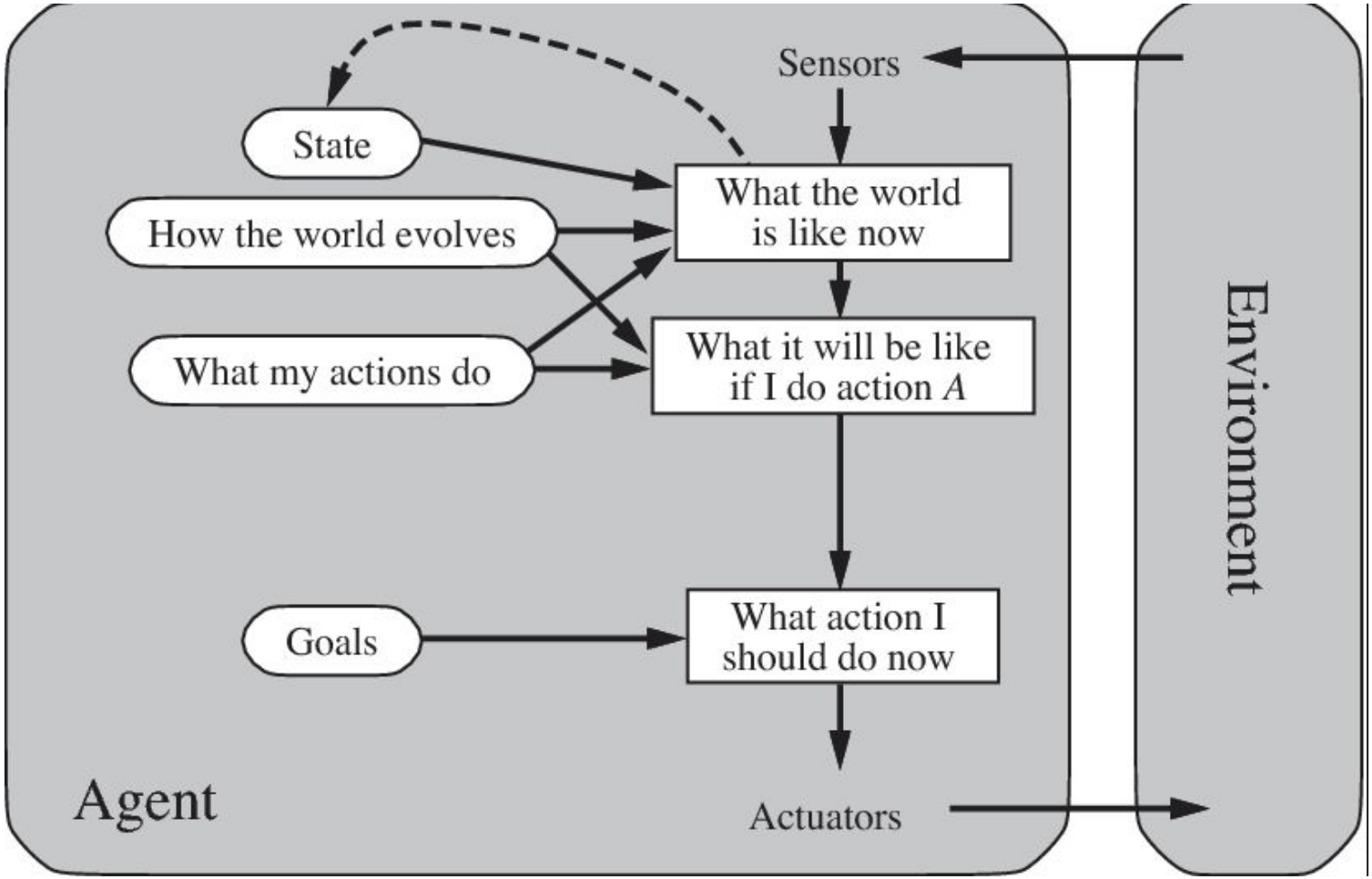
Agentes baseados em modelos: ambientes parcialmente observáveis



Agentes baseados em modelos

function REFLEX-AGENT-WITH-STATE(*percept*) **returns** an *action*
    static: *state*, a description of the current world state
        rules, a set of condition-action rules
        action, the most recent action, initially none
    state ← UPDATE_INPUT(*state, action, percept*)
    rule ← RULE_MATCH(*state, rules*)
    action ← RULE_ACTION[*rule*]
    return *action*

Agentes baseados em objetivos



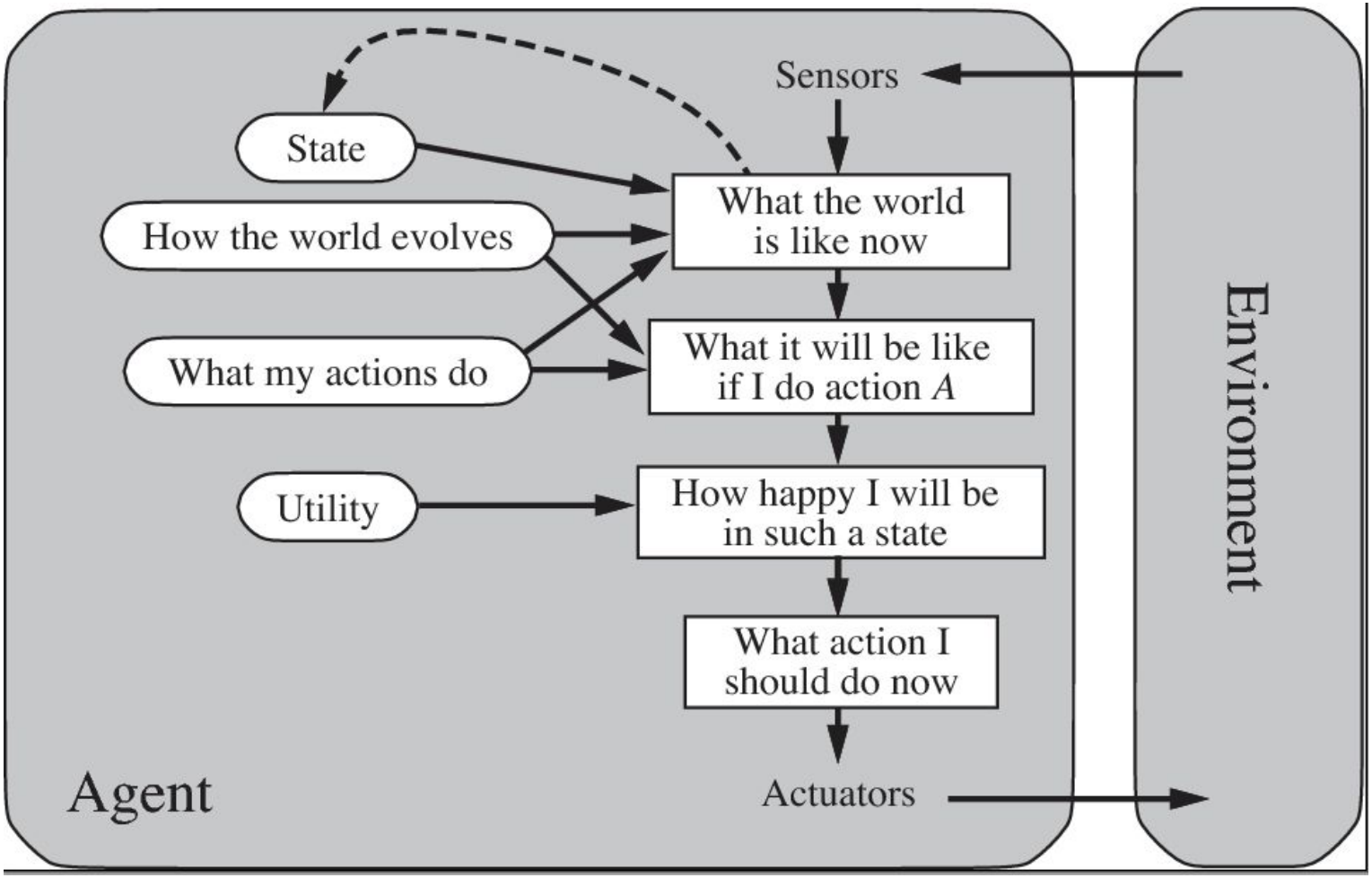
Agente baseado em objetivos

function GOAL -BASED-AGENT(*percept*) **returns** an *action*
    static: *state*, the agent's current conception of the world state
        model , a description of how the next state depends on current state and action
        goal , a description of the desired goal state
        plan, a sequence of actions to take, initially empty
        action, the most recent action, initially none
    state ← UPDATE-STATE(*state, action , percept , model*)
    if GOAL-ACHIEVED (state,goal) then **return** a null action
    if *plan* is empty **then**
        plan ← PLAN(*state,goal ,model*)
        action ← FIRST(*plan*)
        plan ← REST(*plan*)
    return *action*

Agentes baseados em utilidade

- Grau de satisfação
- Estados mais desejáveis terão mais utilidade para o agente
- Função que mapeia um estado para um número real

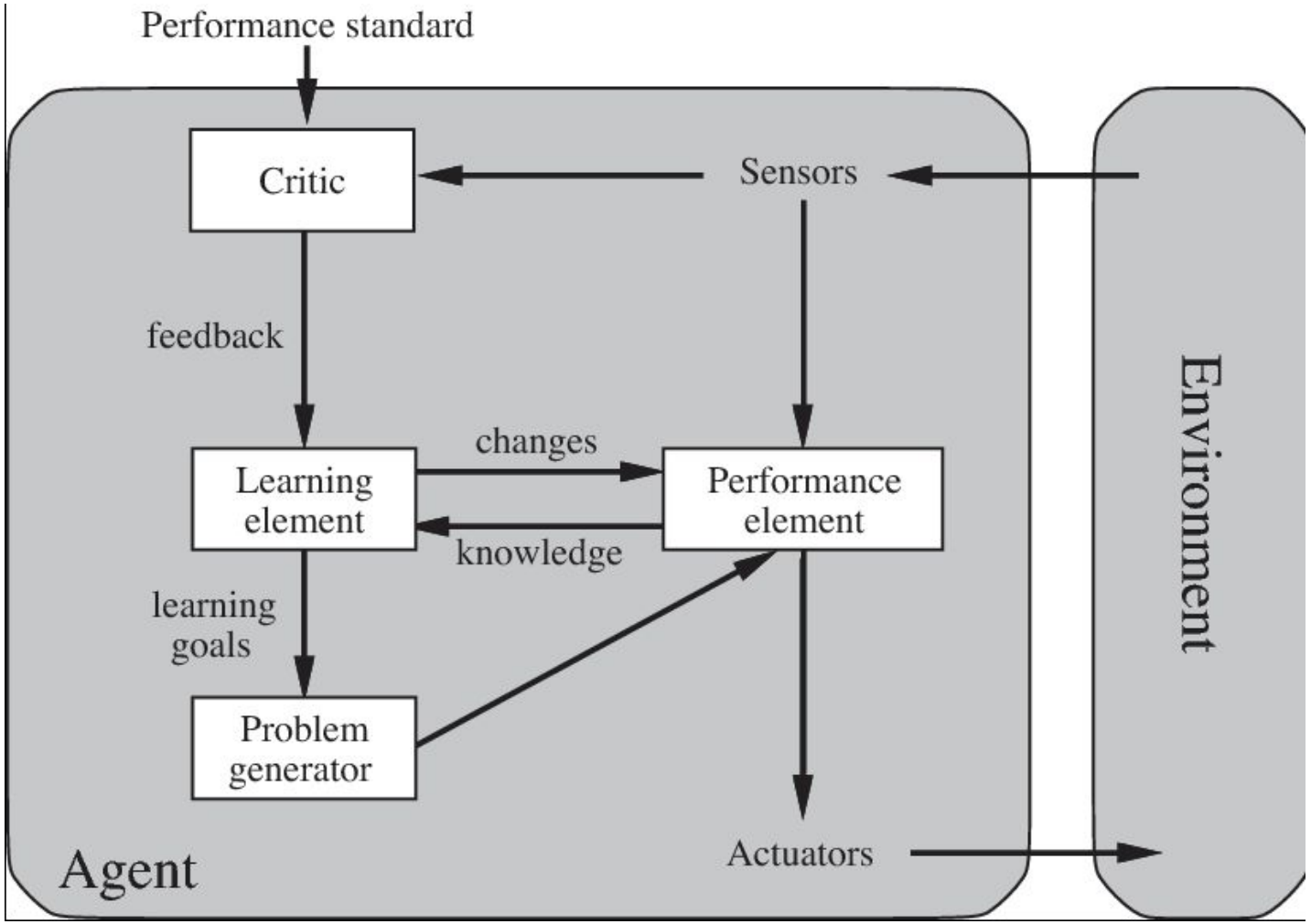
Agentes baseados em utilidade



Agente baseado em utilidade

function UTILITY-BASED-AGENT(*percept*) **returns** an *action*
    static: *state*, the agent's current conception of the world state
        model , a description of how the next state depends on current state and action
        utility-function, a description of the uility function
        goal , a description of the desired goal state
        plan, a sequence of actions to take, initially empty
        action, the most recent action, initially none
    state ← UPDATE-STATE(*state, action , percept , model*)
    if *plan* is empty **then**
        plan ← PLAN(*state, utility-function ,model*)
        action ← FIRST(*plan*)
        plan ← REST(*plan*)
    return *action*

Agentes com aprendizado



Resolução de problemas

- Técnicas de IA consideram
  - complexidade da solução
  - generalizações
  - uso de conhecimento preliminar
  - abstração
  - busca pela solução

Declaração de problemas a serem resolvidos

- Jogue Xadrez!
- É suficiente?

Descrição de um problema

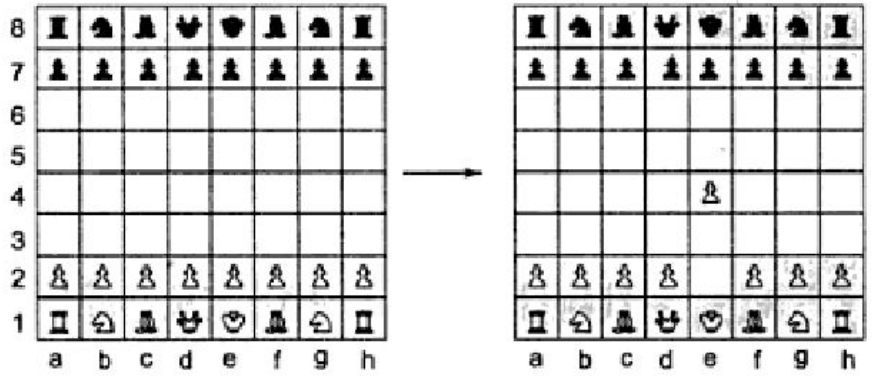
- Estado
  - ocomo representar o tabuleiro
  - oconfiguração inicial do tabuleiro
  - oconfiguração que representa vitória
- Ações
  - oregras que definem as jogadas
- Desempenho
  - oexplicitar que não basta jogar: é preciso vencer!

Descrição de um problema

- Representação do tabuleiro:
  - uma estrutura de dados, matriz 8 X 8, p.ex.
  - cada posição contém um símbolo representando a peça (ou o vazio)
- Configuração inicial do tabuleiro
  - vetor no formato de um tabuleiro inicial
- Meta
  - qualquer configuração do tabuleiro em que o rei oponente está sob ataque e ele não tem uma jogada legal
- Jogadas
  - regras definindo os movimentos legais

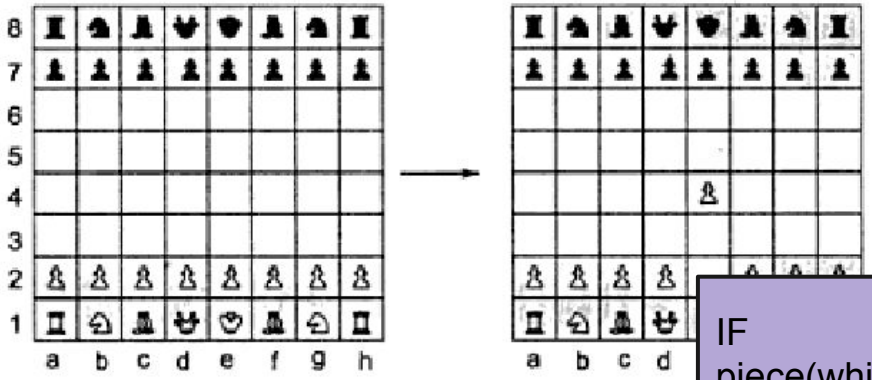
Descrição de um problema

- Regras descrevendo as jogadas
  - lado direito corresponde à configuração corrente
  - lado esquerdo corresponde à configuração depois da jogada
  - Quantas regras?
  - ~10^{120}



Descrição de um problema

- Regras mais eficientes são escritas de forma generalizada



IF
piece(white_pawn) at square(file e, rank 2) AND square(file e, rank 3) is empty AND
square(file e, rank 4) is empty AND
turn(pawn,1)

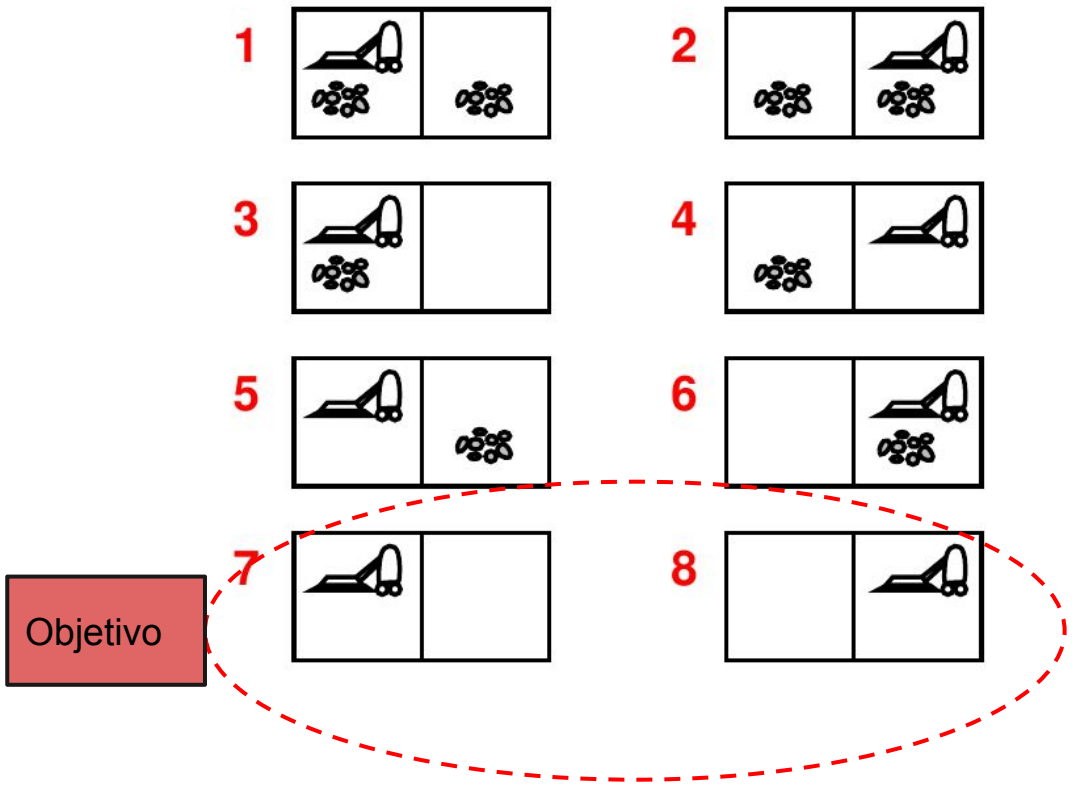
THEN
move piece(white_pawn) from square(file e, rank 2) to square(file e, rank 4)

Representação do espaço de estados

- Definição formal do problema
  - converter uma situação corrente para uma situação desejada
- Definição do processo de resolução do problema
  - combinação de técnicas conhecidas
  - **Busca** pode ser usada para atuar no espaço
    - procurar por uma sequência de ações que alcançam a meta

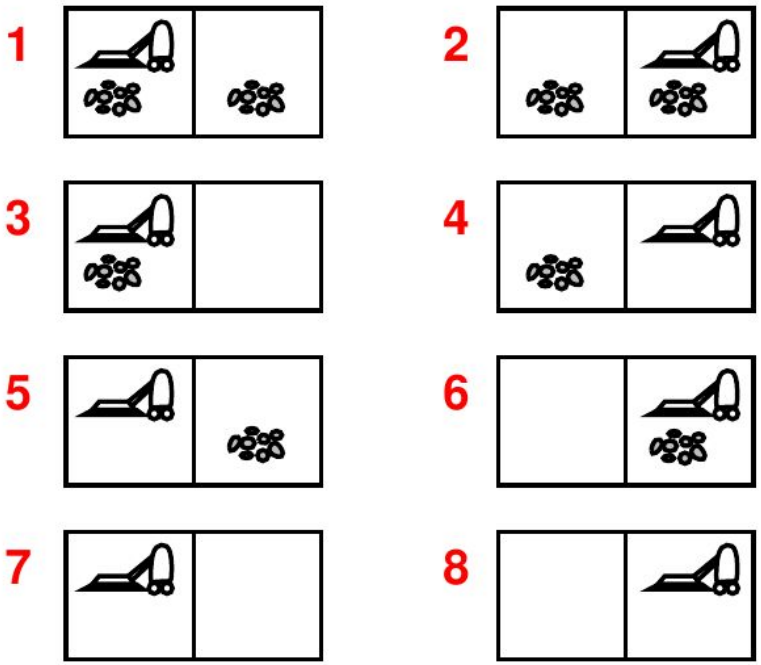
Exemplo: aspirador de pó automatizado

- Se começar no estado 5, qual é a solução?



Exemplo: aspirador de pó automatizado

- Se começar no estado 5, qual é a solução?
  - Vá para a direita (estado 6)
  - e aspire (estado 8)



Componentes de um problema - formalização

- Estado inicial
- Ações disponíveis
  - actions(s): ações que podem ser executadas em um estado ***s***
- Modelo de transição
  - descrição do que cada ação faz
  - result(s,a): estado resultante (**sucessor**) ao aplicar ação ***a*** no estado ***s***

Componentes de um problema - formalização

- Estado inicial
- Ações disponíveis
  - actions(s): ações que
- Modelo de transição
  - descrição do que cada ação faz
  - result(s,a): estado resultante (**sucessor**) da ação ***a*** no estado ***s***

Espaço de estados do problema: conjunto de todos os estados alcançáveis a partir do estado inicial por qualquer sequência de ações

Grafo com os nós sendo os estados e as arestas são as ações

Componentes de um problema - formalização

- Estado inicial
- Ações disponíveis
  - actions(s): ações que podem ser executadas em ***s***
- Modelo de transição
  - descrição do que cada ação faz
  - result(s,a): estado resultante (**sucessor**) ao aplicar ação ***a*** no estado ***s***
- Estado meta
  - específico ou abstrato
- Custo de um caminho no grafo
  - custo de uma sequência de estados: soma das arestas no caminho

Componentes de um problema - formalização

- Estado inicial
- Ações disponíveis
  - actions(s): ações que
- Modelo de transição
  - descrição do que cad
  - result(s,a): estado resultante (**sucesso**
    ação ***a*** no estado ***s***
- Estado meta
  - específico ou abstrato
- Custo de um caminho no grafo
  - custo de uma sequência de estados: soma das arestas no caminho

Solução para o problema: sequência de ações que vão do estado inicial até a meta

Componentes de um problema - formalização

- Estado inicial
- Ações disponíveis
  - actions(s): ações que
- Modelo de transição
  - descrição do que cada ação faz
  - result(s,a): estado resultante (**sucessor**) da ação ***a*** no estado ***s***
- Estado meta
  - específico ou abstrato
- Custo de um caminho no gr
  - custo de uma sequência arestas no caminho

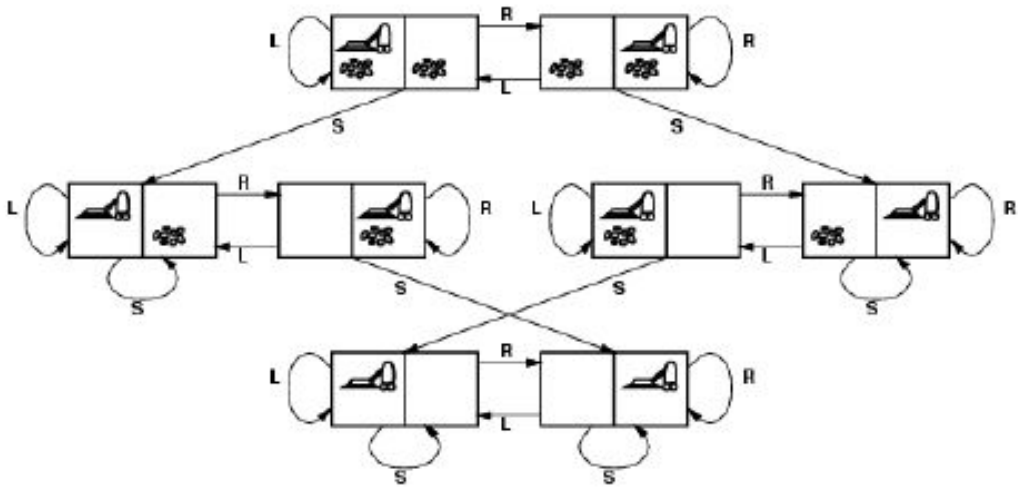
Solução para o problema: sequencia de ações que vão do estado inicial até a meta

Solução ótima: a de menor custo dentre todas as possíveis soluções

Ou seja,

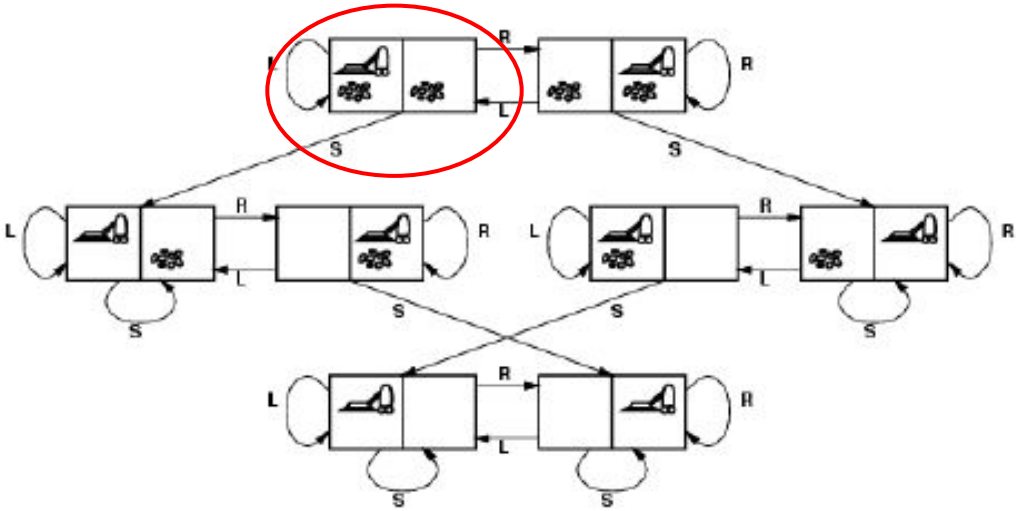
- Um *problema* é um objetivo e um conjunto de meios para atingir o objetivo.
- Para ser resolvido por alguma técnica de IA, o problema é modelado considerando:
  - Estado inicial: por onde começar?
  - Objetivo: onde queremos chegar?
  - Ações: para onde ir a partir de cada estado?
  - Modelo de transição: para onde as ações me levam?
  - Custo: qual o custo ao mudar de estado?

Exemplo: aspirador de pó



Estados: localização do aspirador e da sujeira
2 X 2^2 possíveis estados
Estado:
[Aspirador, Sujeira no 1, Sujeira no 2]

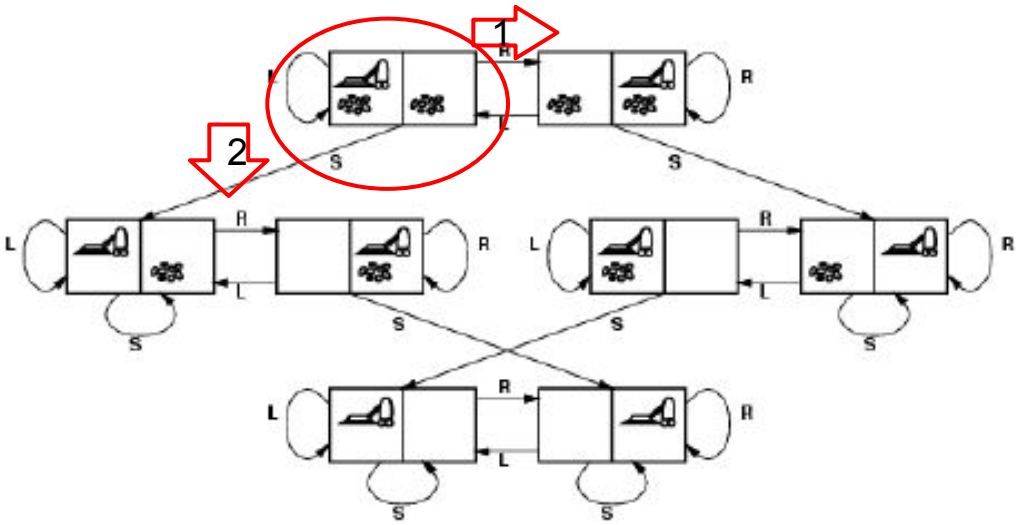
Exemplo: aspirador de pó



Estados: localização do aspirador e da sujeira
at(Aspirador, Sujeira no 1, Sujeira no 2)

- Estado inicial: qualquer um dos estados acima (ex. at(left, yes, yes))

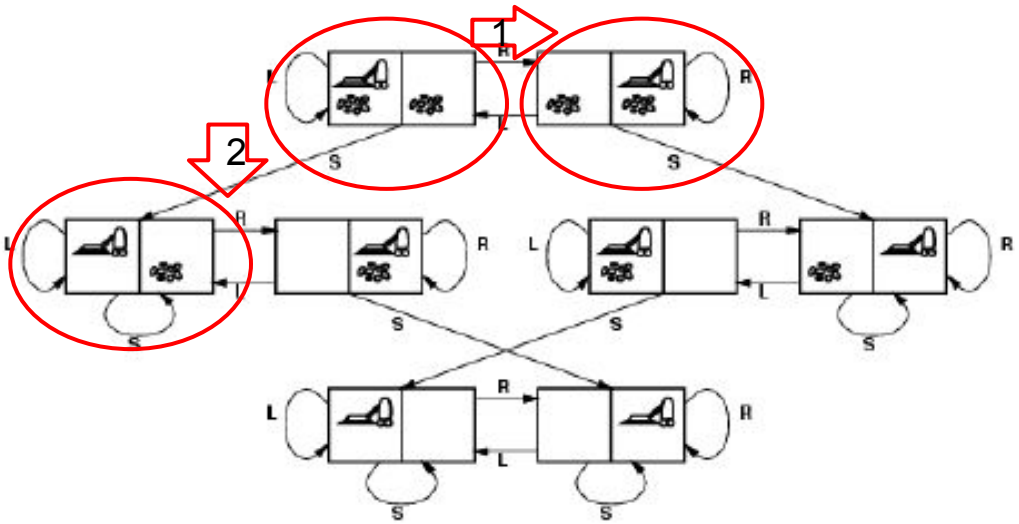
Exemplo: aspirador de pó



Estados: localização do aspirador e da sujeira
at(Aspirador, Sujeira no 1, Sujeira no 2)

- Estado inicial: qualquer um dos estados acima (ex. at(left, yes, yes))
- Ações: left, right, suck
  - ações(at(left, yes, yes)) retorna (1) right, (2) suck

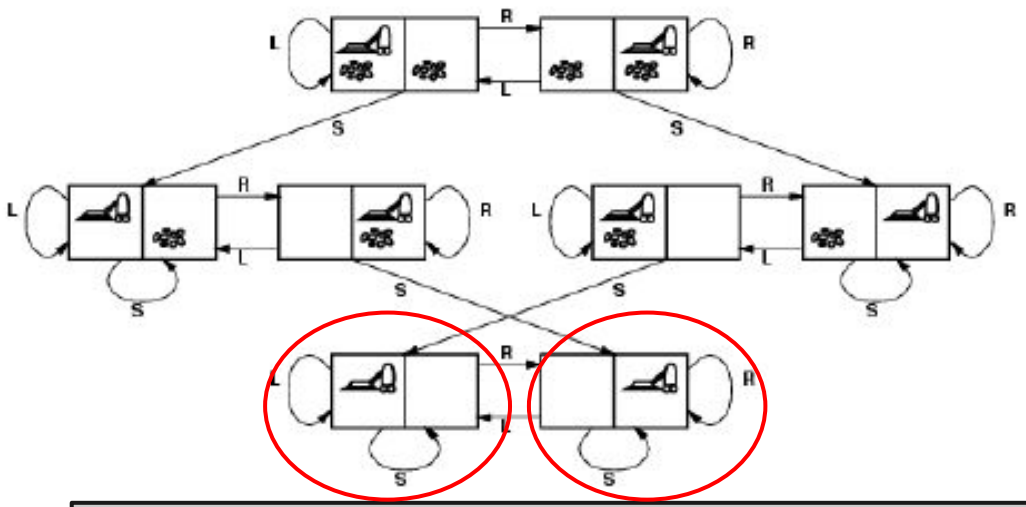
Exemplo: aspirador de pó



Estados: localização do aspirador e da sujeira
at(Aspirador, Sujeira no 1, Sujeira no 2)

- Estado inicial: qualquer um dos estados acima (ex. at(left, yes, yes))
- Ações: left, right, suck
  - actions(at(left, yes, yes)) retorna (1) right, (2) suck
- Modelo de transição
  - result(at(left, yes, yes), right) = at(right, yes, yes)
  - result(at(left, yes, yes), suck) = at(left, no, yes)

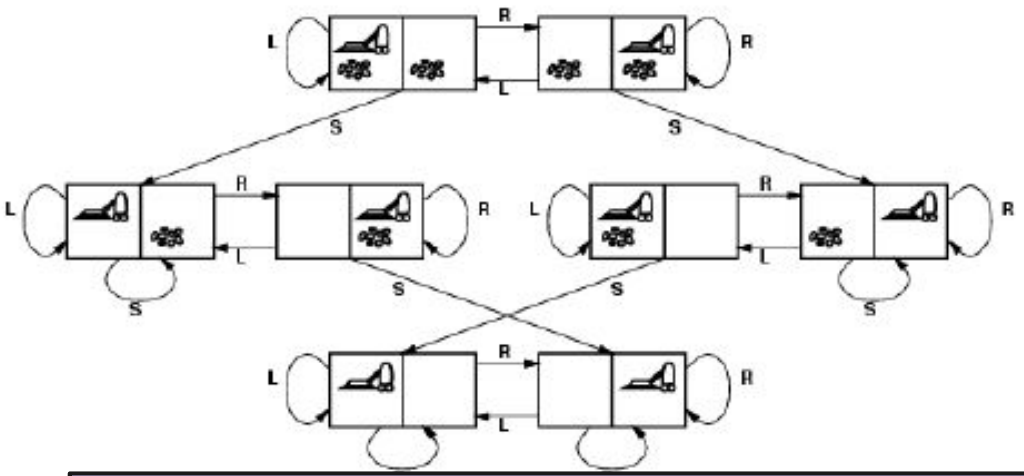
Exemplo: aspirador de pó



Estados: localização do aspirador e da sujeira
at(Aspirador, Sujeira no 1, Sujeira no 2)

- Estado inicial: qualquer um dos estados acima (ex. at(left, yes, yes))
- Ações: left, right, suck
  - actions(at(left, yes, yes)) retorna (1) right, (2) suck
- Modelo de transição
  - result(at(left, yes, yes), right) = at(right, yes, yes)
  - result(at(left, yes, yes), suck) = at(left, no, yes)
- Teste de Meta: Todos os quadrados estão limpos?
  - at(left, no, no) ou at(right,no,no)

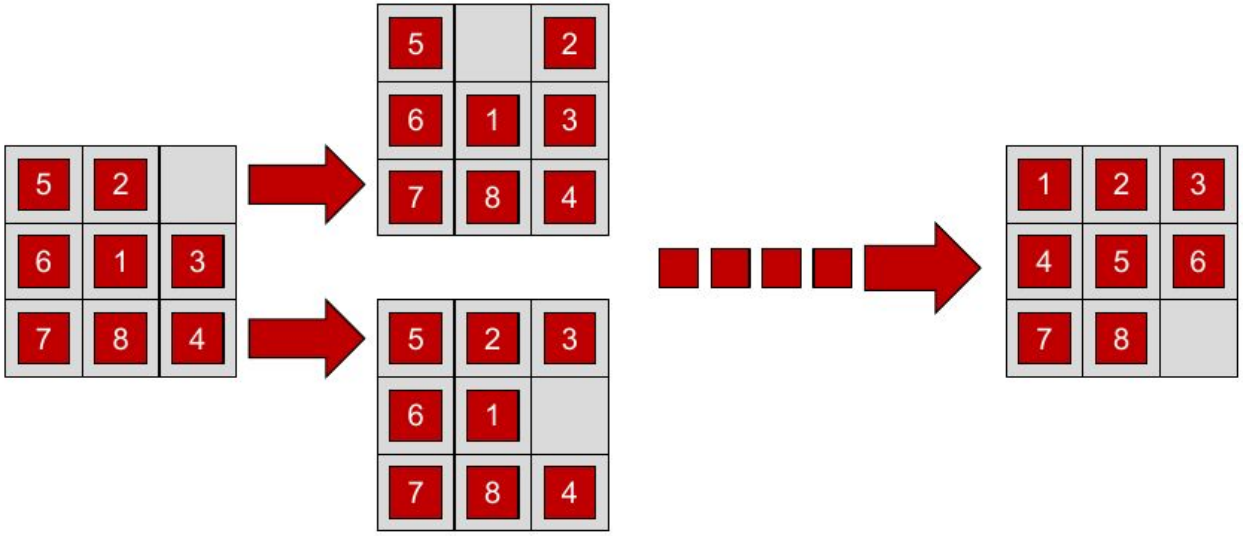
Exemplo: aspirador de pó



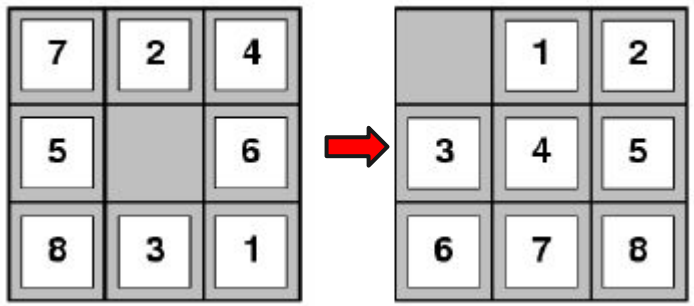
Estados: localização do aspirador e da sujeira
at(Aspirador, Sujeira no 1, Sujeira no 2)

- Estado inicial: qualquer um dos estados acima (ex. at(left, yes, yes))
- Ações: left, right, suck
  - actions(at(left, yes, yes)) retorna (1) right, (2) suck
- Modelo de transição
  - result(at(left, yes, yes), right) = at(right, yes, yes)
  - result(at(left, yes, yes), suck) = at(left, no, yes)
- Teste de Meta: Todos os quadrados estão limpos?
  - at(left, no, no) ou at(right,no,no)
- Custo do caminho: 1

Problema: quebra cabeça de 8



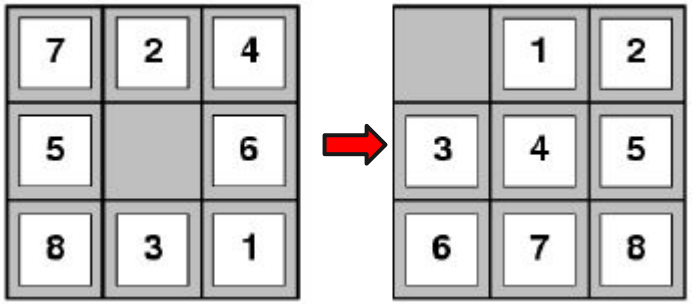
Exercício: quebra-cabeça de 8



Estado:

- Estado inicial:
- Ações:
- Modelo de transição:
- Teste de Meta:
- Custo do caminho:

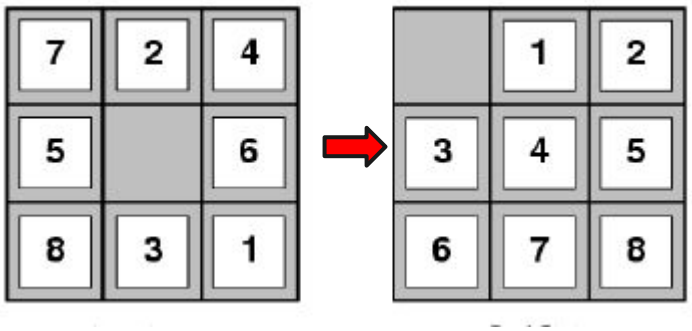
Exercício: quebra-cabeça de 8



Estado: matriz 3X3 ou vetor com 9 posições

- Estado inicial:
- Ações:
- Modelo de transição:
- Teste de Meta:
- Custo do caminho:

Exercício: quebra-cabeça de 8

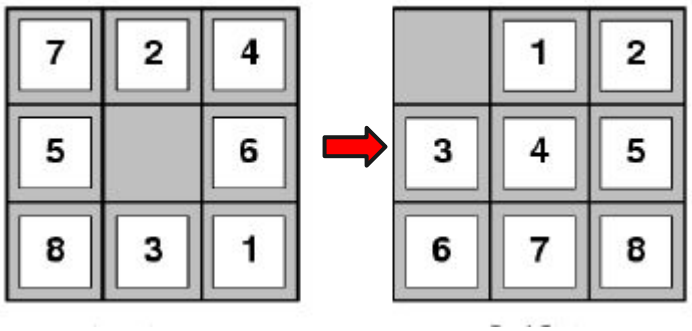


Estado: matriz 3X3

```
[
  [7,2,4]
  [5,b,6]
  [8,3,1]
        ]
```

- Estado inicial: poderia ser qualquer um. Para esse caso, é o estado acima
- Ações:
- Modelo de transição:
- Teste de Meta:
- Custo do caminho:

Exercício: quebra-cabeça de 8

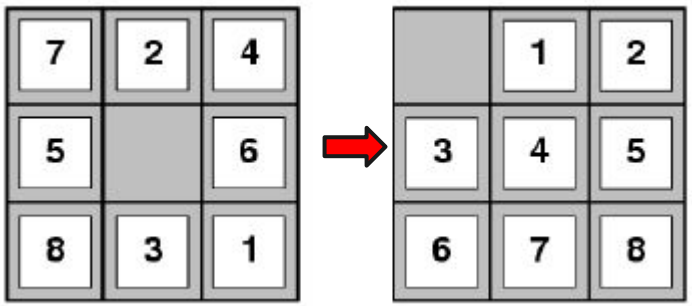


Estado: matriz 3X3

```
[
  [7,2,4]
  [5,b,6]
  [8,3,1]
        ]
```

- Estado inicial: qualquer um
- Ações: mover o vazio para left, right, up, down
  - actions([[7,2,4],[5,b,6],[8,3,1]]) = left, up, right, down
  - actions([[b,7,4],[5,2,6],[8,3,1]]) = right, down
- Modelo de transição:
- Teste de Meta:
- Custo do caminho:

Exercício: quebra-cabeça de 8

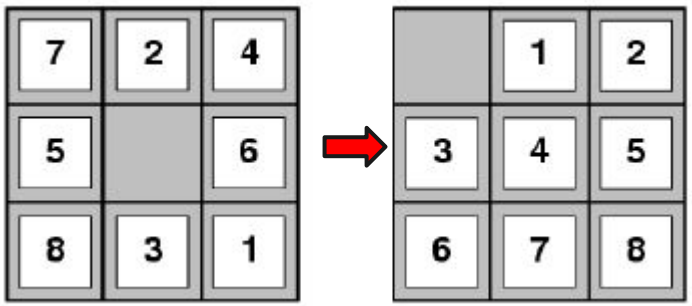


Estado: matriz 3X3

```
[
  [7,2,4]
  [5,b,6]
  [8,3,1]
        ]
```

- Estado inicial: [[7,2,4],[5,b,6],[8,3,1]]
- Ações: mover o vazio para left, right, up, down
  - actions([[7,2,4],[5,b,6],[8,3,1]]) = left, up, right, down
  - actions([[b,7,4],[5,2,6],[8,3,1]]) = right, down
- Modelo de transição:
  - result([[7,2,4],[5,b,6],[8,3,1]],left) = [[7,2,4],[b,5,6],[8,3,1]]
  - Note que é possível generalizar a função result: left (X,Y) = (X,Y-1)
- Teste de Meta:
- Custo do caminho:

Exercício: quebra-cabeça de 8



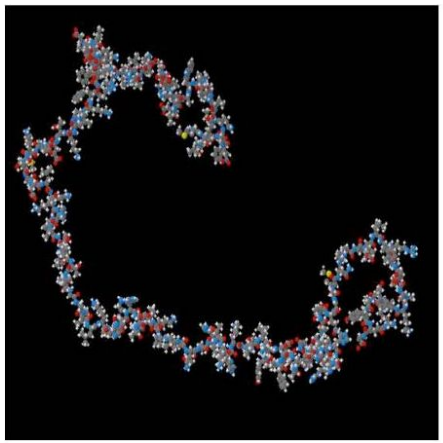
Estado: matriz 3X3

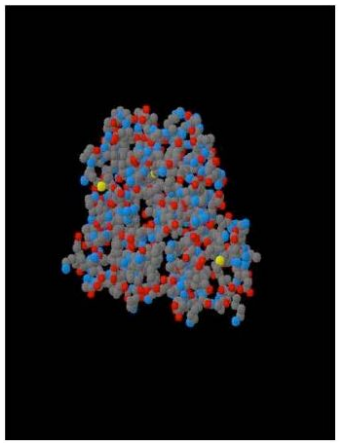
```
[
  [7,2,4]
  [5,b,6]
  [8,3,1]
        ]
```

- Estado inicial: [[7,2,4],[5,b,6],[8,3,1]]
- Ações: mover o vazio para left, right, up, down
  - actions([[7,2,4],[5,b,6],[8,3,1]]) = left, up, right, down
  - actions([[b,7,4],[5,2,6],[8,3,1]]) = right, down
- Modelo de transição:
  - result([[7,2,4],[5,b,6],[8,3,1]],left) = [[7,2,4],[b,5,6],[8,3,1]]
  - Note que é possível generalizar a função result: left (X,Y) = (X,Y-1)
- Teste de Meta: está em [[b,1,2], [3,4,5], [6,7,8]] ? (também pode ser genérico e verificar se está ordenado)
- Custo do caminho: 1

Problemas do mundo real

- TSP: cada cidade é visitada no máximo uma vez
- Projeto de circuito VLSI
- Navegação robótica
- Sequência de montagem automática de objetos
- Projeto de proteínas





Problemas do mundo real

- Descoberta de rota (websites, GPS, roteamento de streams de video, rota de avião, etc)
  - Estados:
  - Estado inicial:
  - Ações:
  - Modelo de transição
  - Teste da meta:
  - Custo do caminho:

Descoberta de rota

- Viagem de avião
  - Estados: **localização, hora, histórico**
  - Estado inicial: **de acordo com a consulta do usuário**
  - Ações: **pegar um voo na localização corrente, em qualquer classe, depois da hora corrente**
  - Modelo de transição: **estado resultante após pegar um voo**
  - Teste da meta: **chegou no destino?**
  - Custo do caminho: **valor da passagem, tempo (espera, voo, imigração, etc), dia, avião, etc**

Navegação robótica

- Tarefa: robô deve transportar objeto de um lado para o outro
- Estado: posição, orientação, ângulos das articulações, se está segurando algum objeto (estado inicial: valor corrente dessas variáveis)
- Ações: flexionar/rotacionar articulações, ativar rodas
- Meta: objeto está no local desejado?
- Custo: energia/tempo consumidos, penalidade se derrubar objeto

Tradução do inglês para francês

- Task: traduzir do inglês para francês

Modelo simples

- Estado: palavras em inglês traduzidas até o momento *E* + palavras ainda não traduzidas *E'* (Estado inicial: E = vazio, E' = frase)
- Ações/transição: escolher palavra em inglês *e in E', e not in E,* e palavra em francês *f*
- Custo: *-fidelidade(e, f)*
- Teste de meta: se a sentença inteira em inglês foi traduzida

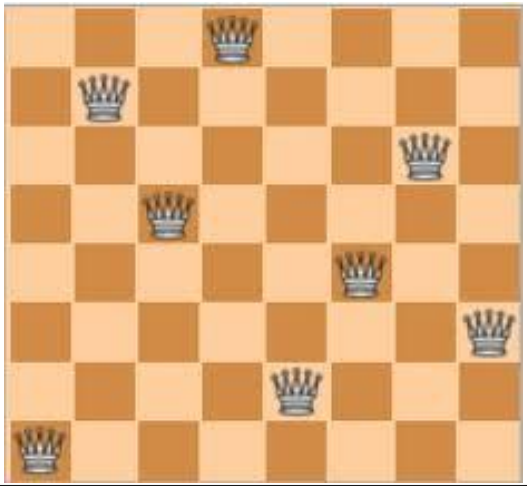
Tradução do inglês para francês

- Task: traduzir do inglês para francês

Modelo melhorado

- incluir função de fluência no custo, para evitar traduções sem sentido
- Custo: *-fidelidade(e, f) - fluency(f',f)*

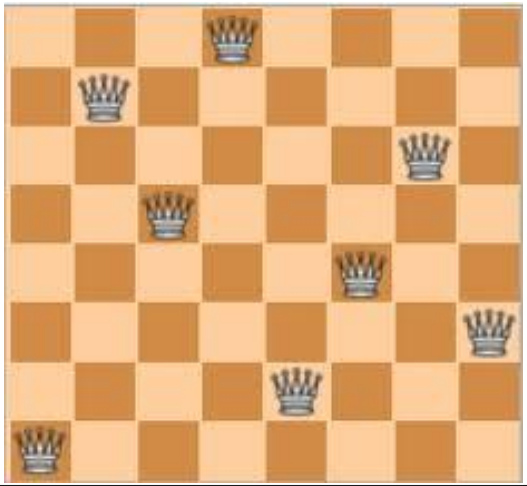
Problema das 8 rainhas (incremental)



Inserir 8 rainhas no tabuleiro, sem que uma ataque a outra
Estado: vetor com posições das rainhas; matriz 8X8

- Estado inicial: Tabuleiro vazio
- Ações: inserir uma rainha no tabuleiro
- Modelo de transição: tabuleiro com a rainha inserida na posição específica
- Teste de Meta: 8 rainhas no tabuleiro, sem que nenhuma seja atacada
- Custo do caminho: 1
- ~1,8 * 10^{14} possíveis sequências

Problema das 8 rainhas (completo)

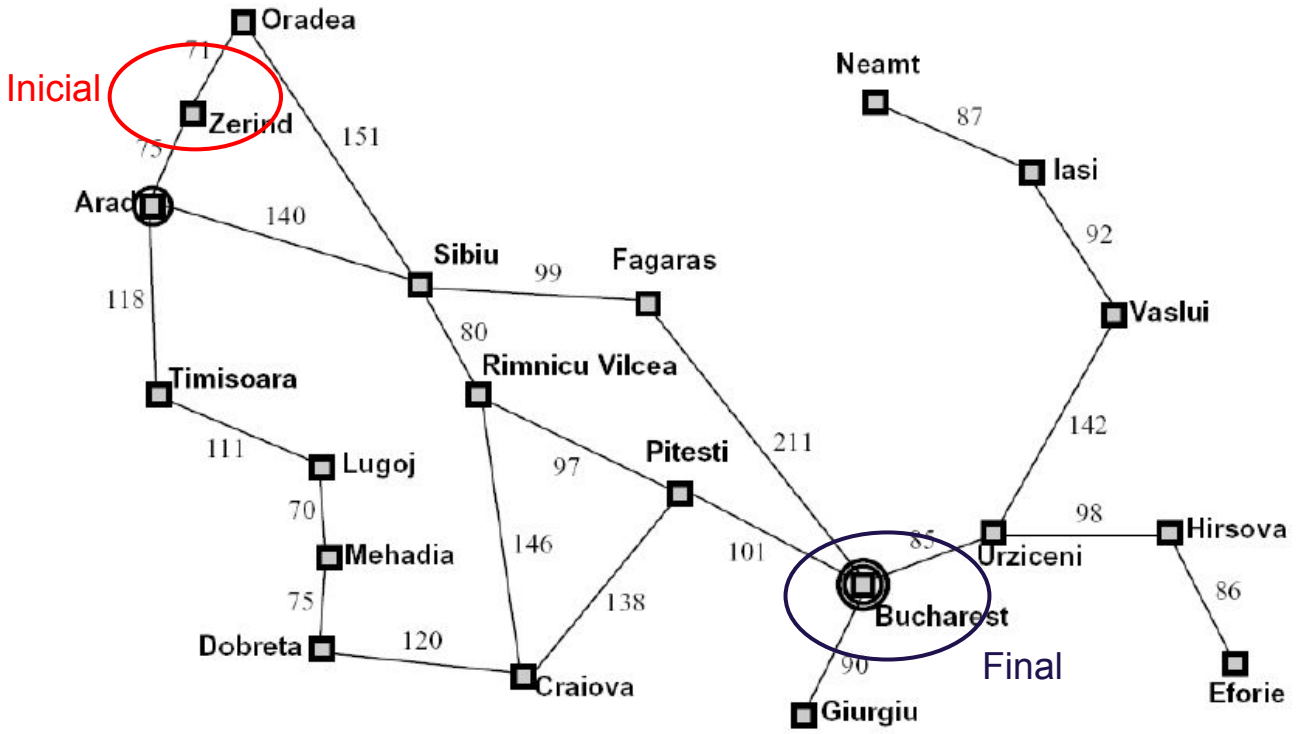


Inserir 8 rainhas no tabuleiro, sem que uma ataque a outra
Estado: vetor com posições das rainhas; matriz 8X8

- Estado inicial: Tabuleiro com 0 a 8 rainhas, uma por coluna
- Ações: inserir uma rainha em qualquer quadrado, sem que nenhuma ataque outra
- Modelo de transição: tabuleiro com a posição alterada de uma rainha
- Teste de Meta: 8 rainhas no tabuleiro, sem que nenhuma seja atacada
- Custo do caminho: 1
- 2057 sequências a serem investigadas

Exercício

1. Descreva o problema abaixo



Exercício

2. Descreva os seguintes problemas (use uma formulação que seja precisa o suficiente para ser implementada).

a. Você tem que colorir um mapa plano usando somente 4 cores, de tal forma que duas regiões adjacentes não tenham a mesma cor.

b. Um macaco de meio metro de altura está em uma jaula onde algumas bananas estão suspensas à três metros e meio do chão. Ele quer pegar as bananas. A jaula contém dois caixotes de um metro e meio cada que podem ser movidos e sobrepostos.

c. Você tem três jarros, medindo 12 litros, 8 litros e 3 litros e uma fonte de água. Você pode encher ou esvaziar os jarros de um para o outro ou no chão. Você quer medir exatamente um litro.

Exercícios

3 - Descrever formalmente os problemas dos Missionários e Canibais e da Torre de Hanói.