Escola Politécnica da USP - Depto. de Enga. Mecatrônica

PMR-3510 Inteligência Artificial
Aula 3 - Agentes Inteligentes e Resolução de Problemas

Prof. José Reinaldo Silva

reinaldo@usp.br

d-l@b

Na aula passada discutimos sobre os diversos tipos de agentes inteligentes:

Diferentes tipos de agentes:

Reflexivos (simple réflex agents)

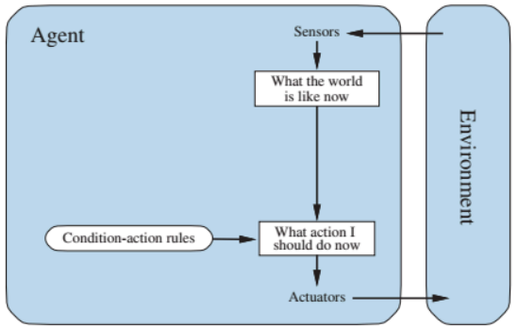


Figure 2.9 Schematic diagram of a simple reflex agent. We use rectangles to denote the current internal state of the agent's decision process, and ovals to represent the background information used in the process.

Agentes baseados em modelos (Model-based agents)

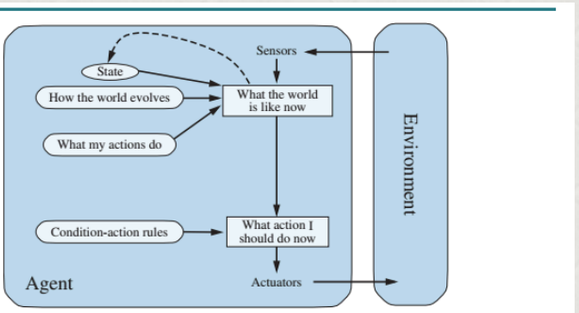


Figure 2.11 A model-based reflex agent.

Agentes baseados em objetivos (Goal-based agents)

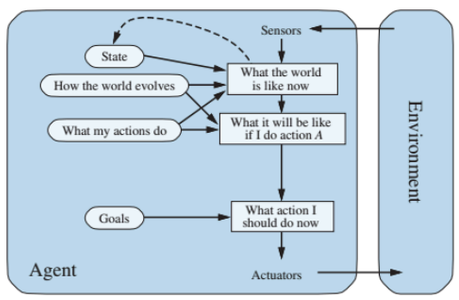


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Agentes baseados em serviço (utility-based agents)

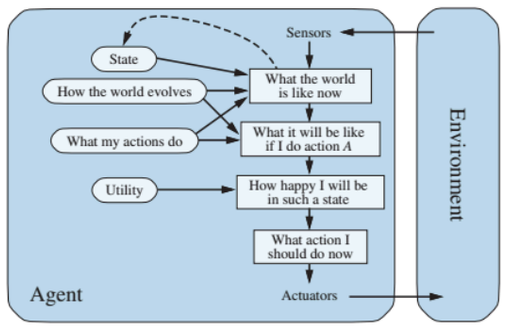


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Agentes que aprendem (Learning agents)

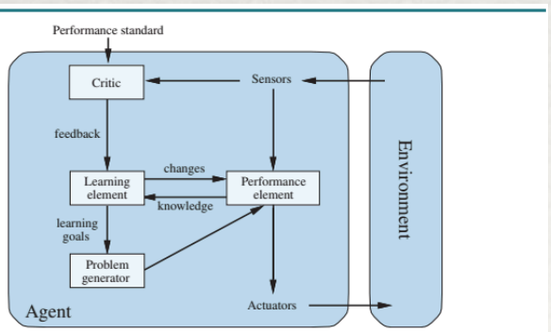
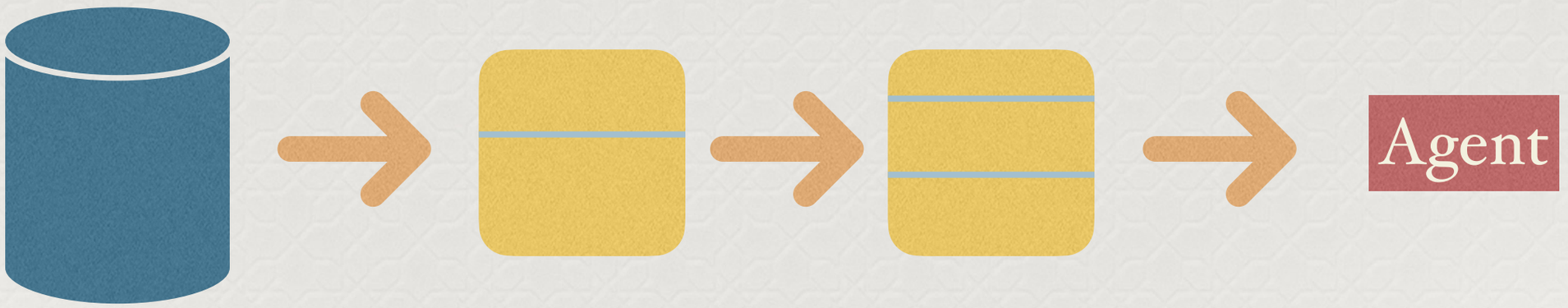


Figure 2.15 A general learning agent. The "performance element" box represents what we have previously considered to be the whole agent program. Now, the "learning element" box gets to modify that program to improve its performance.

A evolução desde dados estáticos até os agentes inteligentes

Portanto o objetivo é representar conhecimento e processos racionais de forma computável. O ponto de partida será a forma mais simples de conhecimento: dados – especialmente aqueles considerados confiáveis e verdadeiros. Mesmo a programação procedural tem dedicado muita atenção à representação de dados, que podem ser usados de forma estática ou interagindo com processos dinâmicos. A linha de evolução nesta área pode ser sintetizada como:



SWISH File Edit Examples Help

5 users online

Search

Mundo de blocos - Rework | Prolog-introdução

fazer nenhum "exercício" de porte médio ou grande. Fique tranquilo!

Lidando com listas

Lista é uma estrutura de dados básica muito usada em computação e também em programas de Inteligência Artificial. A primeira linguagem ligada à IA foi o LISP que lida basicamente com listas. Uma lista é uma estrutura de dados composta de dois elementos: a cabeça da lista, (head) e o corpo ou o restante da lista (tail).

Create a — Program | Query | Markdown | HTML — cell here

head - . - tail

Portanto, se quisermos declarar uma lista de pessoas teríamos algo como...

[maria, claudio, jose, giovana, carlos],

se prcisarmos instanciar variáveis em [X,Y] terímaos

X= Maria

Y=[claudio, jose, giovana, carlos]

Portanto a cabeça da lista é um elemento constituinte enquanto o corpo é uma lista.
Está definda a lista vazia ou [].

Como exercício, vamos contar o número de elementos de uma lista declarando o predicado list_count/2 que dada uma lista retorna o número de elementos. Assim, list_count([a,b,c,d,e],X) deve retornar X=5. Como seria este predicado?

```
1 list_count([],0).
2 list_count([_|X],Y):- list_count(X,Z), Y is Z+1.
```

```
?- list_count([x,y,w,z,m,f],X).
```

Você pode usar no querry o comando "trace" para ver passo a passo o que está acontecendo, e como é feita a instanciação das variáveis e chamada dos procedimentos. Primeiro, será testado o primeiro predicado que checa se estamos contando os elementos de uma lista vazia (o ponto de parada). Se for esse o caso não há necessidade de ir adiante, e simplesmente a variável de retorno (seja lá como seja designada) recebe o valor 0 (zero). Se o primeiro argumento não for uma lista vazia, então chama-se recursivamente o mesmo predicado, agora com o corpo da lista (vamos contar quantos elementos tem o resto da lista). Quando retornar, basta incrementar de um o resultado para obter o

?-

SWISH File Edit Examples Help

5 users online

Search

Mundo de blocos - Rework | Prolog-introdução

?- Your query goes here ...

A resposta aparece logo em seguida instanciando X=marcos. Se não apareceu é porque você digitou algo errado. Vamos agora dar um passo adiante e inserir uma regra ou procedimento. Uma regra pode relaciar de forma lógica diferentes queries associados à instanciação de fatos. Por exemplo, queremos definir o que significa irmao(X,Y), isto é, preparar o nosso programa para checar quem é irmão de quem, sem ter estes fatos diretamente. Vamos portanto admitir que um dado individuo é irmão de alguém se é do sexo masculino e se tem o mesmo pai ou a mesma mãe. Veja que com a base que temos não é possível saber o sexo de um dado elemento (o sistema só "sabe" o que estiver explicitamente definido na base). Portanto vamos ampliar o nosso probrama para inserir esta informação.

Create a Program Query Markdown HTML cell here

```
1 masculino(joao).
2 masculino(clovis).
3 masculino(marcos).
4 masculino(rogerio).
5 masculino(felipe).
6 feminino(marta).
7 feminino(cintia).
8 feminino(telma).
9 feminino(maria).
```

Agora podemos definir a regra "irmao",

```
1 irmao(X,Y):-masculino(X),pai(Z,X),pai(Z,Y).
2 irmao(X,Y):-masculino(Y),mae(Z,X),mae(Z,Y).
```

Uma regra é definida pelo seu functor e variáveis (irmao(X,Y)) seguido pela descrição (separado :-) que aqui diz que X deve ser do sexo masculino (masculino(X)) e ter um pai Z (pai(Z,X) que é o mesmo pai de Y (pai(Z,Y)). Similarmente a segunda regra diz o mesmo para o caso de X e Y terem a mesma mãe. Podemos agora fazer um querie geral perguntando quem é irmão de quem, e usando apenas variáveis: digite na janela de querie abaixo ?-irmao(X,Y). e e clique na set azul à direita apareceu X-clovie e Y=marta? razoável e intuitivo! Agora note que existe também a opção de clicar em "Next" dizendo ao programa para buscar novas alternativas. Faça isso...

?- Your query goes here ...

?-

A proposta do curso

PMR3510

Discussão teórica e conceitual

Aulas

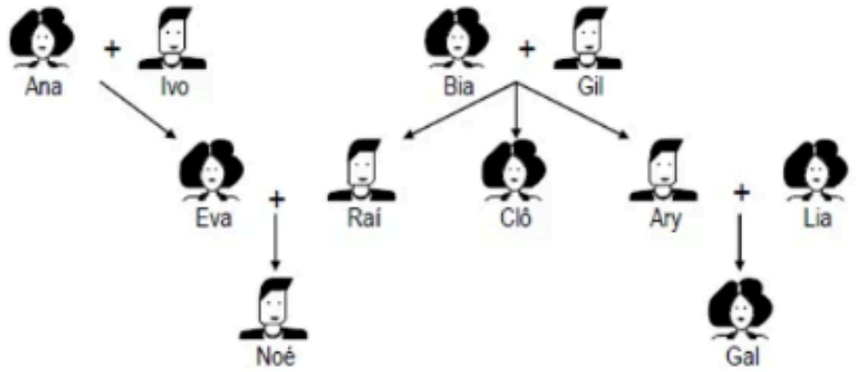
Exercícios

Em equipe

Tutorial Prolog

Online

Considere a árvore genealógica a seguir:



- Usando fatos, defina as relações pai e mãe. Em seguida, consulte o sistema para ver se suas definições estão corretas.

Exercícios

https://pt.slideshare.net/JanaideNogueira/aula-prolog-02-ia-60882534

SWISH File Edit Examples Help

5 users online

Search

Mundo de blocos - Rework | Prolog-introdução

?- Your query goes here ...

A resposta aparece logo em seguida instanciando X=marcos. Se não apareceu é porque você digitou algo errado. Vamos agora dar um passo adiante e inserir uma regra ou procedimento. Uma regra pode relaciar de forma lógica diferentes queries associados à instanciação de fatos. Por exemplo, queremos definir o que significa irmao(X,Y), isto é, preparar o nosso programa para checar quem é irmão de quem, sem ter estes fatos diretamente. Vamos portanto admitir que um dado individuo é irmão de alguém se é do sexo masculino e se tem o mesmo pai ou a mesma mãe. Veja que com a base que temos não é possível saber o sexo de um dado elemento (o sistema só "sabe" o que estiver explicitamente definido na base). Portanto vamos ampliar o nosso probrama para inserir esta informação.

Create a Program Query Markdown HTML cell here

```
1 masculino(joao).
2 masculino(clovis).
3 masculino(marcos).
4 masculino(rogerio).
5 masculino(felipe).
6 feminino(marta).
7 feminino(cintia).
8 feminino(telma).
9 feminino(maria).
```

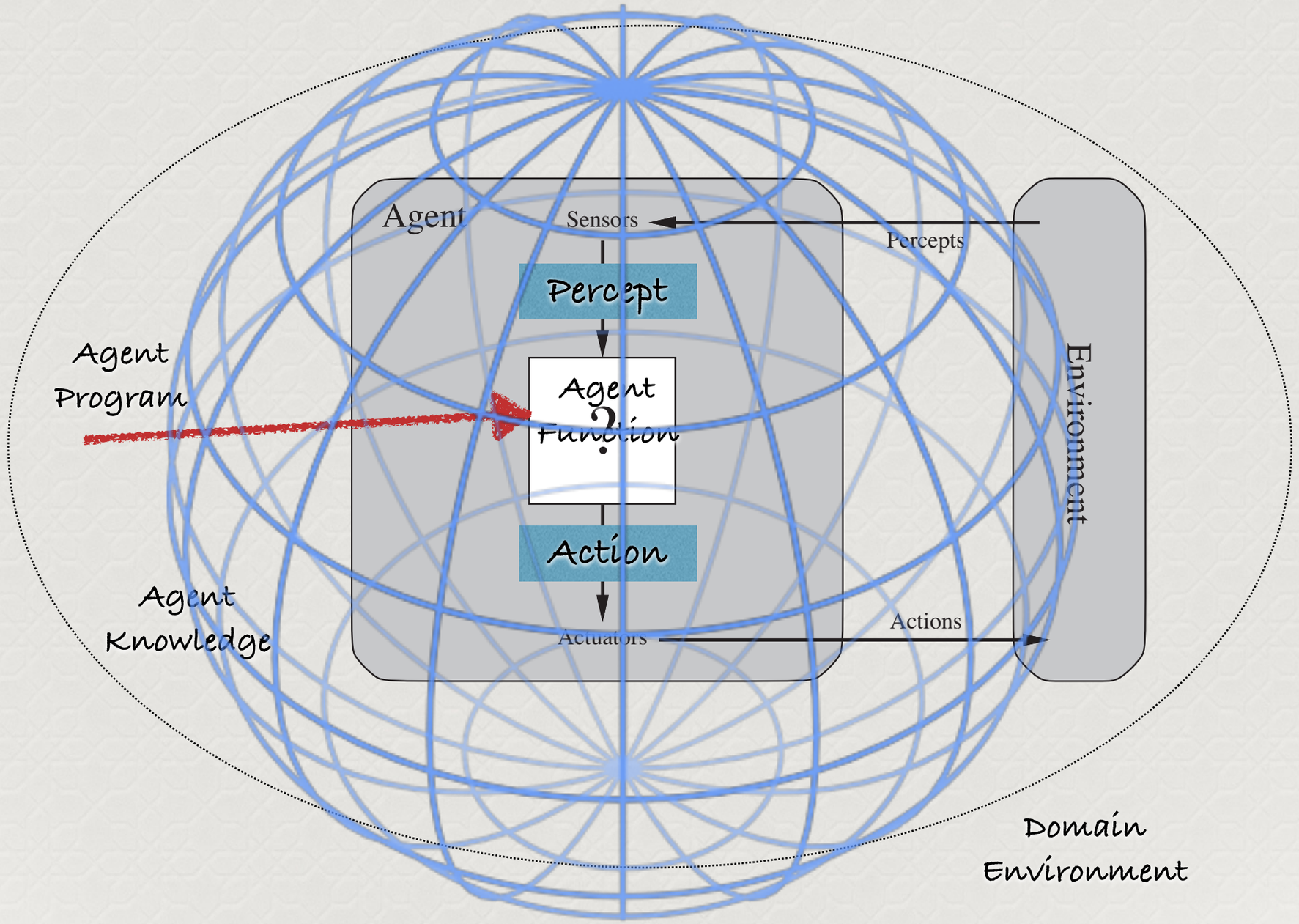
Agora podemos definir a regra "irmao",

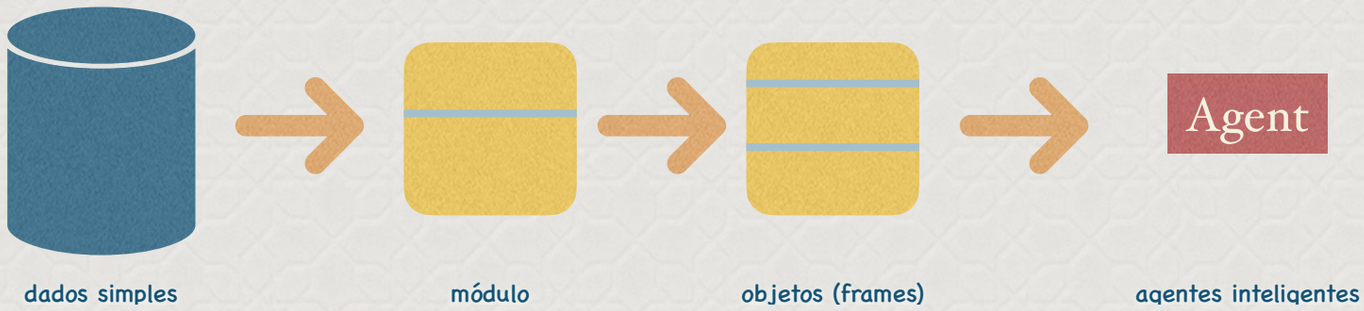
```
1 irmao(X,Y):-masculino(X),pai(Z,X),pai(Z,Y).
2 irmao(X,Y):-masculino(Y),mae(Z,X),mae(Z,Y).
```

Uma regra é definida pelo seu functor e variáveis (irmao(X,Y)) seguido pela descrição (separado :-) que aqui diz que X deve ser do sexo masculino (masculino(X)) e ter um pai Z (pai(Z,X) que é o mesmo pai de Y (pai(Z,Y)). Similarmente a segunda regra diz o mesmo para o caso de X e Y terem a mesma mãe. Podemos agora fazer um querie geral perguntando quem é irmão de quem, e usando apenas variáveis: digite na janela de querie abaixo ?-irmao(X,Y). e e clique na set azul à direita apareceu X-clovie e Y=marta? razoável e intuitivo! Agora note que existe também a opção de clicar em "Next" dizendo ao programa para buscar novas alternativas. Faça isso...

?- Your query goes here ...

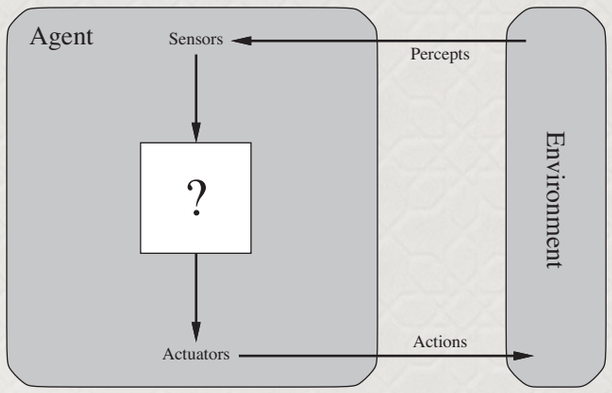
?-

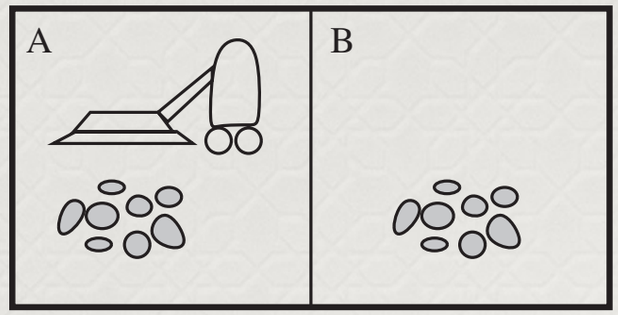

Agent
Sensors
Percepts
Percept
Agent Program
Agent Function
?
Action
Actuators
Actions
Environment
Agent Knowledge
Domain Environment



Evoluindo dos dados até os agentes inteligentes

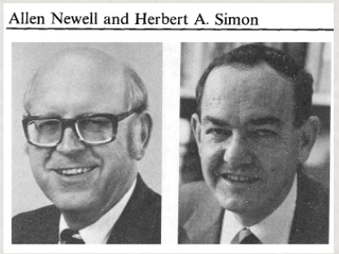
Um agente inteligente é capaz de perceber o que acontece no seu ambiente de trabalho, ou environment, e definir que ação deve tomar, atuando sobre este mesmo ambiente. A ação será dita "inteligente" se for o resultado de inferência. Um exemplo simples seria um aspirador (inteligente) que detecta e localiza o pó a ser removido, traça um caminho (ótimo) até ele, evitando obstáculos, e aciona o aspirador.





Resolvendo problemas com IA!

Após a Conferência do Dartmouth College, a orientação era direcionar a pesquisa na área de IA para processos formais para transcender o processo direto de programação procedural.



Logic Theorist

General Problem Solving (GPS)

Knowledge-based Systems (KBS)

Learning

Machine Learning

Agentes inteligentes que resolvem problemas

When the correct action to take is not immediately obvious, an agent may need to *plan ahead*: to consider a *sequence* of actions that form a path to a goal state. Such an agent is called a **problem-solving agent**, and the computational process it undertakes is called **search**.

ITS World Congress 2022 | ITS 2023 Annual Meeting | ITSA Events

LOS ANGELES
ITS WORLD CONGRESS 2022

September 18-22, 2022
LA Convention Center

REGISTER TODAY

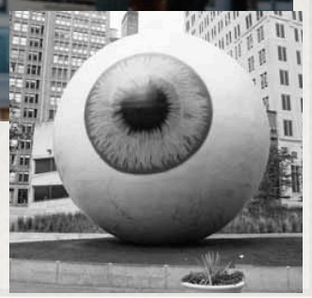
SHOW INFO | EXPERIENCES | PROGRAM | EXHIBIT & SPONSOR | MEDIA | HOTEL & TRAVEL

ATTEND ITS WORLD CONGRESS

Be part of ITS World Congress 2022, the premier event in intelligent transportation, taking place September 18-22 (Conference: September 18 – 22 | Exhibit Hall: September 19 – 22).

REGISTER TODAY

... de olho no mercado!



Intelligent Transportation Systems

Developments in technologies present Road Administrations around the world with the opportunity to transform the way that they manage and operate their highway networks.

Intelligent Transportation Systems (ITS) is a combination of leading-edge information and communication technologies used in transportation and traffic management systems to improve the safety, efficiency, and sustainability of transportation networks, to reduce traffic congestion and to enhance drivers' experiences. The possibilities are endless.

Intelligent Transportation Systems

Overview | Technologies | Solutions | Featured Products | Success Stories | Selected Videos | Links | Resources

Railway Solution & Roadway Solutions

Railway

- Railway Safety Solution
- Intelligent In-Train Solution
- Wayside Solution
- Railway Station Solution

https://www.advantech.com.br/solutions/intelligent-transportation-systems

United States Department of Transportation

About DOT | Briefing Room | Our Activities

About OST-R | Programs | Library

Office of the Assistant Secretary for Research and Technology

Intelligent Transportation Systems Joint Program Office

ENHANCED BY Google

About | Research | ITS Deployment | Communications | Technology Transfer | Resources | Contact Us

HOT TOPICS

- CARMA
- Connected Vehicle Deployer Resources
- Connected Vehicle Pilots
- ITS Strategic Plan 2020-2025
- ITS Training
- Smart Communities
- The Safety Band

Get Updates on JPO News and Events:

Sign Up Now

RESEARCH AREAS

- Accelerating ITS Deployment
- Automation
- Complete Trip - ITS4US
- ITS Cybersecurity Research
- Data Access and Exchanges
- Emerging and Enabling Technologies

Vamos focar em um problema simples de roteamento em cidades.

Planning ahead...

Objetivo? (Goal?)

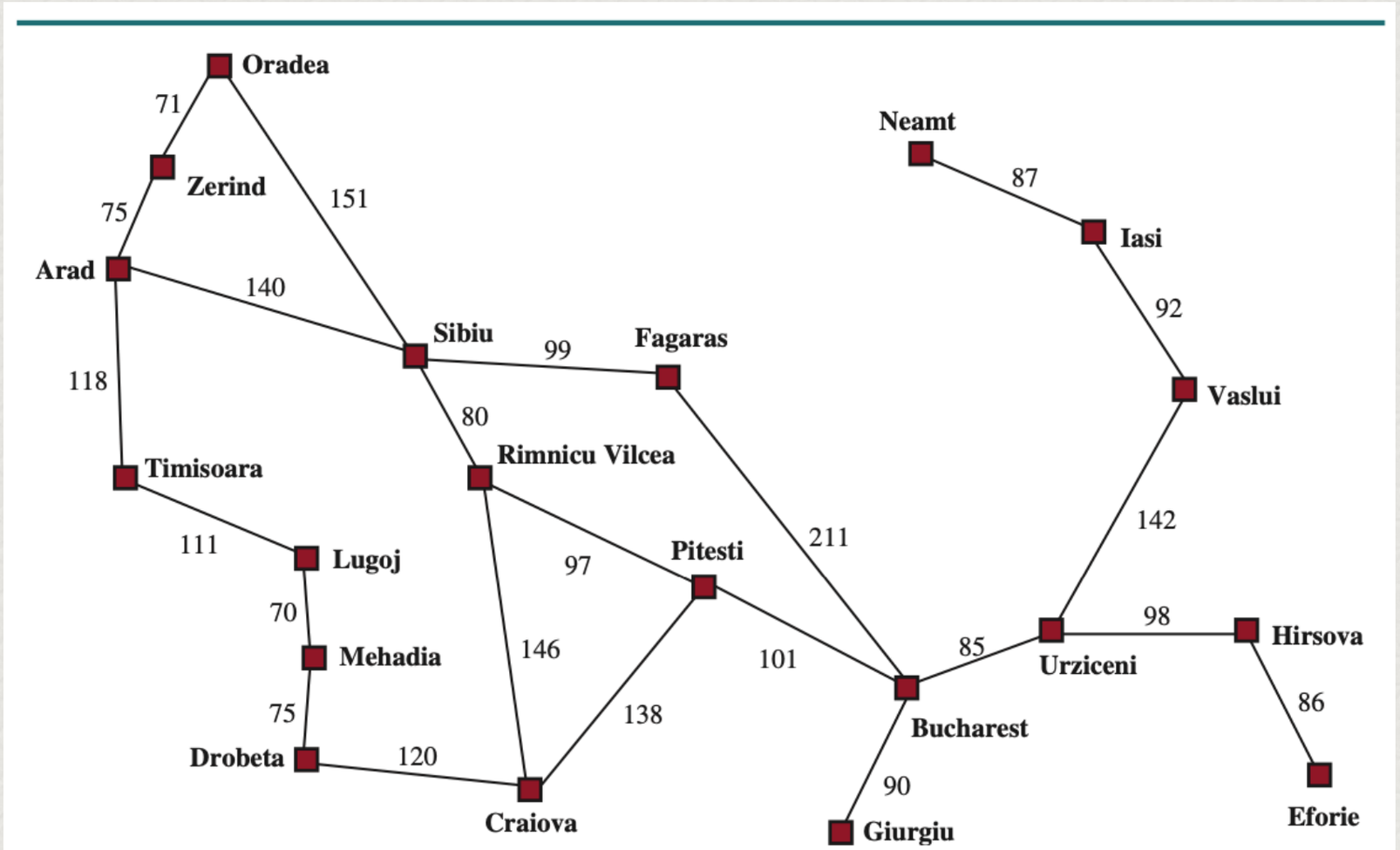


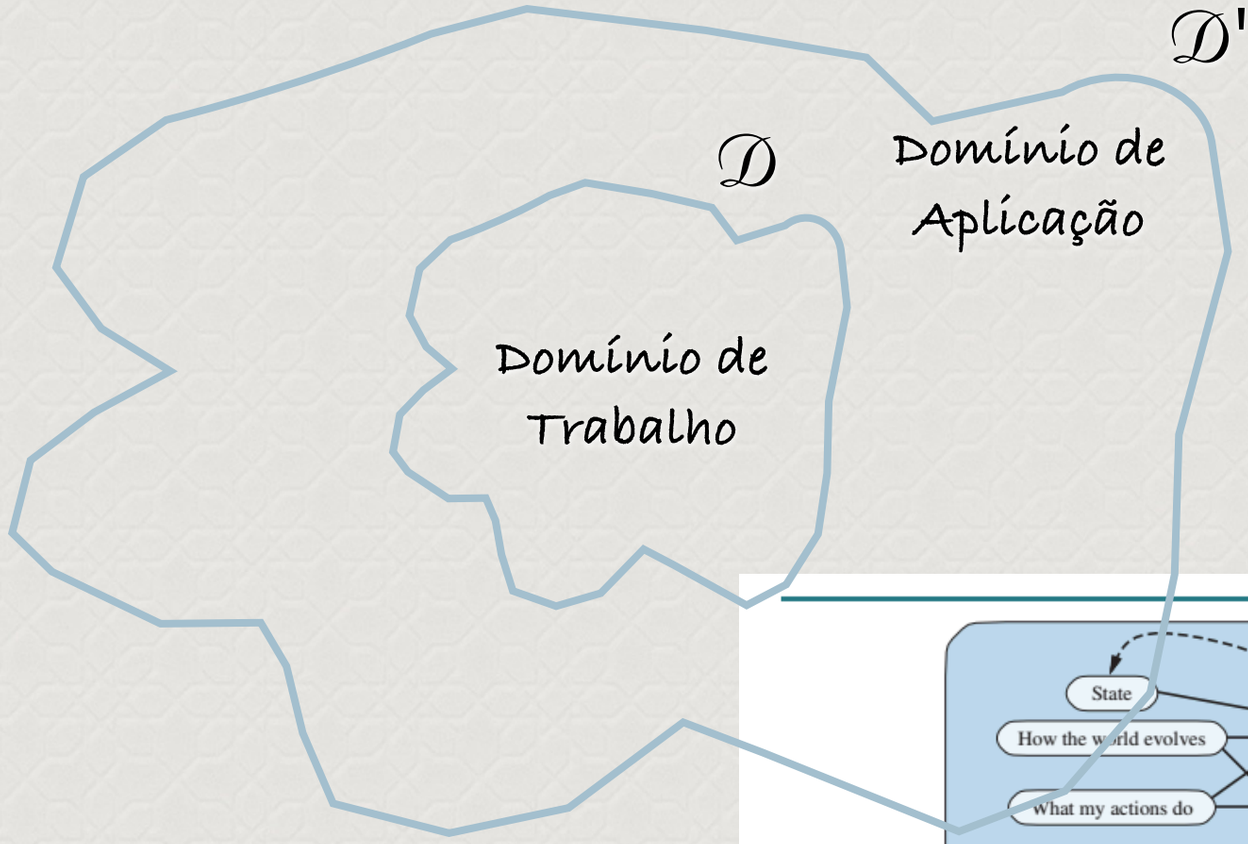
Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Condição inicial

Formulação do Problema

- **Problem formulation**: The agent devises a description of the states and actions necessary to reach the goal—an abstract model of the relevant part of the world. For our agent, one good model is to consider the actions of traveling from one city to an adjacent city, and therefore the only fact about the state of the world that will change due to an action is the current city.
- **Search**: Before taking any action in the real world, the agent simulates sequences of actions in its model, searching until it finds a sequence of actions that reaches the goal. Such a sequence is called a **solution**. The agent might have to simulate multiple sequences that do not reach the goal, but eventually it will find a solution (such as going from Arad to Sibiu to Fagaras to Bucharest), or it will find that no solution is possible.
- **Execution**: The agent can now execute the actions in the solution, one at a time.

Condição inicial; objetivo; ações possíveis (admissíveis); estados do problema.



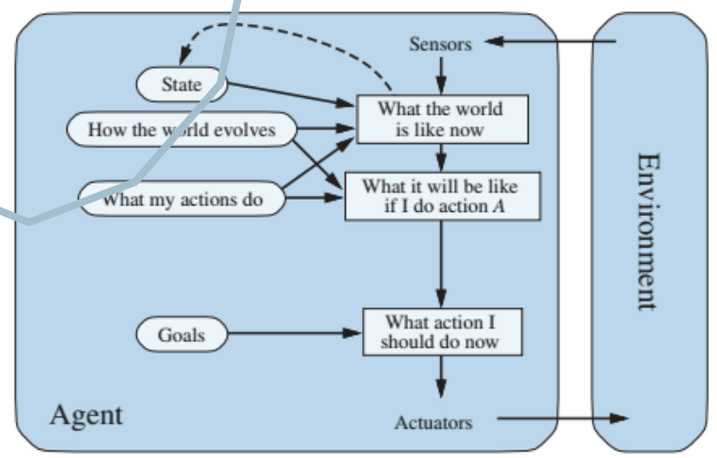
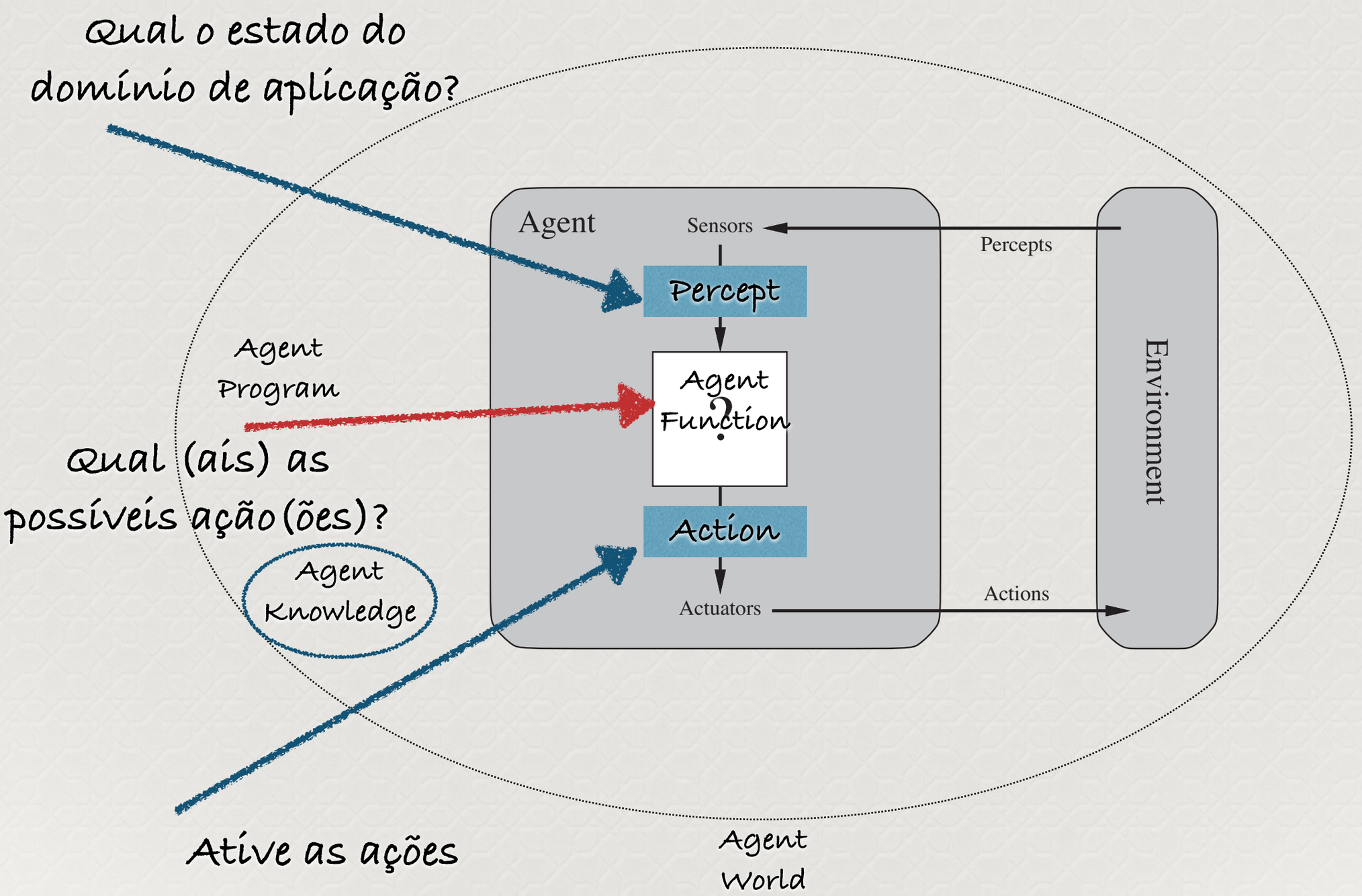


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.


Qual o estado do domínio de aplicação?
Agent
Sensors
Percepts
Percept
Agent Program
Agent Function
Environment
Qual (ais) as possíveis ação(ões)?
Action
Agent Knowledge
Actuators
Actions
Atíve as ações
Agent World

Genericamente vamos assumir a hipótese de que "resolver um problema" significa: levando em conta o domínio de aplicação e o domínio de trabalho, achar uma sequência de ações que leve da condição (estado) inicial ao objetivo (estado final), a situação (estado) em que "o problema é considerado resolvido".

Um procedimento para resolução de problemas deve começar pela identificação do espaço de estados do problema. O estado inicial é identificado neste espaço. Por exemplo, queremos uma rota saindo de Arad.

No caso de agentes baseados em objetivos, temos que identificar qual é este objetivo. Por exemplo, o objetivo pode ser "ir para Bucharest".

Agentes baseados em objetivos (Goal-based agents)

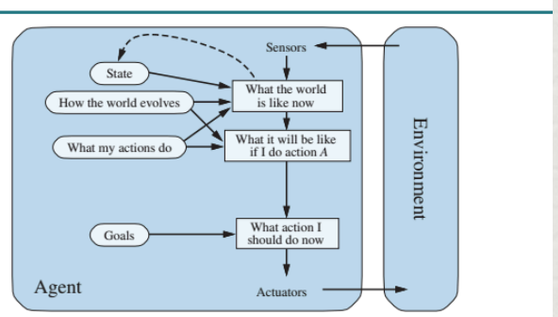


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Quais a ação possível estando em Arad?

- Ir para Sibiu
- Ir para Timisoara
- Ir para Zerind

Que ação escolher para "ir para o objetivo (estado) final"?

- Para avaliar isso é preciso ter uma referência, por exemplo uma função custo que pode ser associada à distância que cada um dos pontos (estados) resultantes da ação está do objetivo (estado) final.

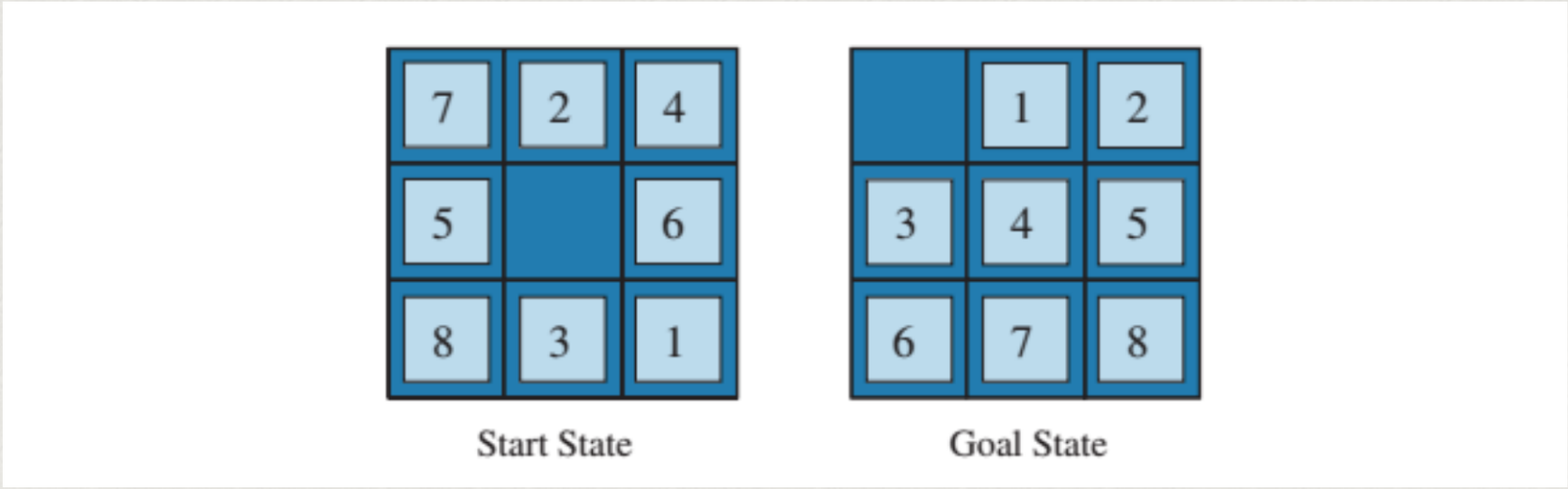
Em uma primeira aproximação (não otimizada), resolver um problema, pode significar apenas achar um caminho entre o estado inicial e o estado final, o que pode ser implementado por um algoritmo de busca.

Problemas-modelo

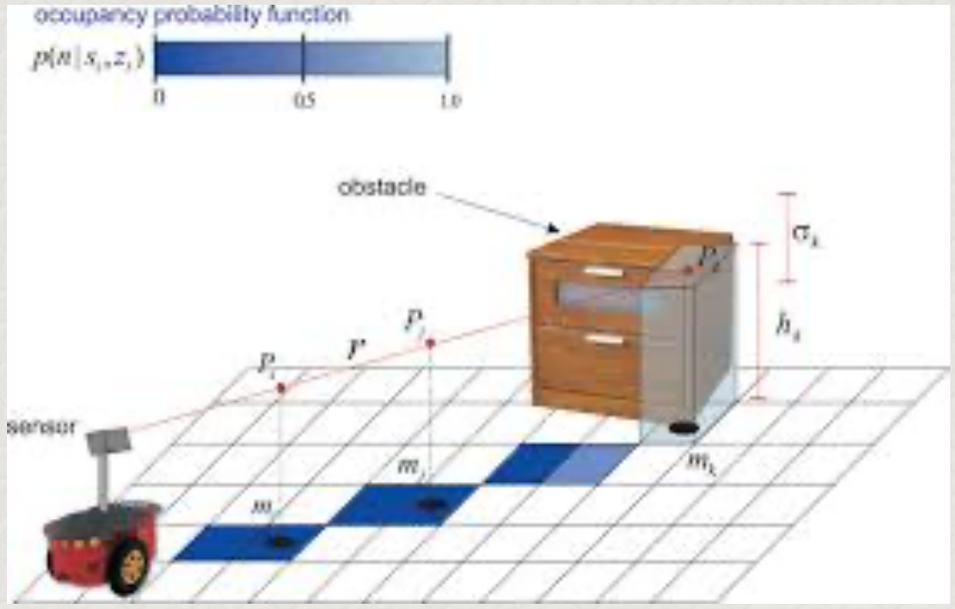
Uma maneira de facilitar a programação dos "Agent Functions" é formalizar soluções gerais que podem ser posteriormente adaptadas a situações reais, geralmente aplicada a problemas-modelo. O STRIPS (que vamos tratar em detalhe mais adiante, generaliza os problemas de planejamento. Outro exemplo é o chamado "grid problem", que pode ser adaptado para determinar a trajetórias de robôs móveis.

Modelo do "Grid problem" em games

Uma adaptação simples do "grid problem" é o jogo de "tiles", mostrado abaixo, que combina a estratégia orientada a objetivos com os movimentos físicos no "grid".



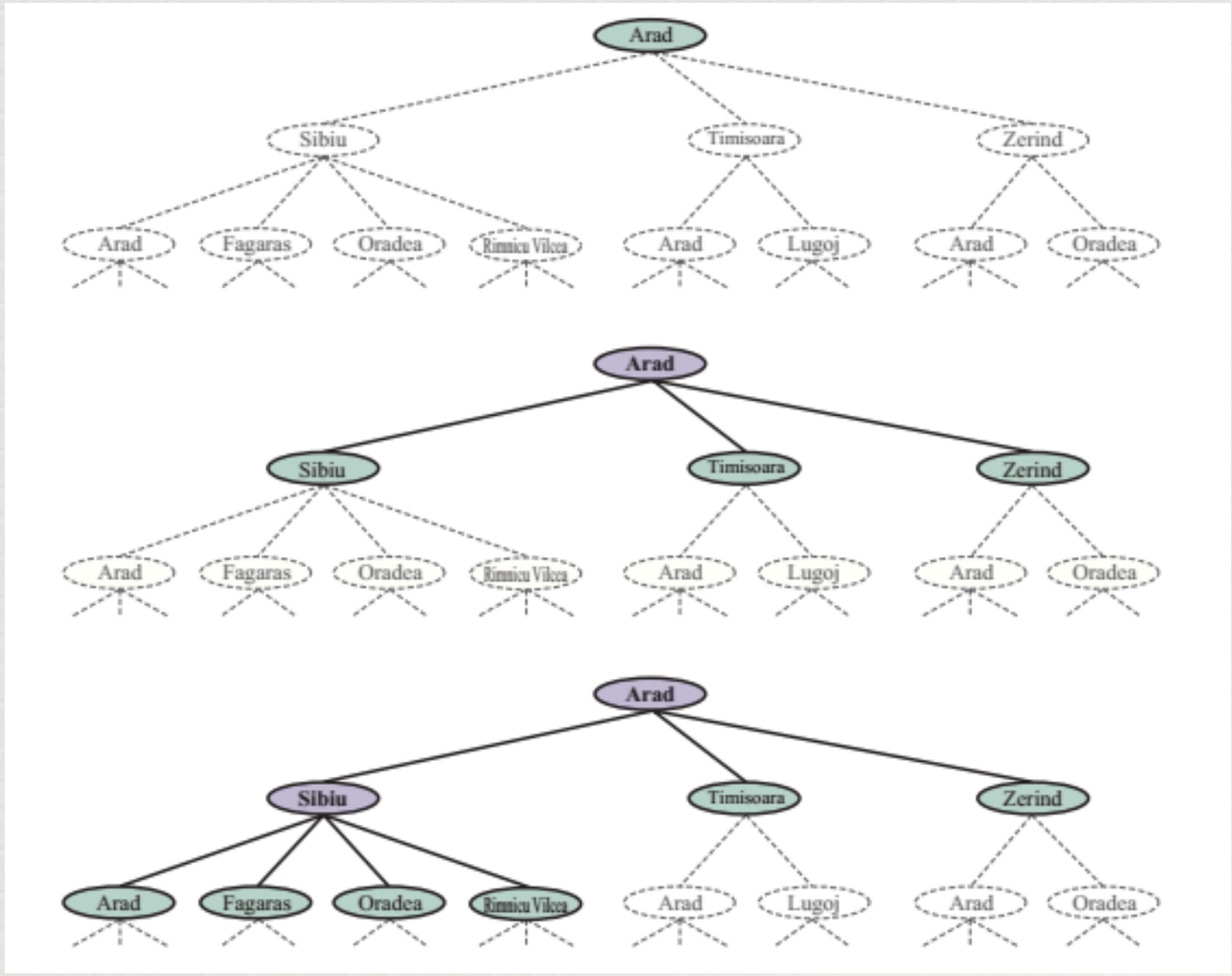
Um algoritmo conhecido como "Occupancy Grid" foi proposto em 1985 para orientar o movimento de robôs móveis. Este algoritmo foi proposto em Carnegie Mellon por Hans Peter Moravec e Alberto Elfes.



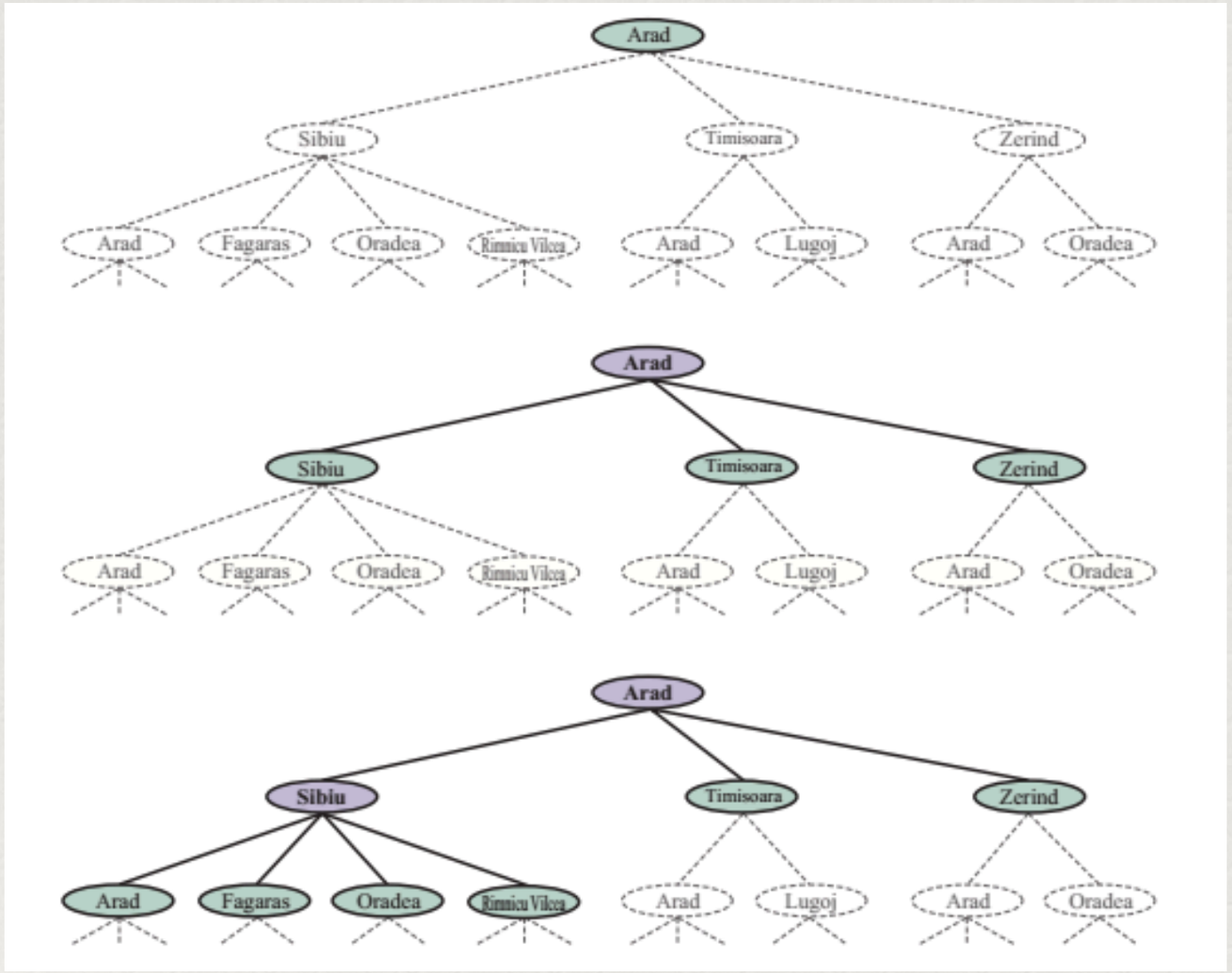
Hans Peter Moravec

Alberto Elfes

Seja qual for a aplicação, e a complexidade dos algoritmos, os métodos de busca são um elemento fundamental no processo de "resolução de problemas".

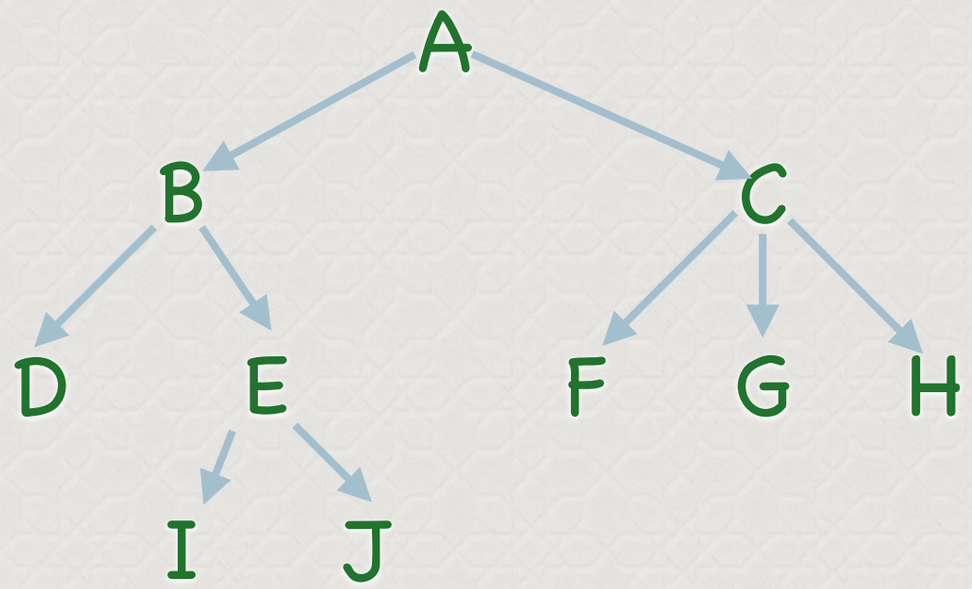


Árvore de Busca



Expandindo a árvore de busca

E como se representa uma árvore de busca em Prolog?.



Listas

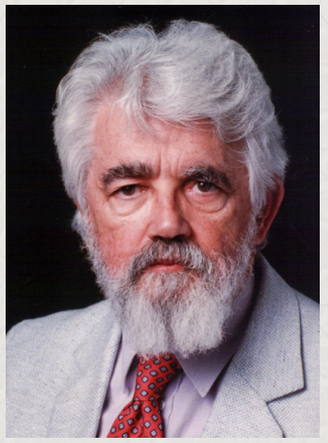
[head | tail]

Elemento

Lista

[ ] - Lista vazia

[a | [b | [d, [e | [i, j]]]], [c | [f, g, h]]]



John McCarthy
1927-2011

https://lisp-lang.org

Build reusable and extensible class hierarchies using the Common Lisp Object System. Design patterns disappear as you adapt the language to your problem domain.

```
(defclass book ()
  ((title :reader book-title
          :initarg :title)
   (author :reader book-author
           :initarg :author))
  (:documentation "Describes a book."))

(make-instance 'book
               :title "ANSI Common Lisp"
               :author "Paul Graham")
```

An example of SxQL, a macro-based SQL DSL

```
(select (:title :author :year)
  (from :books)
  (where (:and (:>= :year 1995)
               (:< :year 2010)))
  (order-by (:desc :year)))

⇒ ((:title "Practical Common Lisp"
    :author "Peter Seibel"
    :year 2005)
   (:title "ANSI Common Lisp"
    :author "Paul Graham"
    :year 1995))
```

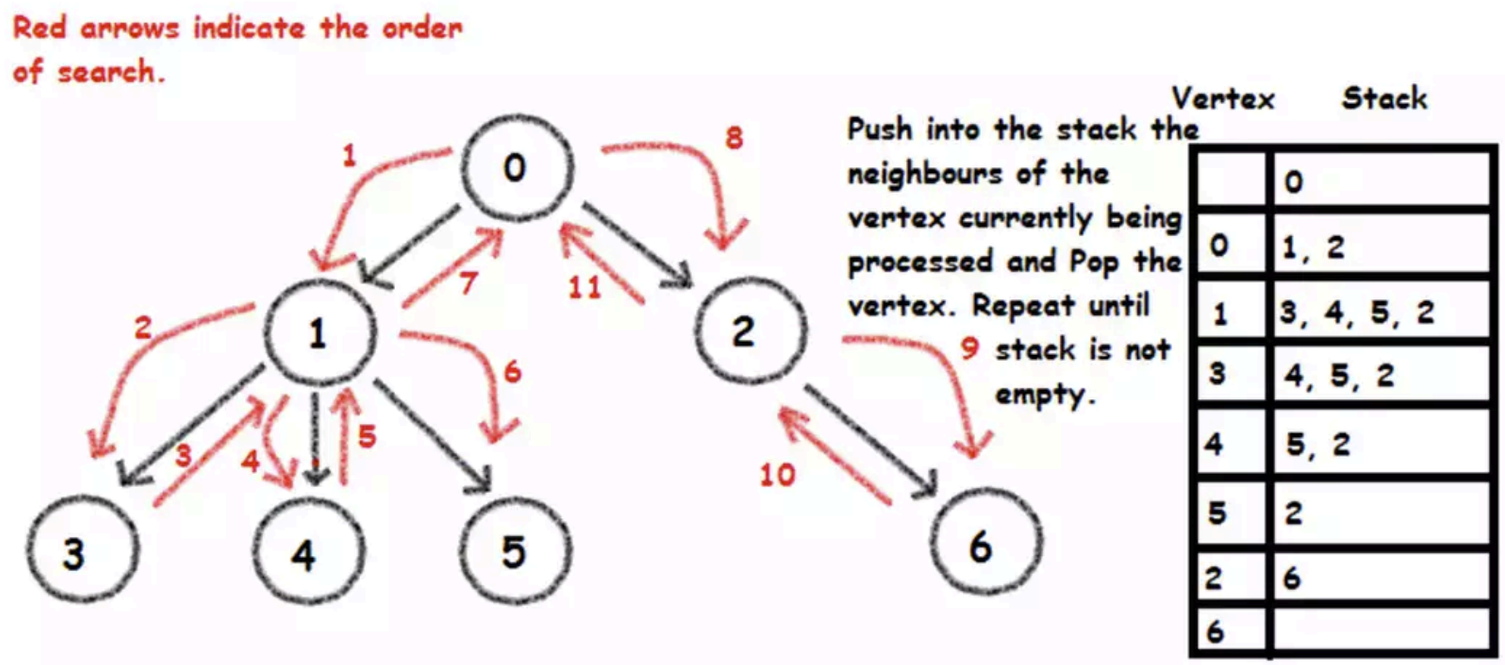
Seja qual for a estrutura de dados utilizada, a busca em árvore se caracteriza como um algoritmo de "varredura" em busca de um dado elemento. Eventualmente este elemento não pertence à árvore e a busca se encerra quando todos os elementos da árvore forem visitados.

A busca "cega" consiste em varrer toda a árvore de busca até encontrar uma solução, ou não ter mais possibilidades (em um espaço de busca finito). A busca cega também é chamada de "busca não-informada", isto é, não existe na base de conhecimento do agente nenhuma informação que permita diferenciar as ações admissíveis a serem escolhidas. (Ter uma função custo associada ao resultado das ações seria uma opção).

Métodos clássicos de busca não-informada:

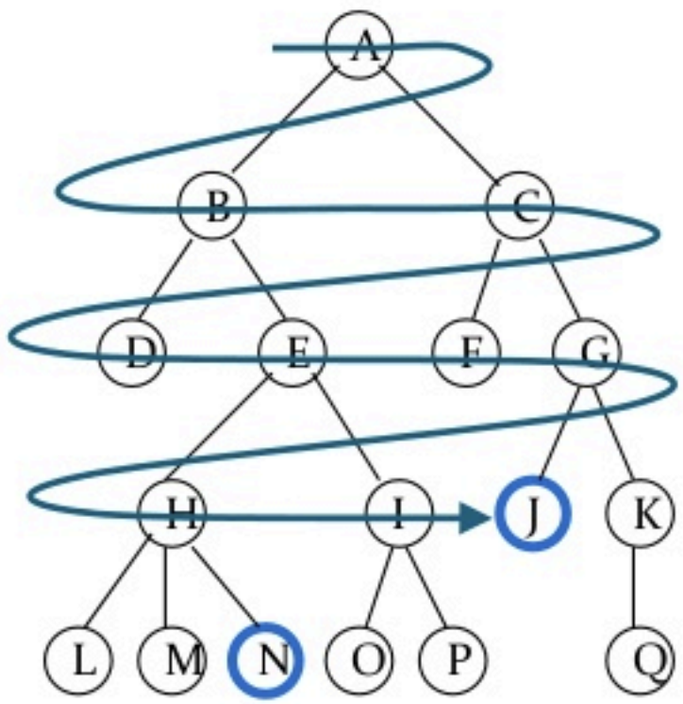
- Busca em profundidade
- Busca em largura

Busca em profundidade



Busca em largura

Breadth-first searching[1]

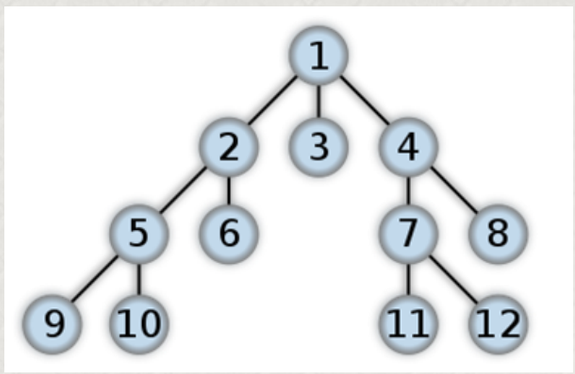


- A breadth-first search (BFS) explores nodes nearest the root before exploring nodes further away
- For example, after searching A, then B, then C, the search proceeds with D, E, F, G
- Node are explored in the order A B C D E F G H I J K L M N O P Q
- J will be found before N

Usando os métodos clássicos de busca não-informada

Comparando os dois métodos podemos concluir que cada um deles pode ser melhor aplicado em situações onde:

i) a solução está mais próxima da raiz, e a busca em largura será mais eficiente;
ii) a solução está nos nós de maior profundidade ou nas folhas, a neste caso a busca em profundidade será mais apropriada;



A busca "informada" consiste consiste em definir uma função custo associada aos nós. Assim, é possível comparar o efeito das ações para escolher, por exemplo, a função de "menor custo", na tentativa de otimizar o caminho até o estado final.

Na próxima aula trataremos dos algoritmos de busca não informada e dos algoritmos básicos, como o "best first" e dos algoritmos A e A*, e iniciaremos a discussão sobre o conceito de "heurística".

Até a próxima aula!