Universidade do Minho

Aula 8: Procura num Espaço de Soluções/Estados

OBJECTIVOS:

- Solucionar Problemas que envolvam uma procura num espaço de soluções/estados

Universidade do Minho

Procura num Espaço de Soluções/Estados:

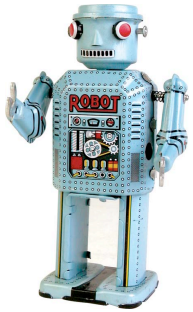
Resolver problemas de procura onde existem estados e transições.

Muitos do **problemas** em ciências da computação podem ser definidos como:

- Um conjunto **S** de ESTADOS (possivelmente infinito);
- Um estado INICIAL **s** $\in$ **S**;
- Uma relação de TRANSIÇÃO **T** ao longo deste espaço de estados, e
- Um conjunto de estados **FINAIS** (objectivos): **G** $\subseteq$ **S** ;

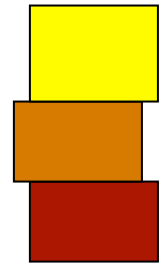
Ora, isto pode ser representado por um **GRAFO**, onde os **nodos** representam **estados** e os **arcos** (ou **conexões**) os pares da relação de **transição**;

EXEMPLOS:



- O estado inicial corresponde a um **robot** e alguns materiais. O objectivo é encontrar um estado em que os materiais formam um **produto final**;

- O estado inicial corresponde a **avião em Sidney** e possíveis rotas de viagem. O objectivo é encontrar a **rota mais curta** entre Sidney e Paris;
- O estado inicial corresponde a uma pilha de caixas onde a base contém B, A está por cima de B e C por cima de A. O objectivo é **rearranjar** as caixas de modo que a caixa A fique acima da caixa B, ficando esta acima da caixa C. Uma caixa só pode ser tirada por de cima de outra e vice-versa.



Universidade do Minho

Grafos

- Um **grafo** é constituído por um conjunto de **nodos** e **arcos/ conexões** entre os nodos;
- Os **arcos** podem ter **sentidos** (**grafo orientado**) e pesos (**grafo pesado**);
- Existem diversas formas de representar grafos em lógica.
- A título de exemplo, será utilizada a seguinte notação:

```
arco(nodo1,nodo2).   % arco entre o nodo1 e nodo2
```

Universidade do Minho

Exemplo de um Grafo:

```
arco(a,b).
arco(b,a).
arco(a,c).
arco(b,d).
arco(b,e).
arco(c,e).
arco(d,f).
arco(e,f).
arco(e,g).
```

Q1: Desenhe o grafo e mostre quais os caminhos entre a e g (In-Class Teams 2 min)

-Grupos de 3 elementos, o que for mais novo é o representante (caneta e papel).

Universidade do Minho

Sistema de Inferência: Caminho em Grafos

- Encontrar o caminho entre dois nodos é uma operação muito comum em grafos;
- Sejam **X** e **Z** dois nodos de um grafo. Um método para achar o **Caminho** consiste em:

```
caminho(X,Z,C) ←caminho(X,Z,[X],Caminho).
caminho(X,X,Caminho,Caminho).
caminho(X,Z,Visitado,Caminho)
            ← arco(X,Y),
               não(membro(Y,Visitado)),
        caminho(Y,Z,[Y | Visitado],Caminho).
```

Caminho entre a e g?

┐caminho(a,g,Caminho).

Universidade do Minho

Sistema de Inferência: caminho.pl (em Prolog)

```
% caminho.pl
caminho(X,Z,C):-caminho(X,Z,[X],Caminho).
caminho(X,X,Caminho,Caminho).
caminho(X,Z,Visitado,Caminho)
              :- arco(X,Y),
                 \+ member(Y,Visitado),
          caminho(Y,Z,[Y | Visitado],Caminho).
```

Caminho entre a e g?

```
?-caminho(a,g,Caminho).
```

Universidade do Minho

Estradas do Norte de Portugal:

A cidade do **Porto**, famosa pelo seu vinho, tem boas estradas até Famalicão e Esposende (e vice-versa);
Famalicão, famosa pelos seus doces, tem boas ligações a Barcelos, Braga e Porto (e vice-versa);
Braga, capital religiosa do Minho, tem uma auto-estrada que liga a Valença, Famalicão e Barcelos (e vice-versa);
Barcelos (famosa pelo seu Galo), tem estradas para Viana, Famalicão, Braga e Esposende (e vice-versa);
No norte do Minho, fica **Viana** (junto ao mar), com ligação a Valença, Esposende e Barcelos (e vice-versa);

Q2: Represente a BC e questione sobre quais os caminhos possíveis entre Viana e Famalicão?

(In-Class Teams 2 min)

-Grupos de 3 elementos, o que for mais velho é o representante (caneta e papel).

Universidade do Minho

O Problema dos Missionários e Canibais!

- Um grande rio divide a missão (lado direito) e a aldeia de canibais (esquerda), existindo um pequeno barco com espaço para duas pessoas;
- Actualmente, o barco está do lado direito (missão), existindo na mesma margem 3 missionários e 3 canibais;
- Todos querem atravessar o rio, mas se num dado momento existirem mais Canibais que missionários numa das margens, os canibais comem os missionários!
- **Como transportar toda a gente em segurança?**

(animação online em: http://www.plastelina.net/games/game2.html)

Q3: Encontre a solução (In-Class Teams, 2 min)

-Grupos de 3 elementos, o que for mais alto é o representante (caneta e papel).

Universidade do Minho

Representação do Conhecimento: Estados

- Podemos representar este problema como uma travessia ao longo de um espaço de estados;
- Nesta solução, os estados serão representados por

cm(CE,ME,CD,MD,boat). % se barco à direita

cm(boat,CE,ME,CD,MD). % se barco à esquerda

onde

CE/ME- **canibais/missionários à esquerda** e

CD/MD – canibais/missionários à direita

- **Estado inicial**: **initial(cm(0,0,3,3,boat)).**
- **Estado final**: **final(cm(boat,3,3,0,0)).**

Universidade do Minho

Representação do Conhecimento: Transicões

- Uma Transição muda de um estado (E1) para outro (E2) via uma dada acção (Action): **transition(E1,Action,E2)**

- Existem **10** transições:

1 Mover 2 Canibais para a esquerda: left(2,0)
2 Mover 1 canibal para a esquerda: left(1,0)
3 Mover 1 Canibal e 1 Missionário para a esq: left(1,1)
4 Mover 2 Missionários para a esquerda: left(0,2)
5 Mover 1 Missionário para a esquerda: left(0,1)
6 Mover 2 Canibais para a direita: right(2,0)
7 Mover 1 canibal para a direita: right(1,0)
8 Mover 1 Canibal e 1 Missionário para a dir: right(1,1)
9 Mover 2 Missionários para a direita: right(0,2)
10 Mover 1 Missionário para a direita: right(0,1)

Universidade do Minho

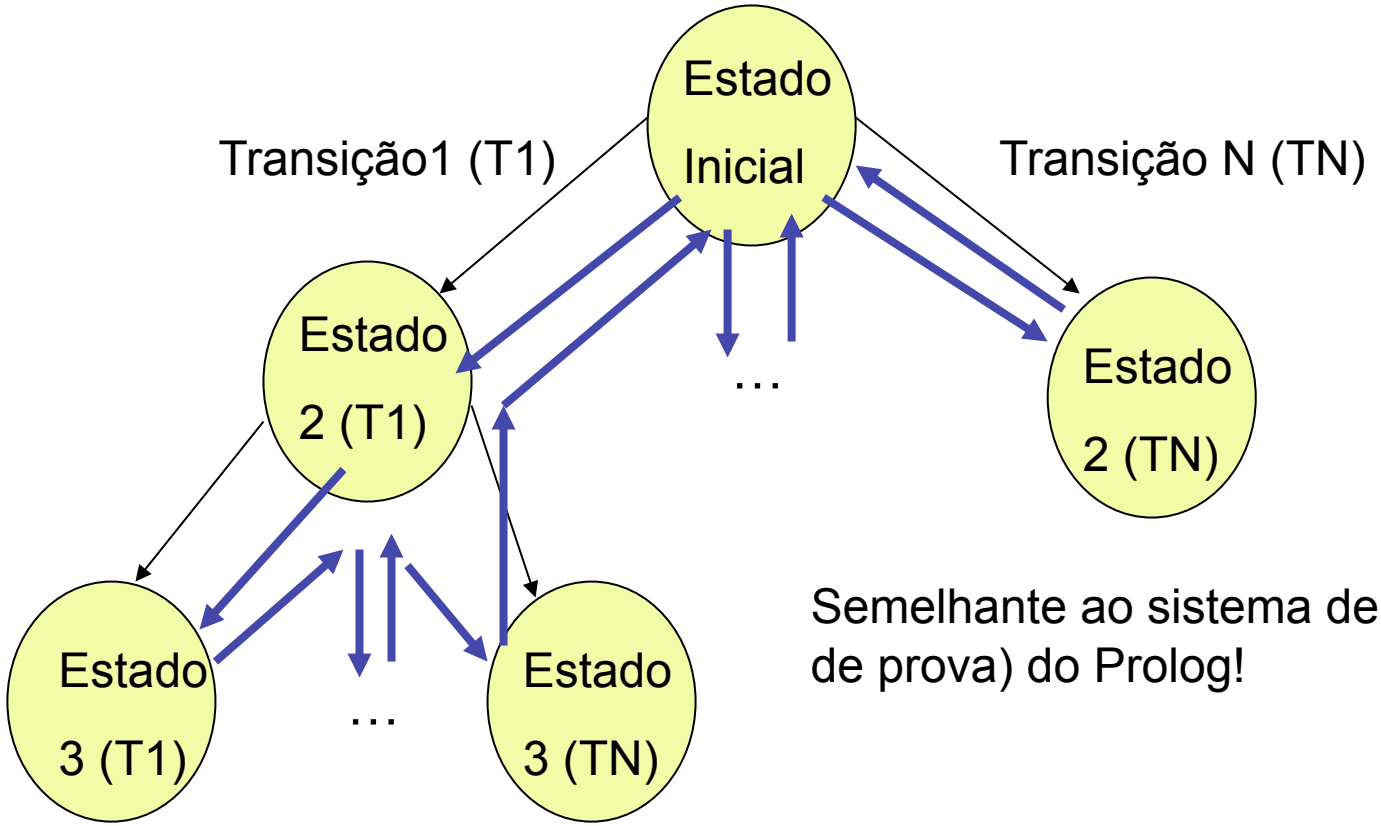
Representação do Conhecimento: Transicões

```
% 1 Mover 2 Canibais para a esquerda: left(2,0)
transition(cm(C1,M1,C2,M2,boat),left(2,0),cm(boat,NC1,M1,NC2,M2)):-
        C2 > 1,
        NC1 is C1 + 2,
        NC2 is C2 - 2,
        legal(NC1, M1),
        legal(NC2, M2).
```

```
% predicado auxiliar, diz se o estado final é válido:
legal(_, 0).
legal(C, M):- C =< M.
```

Sistema de Inferência: em profundidade

- A forma mais simples de efectuar uma procura num grafo consiste no algoritmo de **depth-first** (em profundidade)



Semelhante ao sistema de backtracking (árvore de prova) do Prolog!

Universidade do Minho

Sistema de Inferência: em profundidade

```
solvedf←initial(InitialState),
          solvedf(InitialState, [InitialState], Solution),
          escrever(Solution).
solvedf(State, _, []): ←
          final(State),!.

solvedf(State, History, [Move|Solution]) ←
          transition(State, Move, State1),
          não (membro(State1, History)),
          solvedf(State1, [State1|History], Solution).
```

Universidade do Minho

Sistema de Inferência em Prolog:

```
% solvedf.pl
solvedf:-
        initial(InitialState),
        solvedf(InitialState, [InitialState], Solution),
        write(Solution).
solvedf(State, _, []):-
        final(State),!.
solvedf(State, History, [Move|Solution]):-
        transition(State, Move, State1),
        \+ member(State1, History),
        solvedf(State1, [State1|History], Solution).
```

Uma solução para o problema:

% executar: ?- solvedf.

[left(2, 0), right(1, 0), left(2, 0), right(1, 0), left(0, 2), right(1, 1), left(0, 2), right(1, 0), left(2, 0), right(1, 0), left(2, 0)]

Universidade do Minho

Problema dos Canibais e Missionários Completo

```
% ler o sistema de inferencia
:-[solvedf].
% estados inicial e final
initial(cm(0,0,3,3,boat)).
final(cm(boat,3,3,0,0)).
% 10 transicoes: 5 para o left e 5 para o right
transition(cm(C1,M1,C2,M2,boat),left(2,0),cm(boat,NC1,M1,NC2,M2)):-
        C2 > 1,
        NC1 is C1 + 2,
        NC2 is C2 - 2,
        legal(NC1, M1),
        legal(NC2, M2).
transition(cm(C1,M1,C2,M2,boat),left(1,0),cm(boat,NC1,M1,NC2,M2)):-
        C2 > 0,
        NC1 is C1 + 1,
        NC2 is C2 - 1,
        legal(NC1, M1),
        legal(NC2, M2).
```

Universidade do Minho

Problema dos Canibais e Missionários Completo

```
transition(cm(C1,M1,C2,M2,boat),left(1,1),cm(boat,NC1,NM1,NC2,NM2)):-
        C2 > 0, M2 > 0,
        NC2 is C2 - 1, NM2 is M2 - 1,
        NC1 is C1 + 1, NM1 is M1 + 1,
        legal(NC1, NM1), legal(NC2, NM2).
transition(cm(C1,M1,C2,M2,boat),left(0,2),cm(boat,C1,NM1,C2,NM2)):-
        M2 > 1,
        NM1 is M1 + 2, NM2 is M2 - 2,
        legal(C1, NM1), legal(C2, NM2).
transition(cm(C1,M1,C2,M2,boat),left(0,1),cm(boat,C1,NM1,C2,NM2)):-
        M2 > 0,
        NM1 is M1 + 1, NM2 is M2 - 1,
        legal(C1, NM1), legal(C2, NM2).
transition(cm(boat,C1,M1,C2,M2),right(2,0),cm(NC1,M1,NC2,M2,boat)):-
        C1 > 1,
        NC1 is C1 - 2, NC2 is C2 + 2,
        legal(NC1, M1), legal(NC2, M2).
```

Universidade do Minho

Problema dos Canibais e Missionários Completo

```
transition(cm(boat,C1,M1,C2,M2),right(1,0),cm(NC1,M1,NC2,M2,boat)):-
        C1 > 0, NC1 is C1 - 1, NC2 is C2 + 1,
        legal(NC1, M1), legal(NC2, M2).
transition(cm(boat,C1,M1,C2,M2),right(1,1),cm(NC1,NM1,NC2,NM2,boat)):-
        C1 > 0, M1 > 0,
        NC1 is C1 - 1, NM1 is M1 - 1, NC2 is C2 + 1, NM2 is M2 + 1,
        legal(NC1, NM1), legal(NC2, NM2).
transition(cm(boat,C1,M1,C2,M2),right(0,2),cm(C1,NM1,C2,NM2,boat)):-
        M1 > 1, NM1 is M1 - 2, NM2 is M2 + 2,
        legal(C1, NM1), legal(C2, NM2).
transition(cm(boat,C1,M1,C2,M2),right(0,1),cm(C1,NM1,C2,NM2,boat)):-
        M1 > 0, NM1 is M1 - 1, NM2 is M2 + 1,
        legal(C1, NM1), legal(C2, NM2).
%-----------------------------------------------------------------
legal(_, 0).
legal(C, M):- C =< M.
% buscar a solucao, executar: solvedf.
```

Universidade do Minho

Limitações da Procura em Profundidade

- Em muitos problemas, o número de caminhos e de ramificações é de tal ordem que se torna **impossível** tentar **todos os caminhos**;
- A solução passa por utilizar um conjunto de **heurísticas** para reduzir o número potencial de caminhos;
- Em alternativa, pode utilizar-se a estratégia em **extensão** (***breadth-first***): percorre todos os caminhos de comprimento 1, depois de comprimento 2, etc… até encontrar a solução final;

Para saber mais...

- Consultar o Capítulos 11,12, 13, **Prolog – [Bratko, 1990]:** Bratko, Ivan, Prolog – Programming for Artificial Intelligence, Longman, 1990.

- **Mais exemplos de exercícios resolvidos de Procura via Transição de Estados na sebenta a disponibilizar na página da disciplina…**