PICAT: Uma Linguagem de Programação Multiparadigma

Miguel Alfredo Nunes, Jeferson L. R. Souza, Claudio Cesar de Sá

miguel.nunes@edu.udesc.br

jeferson.souza@udesc.br

claudio.sa@udesc.br

Departamento de Ciência da Computação
Centro de Ciências e Tecnológias
Universidade do Estado de Santa Catarina

9 de abril de 2019

Contribuições

- Alexandre Gonçalves;
- João Herique Faes Battisti;
- Paulo Victor de Aguiar;
- Rogério Eduardo da Silva;
- Hakan Kjellerstrand – (`http://www.hakank.org/picat/`)
- Neng-Fa Zhou – (`http://www.picat-lang.org/`)
- Outros anônimos que auxiliaram na produção deste documento;

Apresentação ao Curso de PICAT – I

- O que é o PICAT?

Apresentação ao Curso de PICAT – I

- O que é o PICAT?
  - Uma linguagem de programação de propósitos gerais
  - Uma evolução do PROLOG (consagrada linguagem dos primórdios da IA)
  - Tem elementos das linguagens Python, Prolog e Haskell
- Uso e finalidades do PICAT:

Apresentação ao Curso de PICAT – I

- O que é o PICAT?
  - Uma linguagem de programação de propósitos gerais
  - Uma evolução do PROLOG (consagrada linguagem dos primórdios da IA)
  - Tem elementos das linguagens Python, Prolog e Haskell
- Uso e finalidades do PICAT:
  - Uso de programas gerais; de simples à complexos (uma reflexão)
  - Provê suporte há vários solvers na área de Pesquisa Operacional
  - Área: IA, programação por restrições, programação inteira, planejamento, combinatória, etc

Apresentação ao Curso de PICAT – II

- Este curso é dirigido a voce?

Apresentação ao Curso de PICAT – II

- Este curso é dirigido a voce?
- Requisitos:

Apresentação ao Curso de PICAT – II

- Este curso é dirigido a voce?
- Requisitos:
  - Conhecimento: noções de lógica matemática (proposional e primeira-ordem), matemática elementar, e alguma outra linguagem de programação
  - Dedicação: depende de você

Apresentação ao Curso de PICAT – II

- Este curso é dirigido a voce?
- Requisitos:
  - Conhecimento: noções de lógica matemática (proposional e primeira-ordem), matemática elementar, e alguma outra linguagem de programação
  - Dedicação: depende de você
- Motivação:

Apresentação ao Curso de PICAT – II

- Este curso é dirigido a voce?
- Requisitos:
  - Conhecimento: noções de lógica matemática (proposional e primeira-ordem), matemática elementar, e alguma outra linguagem de programação
  - Dedicação: depende de você
- Motivação:
  - Dependendo de sua dedicação, ao final voce vai estar apto a resolver problemas computacionais de simples à difíceis
  - Difícil: muitas linhas de código e muito conhecimento de algoritmos seriam necessários
  - Com Picat, há sofisticados esquemas prontos para se construir programas.

Apresentação ao Curso de PICAT – III

- Requisitos computacionais:

Apresentação ao Curso de PICAT – III

- Requisitos computacionais: Um computador qualquer (arquitetura 16, 32 ou 64 bits), com Linux, Mac ou Windows, que tenha um compilador C instalado completo, preferencialmente.
- Comunidade e ações: `http://picat-lang.org`

Apresentação ao Curso de PICAT – III

- Requisitos computacionais: Um computador qualquer (arquitetura 16, 32 ou 64 bits), com Linux, Mac ou Windows, que tenha um compilador C instalado completo, preferencialmente.
- Comunidade e ações: `http://picat-lang.org`
- Códigos e este material, sempre atualizados em:

Apresentação ao Curso de PICAT – III

- Requisitos computacionais: Um computador qualquer (arquitetura 16, 32 ou 64 bits), com Linux, Mac ou Windows, que tenha um compilador C instalado completo, preferencialmente.
- Comunidade e ações: `http://picat-lang.org`
- Códigos e este material, sempre atualizados em:
  - Este PDF e seu texto original: `http://github.com/claudiosa/Slides_Picat`
  - Os códigos de programas: `http://github.com/claudiosa/CCS/picat`
- Além do material aqui disponível em PDF, o mais importante do curso vai estar na interatividade da minha **apresentação oral**.
- Ou seja, este material é um guia para o seu desenvolvimento, mas minhas explicações são horas de estudo e código feito
- Tópicos que serão cobertos no curso:

Sumário I

Buscas

- Requisito: conceitos de listas e recursividade dominados!

Buscas

- Requisito: conceitos de listas e recursividade dominados!
- Além destes: conceitos grafos, árvores de busca, nós, etc

Buscas

- Requisito: conceitos de listas e recursividade dominados!
- Além destes: conceitos grafos, árvores de busca, nós, etc
- Pois, problemas em geral se apresentam como uma conexão complexa tipo um *grafo*, e a varredura sob este grafo é sistemática sob uma *árvore de busca*

Buscas

- Requisito: conceitos de listas e recursividade dominados!
- Além destes: conceitos grafos, árvores de busca, nós, etc
- Pois, problemas em geral se apresentam como uma conexão complexa tipo um *grafo*, e a varredura sob este grafo é sistemática sob uma *árvore de busca*
- Então, computar listas em Picat, é a nossa estratégia de resolver problemas!

Ciclo Euleriano I

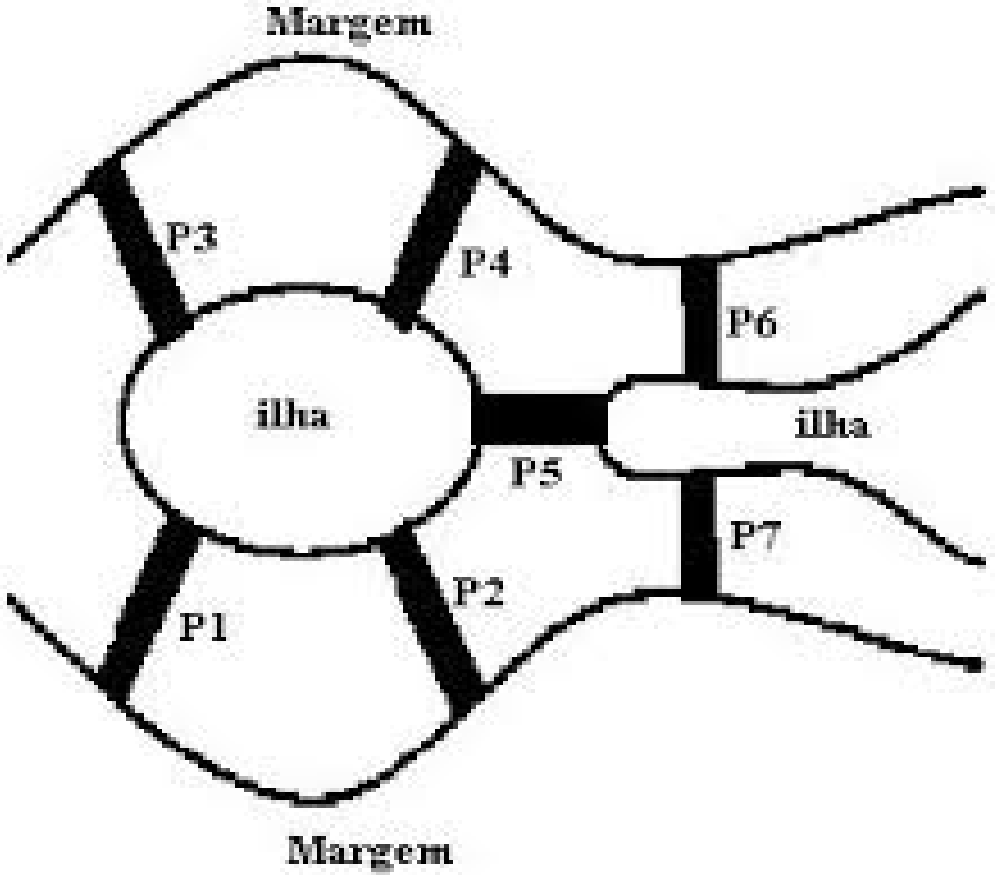

Figura 1: Ciclo Euleriano – Problema das Pontes de Königsberg

Ciclo Euleriano II

- No século 18 havia na cidade de Königsberg (antiga Prússia) um conjunto de sete pontes (identificadas pelas letras de P1 até P7 na figura ao lado) que cruzavam o rio Prególia. Elas conectavam duas ilhas entre si e as ilhas com as margens esquerda e direita.
- Os habitantes daquela cidade perguntavam-se se era possível cruzar as sete pontes numa caminhada contínua sem que se passasse duas vezes por qualquer uma das pontes.
- Embora intrigante, este problema foi atacado por Leonard Euler (1736) e demonstrou que isto não era possível para um grafo qualquer
- Curiosamente, este problema, computacionalmente é *fácil* de resolver!

Caminho Hamiltoniano I

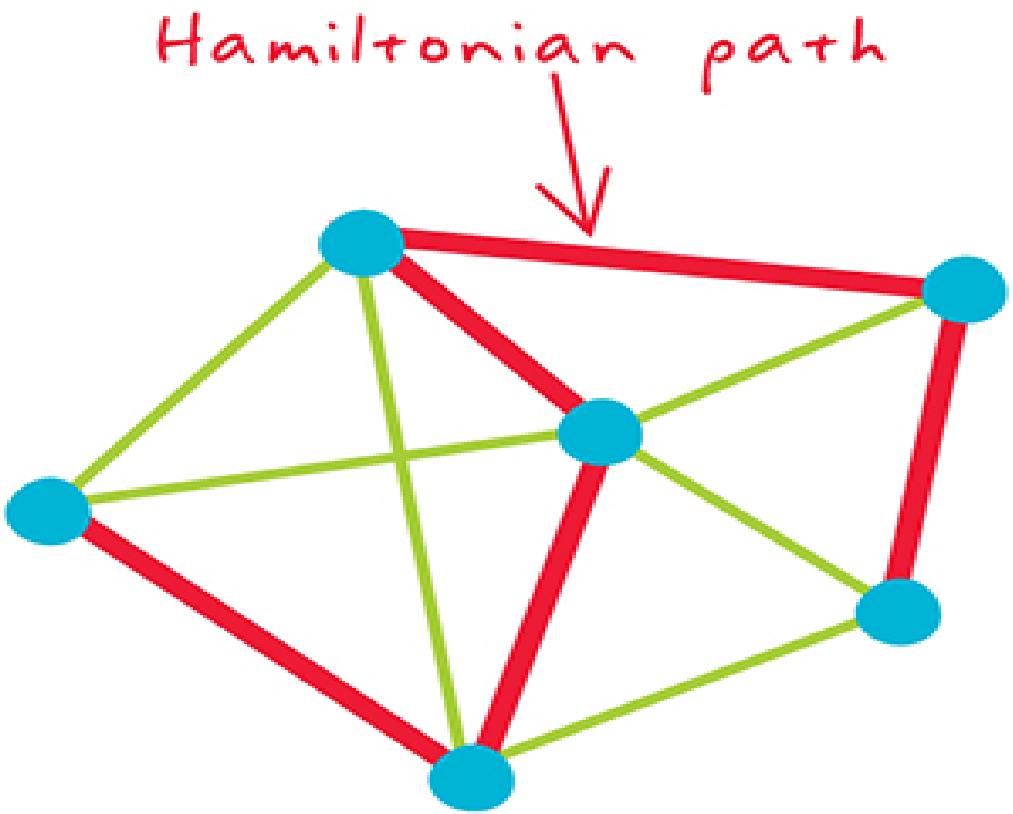

Figura 2: Caminho Hamiltoniano – Há um caminho que passe por todas cidades uma única vez?

Caminho Hamiltoniano II

- Diferente do ciclo Euleriano, o caminho Hamiltoniano, origem e destino são diferentes
- Todos os nós precisam ser visitados uma única vez sem repetição
- Num grafo pode haver muitos caminhos Hamiltonianos, mas, pode não existir nenhum!
- Ao contrário do ciclo Euleriano, este problema, computacionalmente é *difícil* de resolver!
- Mas é este que vamos usar como exemplo, com um algoritmo ingênuo.

Problemas, Estados, Grafos e Árvores de Buscas I

Contextualizando estes termos:

- Em geral, problemas podem ser vistos como *fotografias instantâneas* de uma situação, isto é, **um estado discreto**
- Uma *sucessão* destes estados, compõem *um caminho* de um estado *i* ao estado *j*
- Assim, estes *estados* são representados pelos *nós dos grafos*, e a ligação entre estes, são resultados de *uma ação*, mudança ou evolução do problema
- Há um estado particular chamado *inicial*, vários outros de estados *intermediários*, e outros estados *finais*
- Se o problema tiver várias soluções, o mesmo apresenta vários caminhos do estado inicial ao final.

Problemas, Estados, Grafos e Árvores de Buscas II

- Assim uma sucessão ou transição válida entre estados, é conhecido como uma *solução* ou *prova* do problema
- Essencialmente vamos varrer uma estrutura entre estados ou nós, de modo sistemático até encontrarmos uma solução aceitável/desejável.
- Logo, vamos empregar alguns conceitos da teoria dos grafos, em modelar problemas e resolvê-los por um esquema de busca computacional

Problemas de Grafos se Transformam em Árvores de Buscas

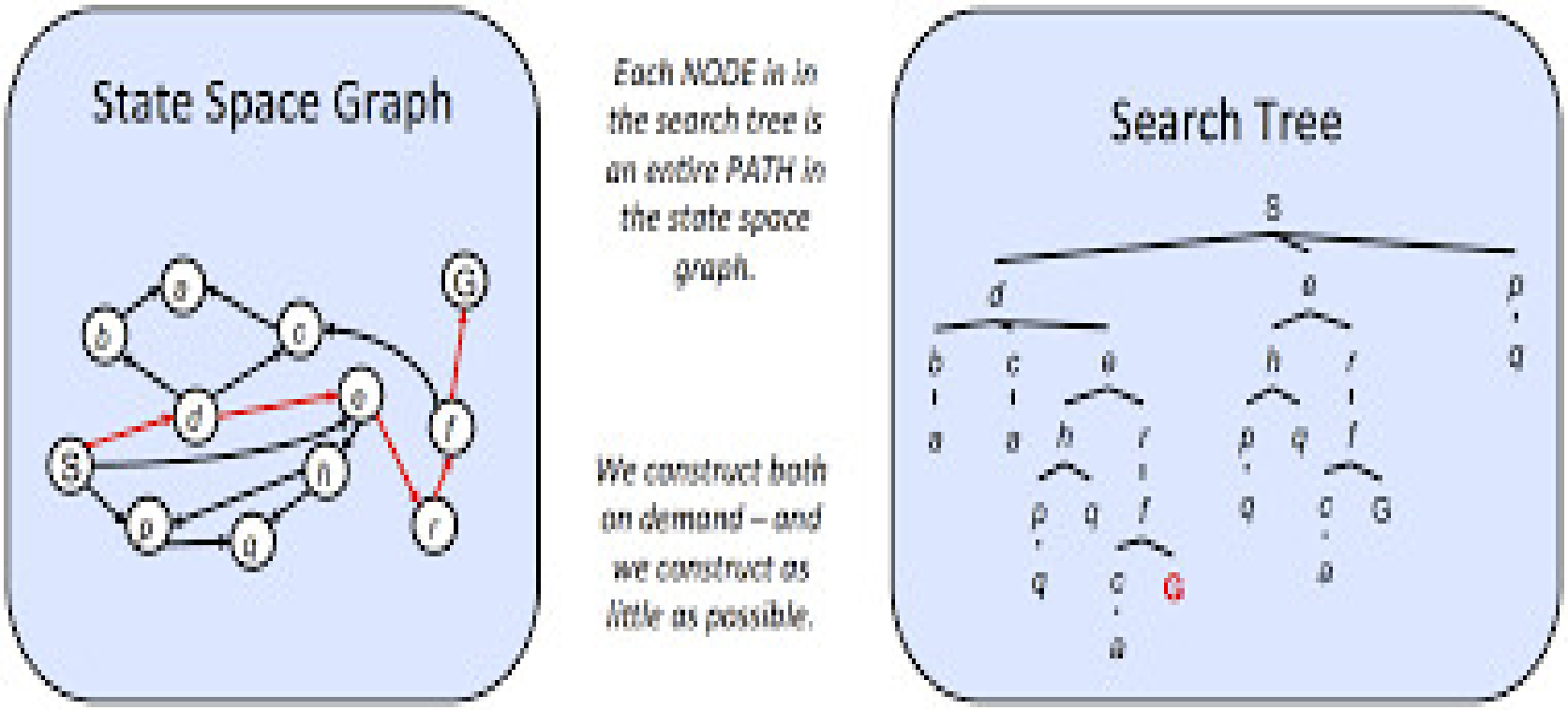

Figura 3: Google ...

Resumindo, os problemas são modelados em estruturas complexas, tais como grafos, mas o processo de solução se mantém: realizar uma busca, tal como uma estrutura de uma árvore

Núcleo Geral de Buscas I

Pseudo-código já em Picat

```
resolve(P) =>
      inicio(Start),
      busca(Start,[Start],Qsol),
      imprime_saida(Qsol,P).

busca(S,P,P) ?=>  objetivo(S).     % objetivo alcancado : FIN
busca(S,Visited,P) =>
     proximo_estado(S,Nxt),        % gera um proximo estado
     estado_seguro(Nxt),           % verifica se este estado
     sem_loop(Nxt,Visited),        % verifica se está em loop
     busca(Nxt,[Nxt|Visited],P).   % continue a busca recursi
```

Núcleo Geral de Buscas II

```
sem_loop(Nxt,Visited) :-
      \+member(Nxt,Visited).

proximo_estado(S,Nxt) =>   < fill in here >.
estado_seguro(Nxt) =>     < fill in here >.
sem_loop(Nxt,Visited) =>  < fill in here >.

inicio(...).
objetivo(...).
```

Vamos reescrever este pseudo-código em um problema!

Caminho Hamiltoniano Aplicado

Seja um viajante que sai cedo de Joinville, e chegar a noite em Blumenau, passando por algumas destas cidades uma única vez!

O nosso viajante I

- Em nosso problema temos 7 cidades pré-escolhidas
- A lista de cidades são:

```
index(-) %% lista de todas cidade do mapa
as_cidades( [ brusque, blumenau, itajai, luiz_alves,
              jaragua, sao_bento, joinville ] ).
```

- Duas cidades em particular

```
index(-)
destino( blumenau ).

index(-)
origem( joinville ).
```

- As estradas transitáveis entre as cidades definem o nosso mapa, consequentemente um grafo entre cidades:

O nosso viajante II

```
%% MAPA da região
index(-,-)
arco(joinville, sbs) .
arco(joinville, itajai) .
arco(jaragua, sbs) .
arco(jaragua, blumenau) .
arco(blumenau, itajai) .
arco(brusque, itajai) .
arco(brusque, blumenau) .
```

- Claro, este problema é pequeno e construindo o grafo dá para perceber que existe algumas soluções
- Para resolver este problema vamos utilizar uma *busca em profundidade*

O nosso viajante III

- Esta *busca em profundidade*, encontra-se inserida no contexto buscas em geral, visto anteriormente

O código

Acompanhar as explicações do código de:
`https://github.com/claudiosa/CCS/blob/master/picat/hamiltoniano_DFS.pi`

Resumindo

- Picat é jovem (nascida em 2013);
- Uma evolução ao Prolog após seus mais de 40 anos de existência e sucesso!
- Sua sintaxe é moderna;
- Código aberto, multi-plataforma, e repleta de possibilidades;
- Uso para fins diversos;
- Muitas bibliotecas específicas prontas: CP, SAT, Planner, etc;
-
- .