PICAT: uma Linguagem Multiparadigma

Claudio Cesar de Sá, Rogério Eduardo da Silva, João Herique Faes Battisti, Paulo Victor de Aguiar

joaobattisti@gmail.com

pavaguiar@gmail.com

claudio.sa@udesc.br

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

Objetivos desta Vídeo-Aula – 01

- Apresentação da Linguagem de Programação (LP): PICAT
- Contexto
- Características
- Como instalar
- Como usar
- Exemplos
- Referências
- Estes slides e outros:
  `https://github.com/claudiosa/CCS/tree/master/picat/slides_picat`
- **Pré-requisitos: noções de lógica e LPs $\Rightarrow$ muitos vídeos bons!**

Sumário

Histórico

- Criada em 2013 por Neng-Fa Zhou e Jonathan Fruhman
- Utiliza o B-Prolog como base de implementação, e ambas utilizam a programação em lógica: Lógica de Primeira-Ordem (LPO)
- Uma evolução ao Prolog após seus mais de 40 anos de sucesso!
- Sua atual versão é a 2.0 (2 de fevereiro de 2017)

O que é multiparadigma?

- Imperativo – Procedural
- Funcional
- Lógico
- Uma boa *mistura* de: Haskell, Prolog e Python

Algumas Características:

- Sintaxe $\Rightarrow$ elegância do código
- Velocidade de execução
- Portabilidade (todas plataformas)
- Extensão há outras ferramentas

Anacrônico de **P.I.C.A.T.**

P: *Pattern-matching*: Utiliza o conceito de casamento padrão da LPO

I: *Intuitive*: oferece atribuições e laços de repetições análogo as outras linguagens de programação

C: *Constraints*: suporta a programação por restrições

A: *Actors*: suporte as chamadas a eventos, os atores (futuro gráfico)

T: *Tabling*: implementa a técnica de *memoization*, com soluções imediatas para problemas de Programação Dinâmica.

Instalação do PICAT

- Baixar a versão desejada de `http://picat-lang.org/download.html`
- Descompactar. Em geral em **/usr/local/Picat/**
- Criar um link simbólico (linux) ou atalhos (Windows):
  `ln -s /usr/local/Picat/picat /usr/bin/picat`
- Se quiser adicionar (opcional) uma variável de ambiente:
  `PICATPATH=/usr/local/Picat/`
  `export PICATPATH`
- ou ainda adicione o caminho: `PATH=$PATH:/usr/local/Picat`
- Finalmente, tenha um editor de código de programa.
  Sugestão: *geany* ou *sublime*

Usando do Picat

- Picat é uma linguagem de multiplataforma, disponível em qualquer arquitetura de processamento e também de sistema operacional. Nesta vídeo-aula: Linux (Manjaro)
- Em seus arquivos fontes utiliza a extensão **.pi**. Exemplo: `programa.pi`
- Existem 2 modos de utilização do Picat: modo linha de comando (ou console) e modo interativo
- Códigos executáveis 100% **stand-alone**: ainda não!
- Neste quesito, estamos em igualdade com Java, Prolog e Python

Fatos e Regras – os pais!

- *pai*(*platao*, *luna*) leia-se: *Platão é o pai de Luna*
- *pai*(*platao*, *pricles*) leia-se: *Platão é o pai de Péricles*
- *pai*(*epimenides*, *platao*) leia-se: *Sócrates é o pai de Platão*
- Codificando tudo isto em Picat

Regras em PICAT (1)

```
1  %%% FATOS ...  desenhe a arvore geneologica
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  index(-,-)     %% definindo FATOS
4          pai(platao, luna).
5          pai(platao, pericles).
6          pai(platao, eratostenes).
7          pai(epimenides, platao).
8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  %% REGRAS: exemplos
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11 %% definindo um avo: pai do pai
12 avo(X,Y) =>  pai(X,Z), pai(Z,Y).
13
14 %% definindo um irmao: alguem que tenha o mesmo pai
15 irmao(X,Y) => pai(Z,X), pai(Z,Y), X !== Y.
16
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18 %% MAIS REGRAS
19 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20
21 listar_pais  ?=>    %%% ?=>  regra "backtrackavel"
```

Regras em PICAT (2)

```
22      pai(X,Y) ,
23      printf("\n ==> %w  e pai de  %w", X , Y)  ,
24      false.
25
26  listar_pais =>
27      printf("\n ")  ,
28      true. %% the final rule of above
29
30  listar_avos  ?=>    %%% ?=>  regra "backtrackavel"
31      avo(X,Y) ,
32      printf("\n ==> %w  e avo de  %w", X , Y)  ,
33      false.
34
35  listar_avos =>
36      printf("\n ")  ,
37      true. %% the final rule of above
38  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39  %% main ... facilidade no uso console
40  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41  main ?=>          %%% ?=>  regra "backtrackavel"
42   %    listar_pais,
43   %    listar_avos,
```

Regras em PICAT (3)

```
44  %    avo(X,Y), printf("\n ==> %w  eh avo de  %w", X , Y)   ,
45      irmao(Z,W), printf("\n ==> %w  eh irmao de  %w", Z , W),
46      false.
47  main => true.
48  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Tipos de Dados

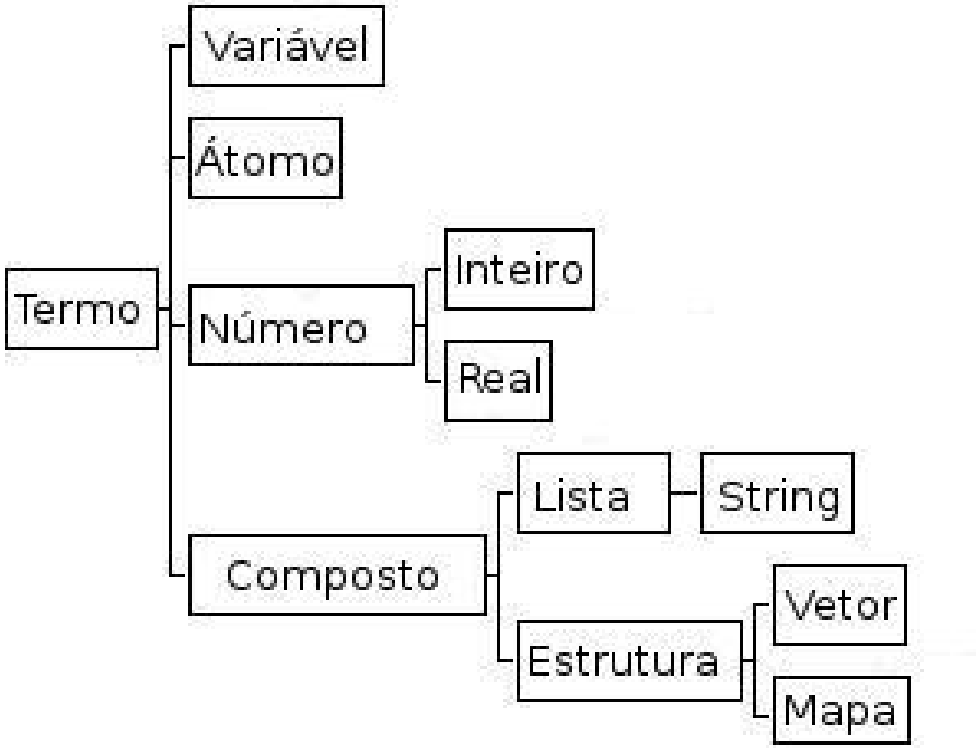

Figura: Hierarquia dos Tipos de dados

Número

```
Picat> A = 5, B = 7, number(A), number(B), max(A, B) =
Maximo, min(A, B) = Minimo.
A = 5
B = 7
Maximo = 7
Minimo = 5
yes.
```

Atribuição

```
Picat> X := 7, X := X + 7, X := X + 7.
X = 21
```

Estruturas de Controle

```
1 teste =>
2       X := 3,
3       Y := 4,
4       if (X >= Y)
5       then
6             printf("\n X eh maior: %d\n" , X)
7       else
8             printf("\n senao Y eh maior: %d\n" , Y)
9       end.
```

Entradas e Saídas

```
1 main =>
2             printf("\n Digite dois numeros:  "),
3             N_real_01 = read_real(),
4             N_real_02 = read_real(),
5             Media = (N_real_01 + N_real_02) / 2,
6             printf(" A media eh: %6.4f ", Media ),
7             printf("\n ..... FIM ..... \n ").
```

Conclusão

- PICAT é uma linguagem nova (2013), desconhecida, revolucionária e com um futuro promissor
- Atualmente há pouco material disponível e uma comunidade pequena de usuários
- Uso muito bom quanto a: Planejamento, Programação por Restrição e PD (diretamente)
- Todos problemas NPs-Completos!

Referências

- O *User Guide* que está no diretório `doc/` da instalação em LaTeX
- Meu GitHub ⇒ `https://github.com/claudiosa/CCS/tree/master/picat`
- `http://picat-lang.org/` – *User Guide on-line* está lá
- Assinem o fórum do PICAT(em inglês: respondo lá também)
- Site do Hakan Kjellerstrand ⇒ `http://www.hakank.org/picat/`
- Site do Roman Barták ⇒ `http://ktiml.mff.cuni.cz/~bartak/`
- Site do Sergii Dimychenko ⇒ `http://sdymchenko.com/blog/2015/01/31/ai-planning-picat/`

Obrigado

π + 

Retornem os comentários para o próximo vídeo!!!