Picat: uma Linguagem Multiparadigma

João Herique Faes Battisti, Paulo Victor de Aguiar

joaobattisti@gmail.com

pavaguiar@gmail.com

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

1 de Agosto de 2016

Sumário

História

- Foi criada em 2013 pelos cientistas da computação Neng-Fa Zhou e Jonathan Fruhman.
- Utilizou o B-Prolog como base de implementação, e ambas utilizam regras lógicas na programação.
- Picat 0.1 – Teve seu lançamento em Maio de 2013.
- Picat 1.0 – Foi lançada Abril de 2015.
- Sua atual versão é a 1.9 com lançamento em Maio de 2016

Picat é Multiparadigma

- Imperativo
- Funcional
- Lógico

Linguagem Multiparadigma

Motivo de existencia dos paradigmas?

- Velocidade de implementação
- Velocidade de execução
- Elegância do código

Linguagem Picat

- A terminologia de picat segue as bases teóricas da linguagem Prolog.
- Lógica de primeira-ordem onde objetos são chamados por Termos.
- O destaque de Picat é a sua natureza declarativa, funcional, tipagem dinâmica, sintaxe simples mas poderosas.
- PICAT é um anacronico onde cada letra representa uma característica marcante de sua funcionalidade.

Características: P-I-C-A-T

Pattern-matching:
Utiliza o conceito de casamento padrão. Um predicado define uma relação e pode ter zero ou várias respostas. Uma função é um predicado especial que sempre retorna uma única resposta. Ambos são definidos com regras de Picat, e seus predicados e funções seguem as regras do casamento.

Características: P-I-C-A-T

Intuitive:
O Picat oferece atribuições e laços de repetições para a programação dos dias de hoje. Uma variável atribuída pode imitar várias variáveis lógicas, alterado seu valor seguindo o estado da computação. As atribuições são úteis para associar os termos, bem como utilizadas nas estruturas de laços repetitivos.

Constraints:
Picat suporta a programação por restrições. Dado um conjunto de variáveis, cada uma possui um domínio de valores possíveis e restrições para limitar os valores a serem atribuídos nas variáveis. O objetivo é atribuir os valores que satisfaçam todas as restrições.

Actors:
Atores são chamadas orientadas à eventos. Em Picat, as regras de ação descrevem comportamentos dos atores. Um ator recebe um objeto e dispara uma ação. Os eventos são postados via canais de mensagem e um ator pode ser conectado há um canal, verificar e/ou processar seus eventos postados no canal. Neste ponto, estes atores desempenham características de área de Inteligência Artificial, especificamente a áreas de planejamento e sistemas multi-agentes.

Tabling:
Considerando que operações entre variáveis podem ser armazenadas parcialmente em uma tabela na memória, permitindo que um programa acesse valores já calculados. Assim, evita-se a repetição de operações já realizadas. Com esta técnica de memoization, o Picat oferece soluções imediatas para problemas de programação dinâmica.

Comparações

Tabela: Comparativo entre algumas linguagens:
`https://rosettacode.org/wiki/Language_Comparison_Table`

	C	Haskell	Java	Prolog	P.I.C.A.T
Paradigma(s)	procedural	funcional	orientado à objetos	lógico	multi-paradigma
Tipagem	fraca	forte	forte	fraca	fraca
Verificação de tipos	estático	estático	estático	dinâmico	dinâmico
Possui segurança?	não	sim	sim	não	sim
Possui coletor de lixo?	não	sim	sim	sim	sim
Passagem de parâ-metros	valor	-	valor	valor	casamento
Legibilidade	baixa	média	média	média	boa

Usos

A linguagem Picat pode ser utilizada para diversas funções:

- Acadêmica
- Industrial
- Pesquisas

Sistema de Programação

- Picat é uma linguagem de multiplataforma, disponível em qualquer arquitetura de processamento e também de sistema operacional
- Utiliza a extensão .pi em seus arquivos de código fonte.
- Existem 2 modos de utilização do Picat: Modo linha de comando e Modo Interativo.

Vantagens

- Enfatiza uma visão moderna e controlável em seu mecanismo de backtracking.
- Clareza em construir regras declarativas.
- Funções disponíveis numa sintaxe análoga a Haskell com um ambiente de programação análogo ao Python.
- Biblioteca é organizada em módulo a exemplo de Haskell e Python.

Desvantagens

- Manteve as letras maiúsculas para variáveis, como feito no B-Prolog.
- A geração de um código executável ainda não é puro, ela ainda se encontra em desenvolvimento
- As estruturas de repetição, comparadas com outras imperativas, ficam com uma sintaxe diferente.

Tipos de Dados

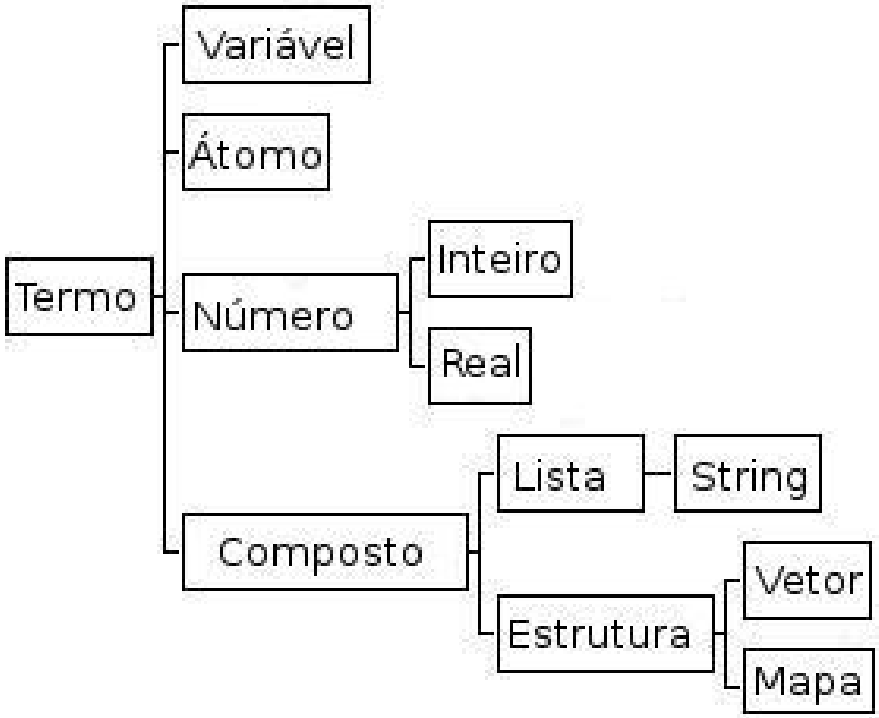

Figura: Hierarquia dos Tipos de dados

Variável

- As variáveis em Picat são similares as variáveis das matemática, pois ambas guardam valores. Diferentemente das linguagens imperativas, as variáveis em Picat não possuem um endereço simbólico na memória do computador.
- Quando uma variável ainda não foi instanciada com um valor, ela fica em um estado livre. Uma vez quando for instanciada com um valor, ela terá a mesma identidade como se fosse um valor até que ela seja liberada de novo.

Átomo

- Um átomo é uma constante simbólica e seu nome pode ser representado tanto com aspas simples ou sem.
- Um átomo não pode ultrapassar uma linha de comando e seu nome tem um limite de mil caracteres.
  Ex: x, x_1, 'a', 'b1'

Número

- Um número é um átomo inteiro ou real. Um número inteiro pode ser representado na forma decimal, binária, octal ou hexadecimal.
- Já o número real usa o ponto no lugar da virgula para separar os valores depois de zero como: 3.1415.

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.
```

Termos Compostos

Um termo composto se divide entre listas, estruturas e outros tipos compostos derivado destes são: *strings*, vetores e mapas.
Entretanto, ambos tem seus elementos acessados via casamento de padrões de fatos, predicados e funções.

Listas

A forma de uma lista reúne um conjunto de termos e os coloca dentro de colchetes: [t1; t2; :::; tn]. Veja o exemplo:

Listas

```
Picat> A=[1,2,3], list(A), length(A)=L_A, B= [4,5,6],
list(B),
length(B) = L_B, A ++ B = C, list(C), length(C) = L_C.
A = [1,2,3]
L_A = 3
B = [4,5,6]
L_B = 3
C = [1,2,3,4,5,6]
L_C = 6
yes.
```

Estruturas

A forma de uma estrutura é definida como $s(t1, t2, ..., tn),onde s é um átomo e $ é usado para diferenciar uma função. Seus principais elementos são o nome da estrutura que é o átomo que fica na frente e a aridade (número de argumentos do predicado). Veja o exemplo:

Estruturas

```
Picat> N = $nome(1,2,3,4,5), struct(N), arity(N) =
Aridade,
to_list(N) = Lista.
N = nome(1,2,3,4,5)
Aridade = 5
Lista = [1,2,3,4,5]
yes.
```

Estruturas

```
Picat> N = $(1,2,3,4,5), struct(N), arity(N) =
Aridade,
to_list(N) = Lista.
N = (1,2,3,4,5)
Aridade = 2
Lista = [1,(2,3,4,5)]
yes.
```

Exemplos

Atribuição

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

Estruturas de Controle

```
ex1 =>
X:=3, Y:=4,
if(X >= Y)
then printf("%d", X)
else printf("%d", Y)
end.
```

Soma até N

```
Soma como predicado
soma_p(0,S) => S = 0.
soma_p(N,S), N > 0 =>
                    soma_p(N-1, Parcial),
                    S = N + Parcial.

Soma como Função - Classic
soma_f1(0) = S => S = 0.
soma_f1(N) = S, N >= 1 => S = N + soma_f1 (N-1).

Soma como função de fatos - próx.  à Haskell
soma_f2(0) 0.
soma_f2(N) = N + soma_f2 (N-1).
```

Entradas e Saídas

```
main =>
printf("Digite dois números:  "),
N_real01 = read_real(),
N_real02 = read_real(),
Media = (N_real01 + N_real02)/2,
printf("A média é:  %6.2f", Media),
printf("\n.....FIM.....\ n").
```

Conclusão

- PICAT é uma linguagem nova (2013), desconhecida, revolucionária e com um futuro promissor para áreas de pesquisas e utilização comercial.
- Atualmente há pouco material disponível e uma comunidade pequena de usuários, mas existe um site atualizado e mantido por Hakan Kjellerstrand e um fórum de discussão no próprio site que está cada dia mais ativo, graças ao crescimento de usuários desta linguagem.

Referências

- https://github.com/claudiosa/CCS/tree/master/picat
- http://picat-lang.org/

Questionário

1. Qual característica do P.I.C.A.T é mais chamativa?
2. Em quais aplicações você usaria P.I.C.A.T?
3. Quais são os pontos positivos e negativos do P.I.C.A.T que você identifica?
4. Se pudesse melhorar algo no P.I.C.A.T, o que melhoraria?
5. O P.I.C.A.T pode substituir alguma linguagem?