INE5416 Paradigmas de Progração (Lógico)

Prof. A. G. Silva

11 de agosto de 2016

Disciplina

INE5416 – Paradigmas de Programação

- **Turma:** 04208
- **Professor:** Alexandre Gonçalves Silva
  - `www.inf.ufsc.br/~alexandre.silva`
  - `alexandre.goncalves.silva@ufsc.br`
  - Sala INE-506
- **Carga horária:** 90 horas-aula • Teóricas: 30 • Práticas: 60
- **Curso:** Ciência da Computação (208)
- **Requisitos:** INE5408 – Estruturas de Dados
- **Período:** 2º semestre de 2016
- **Materiais:** https://moodle.ufsc.br/course/view.php?id=65423
- **Horários:**
  - 4ª 13h30 (2 aulas) - CTC113 – Prof. João C. L. Dovicchi
    Paradigmas: Imperativo e Funcional
  - 5ª 07h30 (3 aulas) - LABPCT – Prof. Alexandre Gonçalves Silva

    Paradigma: Lógico

Ementa

Caracterização e classificações dos paradigmas. Problemas tratáveis pelos paradigmas. Definição e caracterização dos principais paradigmas declarativos e imperativos. Programação em Lógica. Programação Funcional. Prática de programação com os principais paradigmas apresentados.

Objetivos

- **Geral:** Capacitar o aluno a compreender os principais aspectos inerentes ao projeto de linguagens de programação e suas principais construções, as características inerentes aos paradigmas de construção de linguagens de programação e a desenvolver programas utilizando o Paradigma de Programação em Lógica e o Paradigma de Programação Funcional.
- **Específicos:** • Descrever os aspectos históricos das principais linguagens de programação. • Compreender o processo de descrição formal de linguagens de programação. • Identificar as características do Paradigma de Programação Imperativo. • Descrever os principais aspectos associados à implementação de linguagens de programação. • Compreender o uso de funções matemáticas como base de programação. • Compreender o cálculo de predicados e sua utilização como base de programação. • Utilizar o paradigma de Programação Funcional. • Utilizar o paradigma de Programação em Lógica.

Conteúdo programático

- Descrever os aspectos históricos das principais linguagens de programação
- Descrição formal de linguagens de programação: sintaxe e semântica
- O paradigma imperativo
- Funçoes e Cálculo Lâmbda
- Cálculo de Predicados
- Linguagens Funcionais: Lisp, Scheme, ML, Haskel
- Linguagem em Lógica: Prolog
- Prática de programação Funcional
- Prática de programação em Lógica

Metodologia e avaliação

Metodologia:

- Aulas teóricas expositivas com apresentação de slides e discussão de textos retirados da bibliografia básica indicada. Aulas práticas realizadas a partir do desenvolvimento de exercícios de programação com a implementação de algoritmos baseados nos conceitos estudados em aula e pequenos projetos práticos ilustrando o uso das linguagens estudadas. Em algumas aulas práticas, os alunos serão acompanhados por aluno de pós-graduação em estágio de docência

Avaliação:

- As avaliações serão feitas por meio de seis trabalhos práticos, sendo três referentes ao paradigma de programação funcional (TF_i) e três referentes ao paradigma de programação em lógica (TL_i). A nota final será calculada com base na média aritmética das notas dos trabalhos

$$MF = \frac{TF_1 + TF_2 + TF_3 + TL_1 + TL_2 + TL_3}{6}$$

Cronograma

- **11ago** – Apresentação da disciplina. Introdução ao paradigma lógico.
- **18ago** – Fundamentação e conceitos básicos. Lógica de predicados.
- **25ago** – Formas normais canônicas. Resolução e unificação.
- **01set** – Fatos, regras, variáveis, conjunções e backtracking.
- **08set** – Átomo, números, variáveis, operadores, estruturas de dados, listas, recursão, concatenação e acumuladores.
- **15set** – Acumuladores, backtracking e corte.
- **22set** – Recomendações de estilo. Alguns predicados pré-definidos. Depuração de programas.
- **29set** – Leitura e escrita.
- **06out** – Leitura e escrita.
- **13out** – Gramáticas.
- **20out** – *Sem aula (participação em congresso).*
- **27out** – Exercícios sobre gramáticas.
- **03nov** – Árvores e grafos. Representações. Algoritmos em grafos.
- **10nov** – Árvores e grafos. Representações. Algoritmos em grafos.
- **17nov** – Interface do SWI-Prolog com Java, C++ e Python.
- **24nov** – Desenvolvimento e entrega dos trabalhos.
- **30nov** – Desenvolvimento e entrega dos trabalhos.
- **dez** – Desenvolvimento e entrega dos trabalhos.

Bibliografia

Básica:

- SEBESTA, Robert W. Conceitos de Linguagens de Programação. 5a. Ed. Porto Alegre: Bookman, 2003.
- BRATKO, Ivan. Prolog programming for Artificial Intelligence. Glasgow: Berkeley, 1986.
- HUDAK, Paul. The Haskell School of Expression: Learning Functional Programming through Multimedia, Cambridge University Press, New York, 2000, 416 pp, ISBN 0521644089, ISBN 0521643384.
- DE SÁ, Claudio Cesar, DA SILVA, Marcio Ferreira. Haskell: Uma Abordagem Prática, Novatec Editora Ltda., 2006, 296 pages, ISBN 85-7522-095-0.

Complementar:

- DERSHEM, H. & JIPPING, M. Programming languages: Structures and Models. Belmont: Wadsworth Publishing Company, 1990.
- GHEZZI, Carlo; JAZAYERI, Mehdi. Conceitos de Linguagens de Programação. Rio de Janeiro: Campus, 1991.
- FRIEDEMAN, Daniel P., WAND, Mitchell, HAYNES, Chistopher T. Fundamentos de linguagem de programação. São Paulo: Berkeley, 2001.ISBN: 85-7251-605-0
- STERLING, Leon, SHAPIRO, Ehud. The Art of Prolog. MIT Press. Cambridge, 1999.
- CURRY, Haskell B. Foundations of mathematical logic. New York: Dover, c1977. 407p ISBN 0486634620
- MEIRA, Silvio Romero de Lemos. Introdução a programação funcional. Campinas: UNICAMP, 1988.
- BARENDREGT, Hendrik Pieter. The lambda calculus: its syntax and semantics Rev. ed.- Amsterdam: North-Holland, 1984, ISBN 0 444 87508 5.

Paradigma lógico

- Também chamado de declarativo
- Linguagens declarativas se preocupam com os fatos e não com os procedimentos
- O conhecimento acerca do problema é descrito usando lógica simbólica

Lógica

- Aristóteles (Grécia, 384–322 a.C.)
  - Estabelecimento dos fundamentos da lógica de maneira sistemática
- Euclides (Grécia)
  - "Os elementos" (300 a.C.)
  - Método dedutivo
  - Axiomas são usados para inferir outras proposições válidas (teoremas)
- George Boole (Grã-Bretanha, 1815–1864)
  - "The Mathematical Analysis of Logic" (1847)
  - Trabalho fundamental para a construção e programação dos computadores eletrônicos iniciada cerca de 100 anos mais tarde
  - Introduz conceitos de lógica simbólica
  - A lógica entra para o campo da matemática, podendo ser representada por equações algébricas
  - Linguagem formal que permite a realização de inferências

Lógica

- Gottlöb Frege (Alemanha, 1848–1925)
  - Criação de um sistema de representação simbólica para representar formalmente a estrutura dos enunciados lógicos e suas relações
  - Decomposição funcional da estrutura de frases em função-argumento (em lugar de sujeito-predicado)
  - Operações de quantificação sobre variáveis (exemplos: "para todo x"; "existe um x")
  - Formalização de regras de demonstração, iniciando com regras elementares, bem simples, sobre cuja aplicação não houvesse dúvidas
  - Publicação da primeira versão do que hoje é o cálculo dos predicados (ou lógica dos predicados), dando início à lógica matemática moderna

- Final do século XIX - a lógica passa a ser estudada com rigor por teóricos como David Hilbert, Giuseppe Peano, George Cantor, Ernst Zermelo, Leopold Lowenheim e Thoralf Skolem.
  Sugestão de leitura:
  ftp://ftp.cle.unicamp.br/pub/arquivos/educacional/ArtGT.pdf

Lógica

- Século XX
  - Kurt Gödel e Jacques Herbrand (1930) – existem sistemas lógicos nos quais toda fórmula verdadeira pode ser provada (completude)
  - Alfred Tarski (1934) – definição formal da semântica da lógica
  - Alonzo Church, Alan Turing, Post (1936) – indecidibilidade

Lógica

Sistema lógico

- Conjunto de fórmulas que podem assumir valores verdadeiro ou falso
  - Fórmula válida – uma fórmula pode ser válida se há uma atribuição de valores verdade que a faça verdadeira
  - Tautologia – a fórmula que é sempre verdadeira
  - Teoria dos Modelos
- Conjunto de regras de inferência
  - Quando aplicada repetidamente à fórmulas verdadeiras, gera novas fórmulas verdadeiras (dedução)
  - As fórmulas geradas constituem uma prova
  - Teoria das Provas

Programação em lógica

- Formalismo lógico-computacional fundamentado em três princípios básicos:
  - Uso de **linguagem formal** para representação de conhecimento
  - Uso de **regras de inferência** para manipulação de conhecimento
  - Uso de uma **estratégia de busca** para controle de inferências
- O programador deve preocupar-se somente em descrever o problema a ser solucionado (especificação), deixando a critério da máquina a busca pela solução

Programação em lógica – linguagem formal

- Uma linguagem natural é ambígua
  - "Ana viu um homem numa montanha usando um binóculo"
  - Quem usava o binóculo?
    - ★ "Ana, usando um binóculo, viu um homem numa montanha"
    - ★ "Ana, estando numa montanha, viu um homem que usava um binóculo"
- Uma linguagem formal é precisa
  - Suas sentenças são objetos (fórmulas) com significado único, e têm sintaxe e semântica bem definidas
  - Mas também pode ser menos expressiva

Programação em lógica – regra de inferência

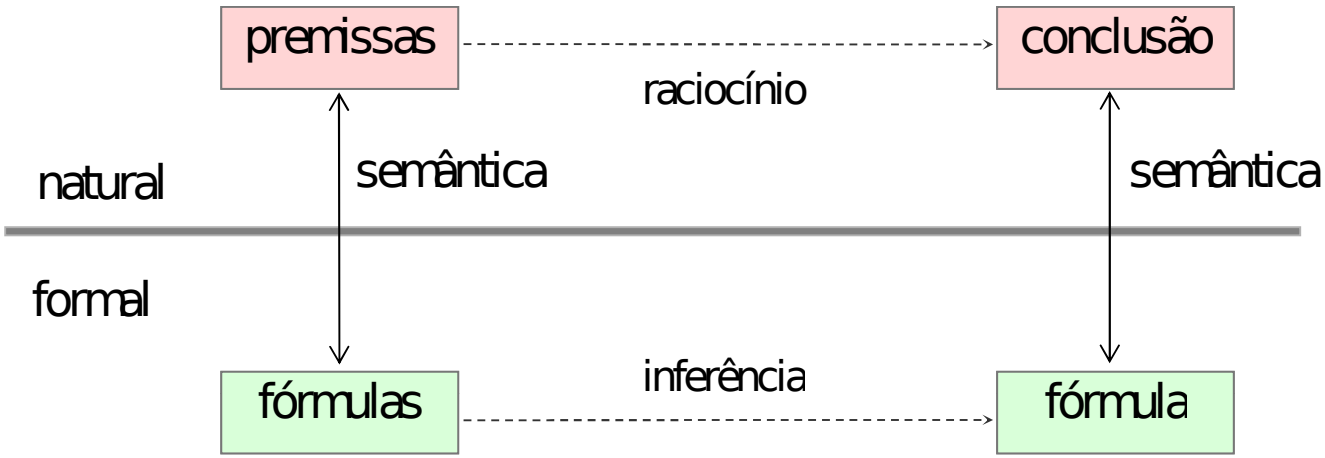
- Regra de inferência é um padrão de manipulação sintática que:
  - Permite criar novas fórmulas a partir de outras existentes
  - Em geral, simulam formas de raciocínio válidas
- Exemplo (modus ponens)

$$\alpha \rightarrow$$

Programação em lógica – estratégia de busca

- Um agente pode ter uma enorme quantidade de conhecimento armazenado
- Assim como nós, ele precisa usar apenas parte de seu conhecimento para resolver um problema
- Estratégia de busca serve para decidir que parte do conhecimento armazenado deve ser explorada em busca da solução

Programação em lógica – ideia básica

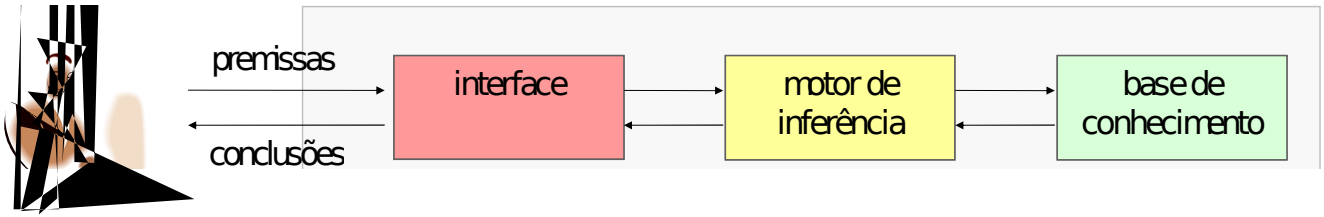


Autor: Silvio do Lago Pereira

- A idéia básica da programação em lógica é oferecer um arcabouço que permita inferir conclusões desejadas, a partir de premissas representando o conhecimento disponível, de uma forma que seja computacionalmente viável

O sistema Prolog

- Prolog é o sistema de programação em lógica mais popular que existe!



Autor: Silvio do Lago Pereira

- **Interface:** permite que o usuário entre com premissas codificadas em uma linguagem lógica e faça consultas para extrair conclusões destas premissas
- **Motor de inferência:** atualiza a base de conhecimento com premissas fornecidas pelo usuário e faz inferências para extrair informações implícitas
- **Base de conhecimento:** armazena as premissas fornecidas pelo usuário

O sistema Prolog – vantagens

- Prolog permite representar o conhecimento que um agente tem sobre seu mundo de uma forma simples e direta, em uma linguagem de alto nível, tornando os programas mais compactos, flexíveis e inteligíveis

- Prolog permite programação declarativa; em vez de especificar como o computador deve proceder para resolver um problema, precisamos apenas declarar o conhecimento que temos acerca do problema e, em seguida, consultar o sistema para que ele encontre a solução desejada

- Em outras palavras, em Prolog, basta especificar corretamente o problema que o motor de inferência se encarrega de descobrir como obter sua solução

Programação em lógica

- A descrição dos fatos é feita por meio de cláusulas
  - fatos
  - regras

- Exemplo 1 – base de dados e exemplo de teste

```
progenitor(boris, jane).
progenitor(boris, marcia).
progenitor(adelia, jane).
progenitor(jane, tiago).
avo(X,Z) :- progenitor(X,Y), progenitor(Y,Z).

?- avo(A, tiago).
Substituicao :  (X = boris) (Y = jane)
Substituicao :  (X = adelia) (Y = jane)
A = boris ;
A = adelia ;
```

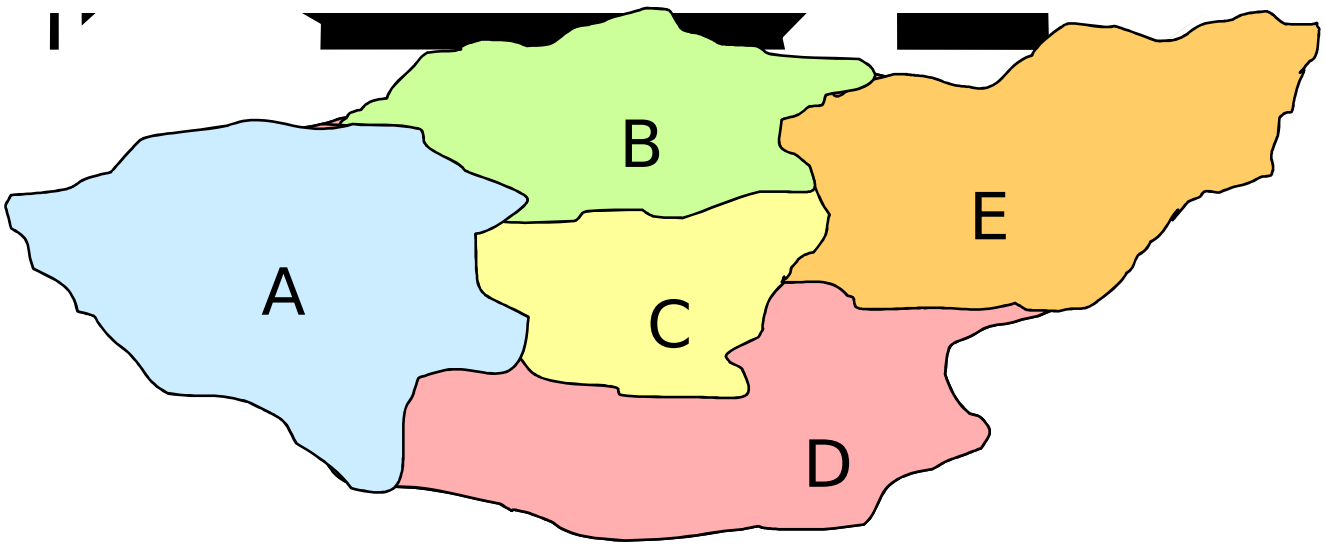
Programação em lógica

- Exemplo 2 – base de dados

```
região(planicie).
região(vales).
região(altiplano).

cidade('Cobija').
cidade('Trinidad').
cidade('La Paz').
cidade('Oruro').
cidade('Santa Cruz').
cidade('Sucre').
cidade('Potosi').
cidade('Tarija').
cidade('Cochabamba').

clima(planicie,tropical).
clima(vales,temperado).
clima(altiplano,frio).

altitude('Cobija',240).
altitude('Trinidad',250).
altitude('La Paz',3200).
altitude('Oruro',4000).
altitude('Santa Cruz',200).
altitude('Sucre',2800).
altitude('Potosi',3000).
altitude('Tarija',2500).
altitude('Cochabamba',2700).

local('Cobija',planicie).
local('Trinidad',planicie).
local('La Paz',altiplano).
local('Oruro',altiplano).
local('Santa Cruz',planicie).
local('Sucre',vales).
local('Potosi',altiplano).
local('Tarija',vales).
local('Cochabamba',vales).

tensão('Cobija',220).
tensão('Trinidad',220).
tensão('La Paz',110).
tensão('Oruro',220).
tensão('Santa Cruz',220).
tensão('Sucre',220).
tensão('Potosi',220).
tensão('Tarija',220).
tensão('Cochabamba',220).
```

Exemplo – Coloração de mapas

- **Problema:** como colorir um mapa, usando no máximo quatro cores, de modo que regiões adjacentes tenham cores distintas?



Autor: Silvio do Lago Pereira

Exemplo – Coloração de mapas

- **Solução:**
  - Primeiramente, declaramos as cores que podem ser usadas na coloração; isto é feito por meio de sentenças denominadas **fatos**. Por exemplo, o fato `cor(azul)` estabelece que azul é uma das cores disponíveis.
  - Em seguida, declaramos que a tupla `(A,B,C,D,E)`, cujos componentes correspondem às regiões do mapa, é uma coloração válida se cada um de seus componentes é uma cor e se componentes representando regiões adjacentes no mapa têm valores distintos; isto é feito por meio de uma sentença denominada **regra**.

Exemplo – Coloração de mapas

- Base:

```
% colorir.pl - colore um mapa usando no máximo quatro cores
% cores disponíveis:
  cor(azul) .
  cor(verde) .
  cor(amarelo) .
  cor(vermelho) .
% restrições para a solução:
  coloração(A,B,C,D,E) :-
      cor(A), cor(B), cor(C), cor(D), cor(E),
      A\=B, A\=C, A\=D, B\=C, B\=E, C\=D, C\=E, D\=E.
```

- Teste:

```
?  - coloração(A,B,C,D,E).
```

A = azul, B = verde, C = amarelo, D = verde, E = azul .

Exemplo – Geração de binários

- **Problema:** gerar todos os números binários compostos por três dígitos

- **Solução:**

  - Declarar que dígitos podem ser usados na composição de um número binário

  - Definir restrições sobre componentes de uma estrutura representando um número binário de três dígitos

Exemplo – Geração de binários

- Base:

```
% binário.pl
% dígitos binários
  dígito(0).
  dígito(1).
% restrições para a solução
  binário(N) :-
      N = (A,B,C),
      dígito(A),
      dígito(B),
      dígito(C).
```

- Teste:

```
?  - binário(N).
```

N = (0,0,0) ;
N = (0,0,1) ;
N = (0,1,0) ;
...

Programação em lógica

Programas convencionais	**Programas em lógica**
Processamento numérico	Processamento simbólico
Soluções algorítmicas	Soluções heurísticas
Estruturas de controle e conhecimentos integradas	Estruturas de controle e conhecimento separadas
Difícil modificação	Fácil modificação
Somente respostas totalmente corretas	Incluem respostas parcialmente corretas
Somente a melhor solução possível	Incluem todas as soluções possíveis

Programação em lógica

- Linguagens representativas do paradigma declarativo
  - Gödel
    - ★ Criada em 1992 por John Lloyd e Patricia Hill
    - ★ Uso didático
    - ★ Hill, Patricia; Lloyd, John. *The Gödel Programming Language*. Cambridge: The MIT Press, 1994. 350 p. ISBN 0-262-08229-2
  - Prolog
    - ★ Acrônimo de *PROgramming in LOGic*
    - ★ Desenvolvida na década de 70 por Alain Colmerauer e Phillipe Roussel (Artificial Intelligence Group - Université Aix-Marseille) e Robert Kowalski (Departament of Artificial Intelligence – University of Edimburgh)
    - ★ 1972 - Desenvolvimento do 1° interpretador PROLOG em Marseille
    - ★ Meados de 70 – fim da cooperação e surgimento de 2 dialetos distintos

Linguagem Prolog

- 1981 – Projeto Fifth Generation Computing Systems (Japão)
  - Desenvolvimento de máquinas inteligentes
  - Atenção voltada para a Inteligência Artificial e a Programação em Lógica
- Surgimento de variações de PROLOG com diferentes sintaxes
- Linguagem utilizada no curso
  - SWI-Prolog – Página oficial: http://www.swi-prolog.org/

Exercícios

1. Instalação
   - Logar na máquina, utilizando Windows neste primeiro momento (alternativamente, logar na máquina "alunos", que já possui o Prolog instalado, ou utilizar a máquina virtual Linux à disposição).
   - Baixar a versão "portable" do SWI-Prolog: http://portableapps.com/apps/development/swi-prolog_portable
   - Executar e instalar em uma pasta com acesso livre (pode ser em um pen-drive)

Exercícios

3. Teste
   - Criar um novo arquivo com extensão `.pl` com o **exemplo 2** desta apresentação
   - Carregar tal arquivo (ou base de dados), por meio de 'consult', no interpretador Prolog
   - Criar perguntas simples e variadas para obtenção de dados sobre as cidades