

Programação em Lógica

Prof. A. G. Silva

18 de agosto 2016

Introdução (I)

- Paradigma de **programação lógico**
- **Linguagens de programação lógica** ou **linguagens declarativas**
 - ▶ Programas declarativos em vez de procedurais
 - ▶ Especificações dos resultados desejados são expressas (em vez de os procedimentos detalhados para produzi-los)
 - ▶ Sintaxe notavelmente diferente da sintaxe das linguagens funcionais e imperativas
 - ▶ Expressão de programas em uma forma de **lógica simbólica**
 - ▶ Processo de inferência lógico para produzir resultados

Introdução (II)

- A lógica possui uma longa história, de mais de 23 séculos, que remonta aos antigos filósofos gregos, principalmente Aristóteles, que estabeleceu os seus fundamentos de maneira sistemática
- Lógica Booleana – Boole (1847) traz a lógica para o campo da matemática
- Lógica de Predicados – Fregue (1879) – lógica moderna
- A lógica lida com dois conceitos fundamentais
 - ▶ Verdade (teoria de modelos)
 - ▶ Prova (teoria das provas)

Introdução (III)

- Teoria dos modelos

- ▶ Trata da validade das fórmulas lógicas - é possível que uma dada fórmula apresente o valor de verdade verdadeiro (uma proposição é uma asserção declarativa, ou seja, afirma ou nega um fato, e tem um valor de verdade, que pode ser verdadeiro ou falso.)
 - ★ Tautologia – sempre verdade
 - ★ Fórmula válida – quando há pelo menos uma interpretação que torne a fórmula verdadeira

Introdução (IV)

- Teoria das provas
 - ▶ Dado um conjunto de axiomas e alguma regra de inferência, verificar se uma fórmula pode ser deduzida a partir dos axiomas
 - ▶ A sequência de fórmulas geradas constitui uma prova

Cálculo de predicados – Introdução (I)

- Base da programação lógica: **lógica formal**
- Consiste em objetos e seus relacionamentos
- Lógica formal como método para descrever **proposições**
- Proposições formalmente descritas podem ser verificadas em relação a sua validade

Cálculo de predicados – Introdução (II)

- A **lógica simbólica** pode ser usada para as três necessidades básicas da lógica formal:
 - ▶ Expressar proposições
 - ▶ Expressar os relacionamentos entre proposições
 - ▶ Descrever como novas proposições podem ser inferidas a partir de outras proposições tidas como verdadeiras
- Forte relacionamento entre lógica formal e matemática
- Muito da matemática pode ser pensado em termos de lógica

Cálculo de predicados – Introdução (III)

- Os axiomas fundamentais de teoria dos números e dos conjuntos são proposições iniciais, tidos como verdadeiros
- Teoremas são as proposições adicionais que podem ser inferidas
- A forma particular de lógica simbólica na programação lógica é chamada de **cálculo de predicados de primeira ordem** (ou simplesmente cálculo de predicados)

Cálculo de predicados – Proposições (I)

- Os **objetos** em proposições são representados por termos simples, **constantes** ou **variáveis**
- Uma **constante** é um símbolo que representa um objeto
- Uma **variável** é um símbolo capaz de representar objetos diferentes em momentos diferentes
- Uma **proposição atômica** é formada por um símbolo de predicado seguido por uma lista de termos entre parênteses ou termos compostos. Exemplos:
 - irmão(ricardo, joão)
 - casado(pai(ricardo), mãe(joão))

Cálculo de predicados – Proposições (II)

- Um **termo composto** é um elemento de uma relação matemática (aparência de uma notação de função matemática)
- Um termo composto tem duas partes:
 - ▶ Um **functor**, símbolo da função que nomeia a relação
 - ▶ Uma lista ordenada de parâmetros
- Um termo composto com um único parâmetro é um 1-tupla, com dois é uma 2-tupla, e assim por diante

Cálculo de predicados – Proposições (III)

- Exemplos de proposições:

`man(jake)`

`like(bob, steak)`

que dizem que `{jake}` é uma 1-tupla na relação `man`, e que `{bob, steak}` é uma 2-tupla na relação `like`.

- ▶ Termos simples – `man`, `jake`, `like`, `bob` e `steak` – são constantes
- ▶ Sem semântica intrínseca (pode significar que `bob` gosta de `steak`, ou que `steak` gosta de `bob`, ou que `bob` é de alguma forma similar a um `steak`)

Cálculo de predicados – Proposições (IV)

- Dois modos de definição de proposições:
 - Definidas como verdadeiras: **fatos**
 - No qual a verdade é algo que deve ser determinado: **consultas**
- Proposições compostas** têm duas ou mais proposições atômicas conectadas por conectores lógicos ou operadores:

<i>Nome</i>	<i>Símbolo</i>	<i>Exemplo</i>	<i>Significado</i>
negação	\neg	$\neg a$	não a
conjunção	\wedge	$a \wedge b$	a e b
disjunção	\vee	$a \vee b$	a ou b
equivalência	\equiv	$a \equiv b$	a é equivalente a b
implicação	\rightarrow	$a \rightarrow b$	a implica em b
	\leftarrow	$a \leftarrow b$	b implica em a

Cálculo de predicados – Proposições (V)

- Semântica (significado das sentenças)
 - ▶ **Interpretação:** associação entre proposições e valores-verdade (V ou F). Uma fórmula contendo n proposições admite 2^n interpretações distintas
 - ▶ **Tabela-verdade:** avalia uma fórmula em cada interpretação possível

a	b	$\neg a$	$a \wedge b$	$a \vee b$	$a \rightarrow b$
F	F	V	F	F	V
F	V	V	F	V	V
V	F	F	F	V	F
V	V	F	V	V	V

Cálculo de predicados – Proposições (VI)

- Tipos de fórmulas

- ▶ **Válida (tautológica):** é verdadeira em toda interpretação
- ▶ **Satisfatível (contingente):** é verdadeira em alguma interpretação
- ▶ **Insatisfatível (contraditória):** é verdadeira em nenhuma interpretação

- Exercício – Classifique cada fórmula como válida, satisfatível ou insatisfatível

- ▶ $a \vee \neg a$
- ▶ $a \wedge \neg a$
- ▶ $(a \rightarrow b) \wedge a \rightarrow b$
- ▶ $(a \rightarrow b) \wedge b \rightarrow a$
- ▶ $(a \rightarrow b) \wedge \neg b \rightarrow \neg a$
- ▶ $(a \rightarrow b) \wedge a \wedge \neg b$
- ▶ $a \wedge (a \rightarrow b) \wedge (b \rightarrow c)$

Cálculo de predicados – Proposições (VII)

- Outros exemplos de proposições compostas:

- ▶ $a \wedge b \rightarrow c$

- ▶ $a \wedge \neg b \rightarrow d$ ou $(a \wedge (\neg b)) \rightarrow d$

- ▶ $\neg \text{irmão}(\text{pernaEsquerda}(\text{Ricardo}), \text{João})$

- ▶ $\text{irmão}(\text{Ricardo}, \text{João}) \wedge \text{irmão}(\text{João}, \text{Ricardo})$

- ▶ $\text{rei}(\text{Ricardo}) \vee \text{rei}(\text{João})$

- ▶ $\neg \text{rei}(\text{Ricardo}) \rightarrow \text{rei}(\text{João})$

- O operador \neg tem precedência mais alta
- Os operadores \wedge , \vee e \equiv têm precedência mais alta do que \rightarrow e \leftarrow

Cálculo de predicados – Proposições (VIII)

- Variáveis podem ser introduzidas por símbolos especiais chamados de **quantificadores**
- O cálculo de predicados inclui dois quantificadores, onde X é uma variável e P é uma proposição:

<i>Nome</i>	<i>Exemplo</i>	<i>Significado</i>
universal	$\forall X.P$	Para todo X , P é verdadeiro
existencial	$\exists X.P$	Existe um valor de X tal que P é verdadeiro

- Exemplos:
 $\forall X.(\text{woman}(X) \rightarrow \text{human}(X))$
 $\exists X.(\text{mother}(\text{mary}, X) \wedge \text{male}(X))$
- Os quantificadores universal e existencial têm precedência mais alta do que a de qualquer um dos operadores

Cálculo de predicados – Proposições (IX)

- Quantificadores aninhados são usados em proposições (sentenças) complexas compostas
- Exemplos:
 - ▶ *Irmãos são parentes*
 $\forall x \forall y. \text{irmão}(x, y) \rightarrow \text{parente}(x, y)$
 - ▶ *Parentesco é uma relação simétrica*
 $\forall x \forall y. \text{parente}(x, y) \equiv \text{parente}(y, x)$
 - ▶ *Todo mundo ama alguém*
 $\forall x \exists y. \text{ama}(x, y)$
 - ▶ *Existe alguém que é amado por todo mundo*
 $\exists y \forall x. \text{ama}(x, y)$

Cálculo de predicados – Proposições (X)

- Quantificadores aninhados por meio de negação

- Exemplos:

- ▶ *Todo mundo detesta cenouras* \equiv *não existe alguém que goste de cenouras*

$$\forall x. \neg \text{gosta}(x, \text{cenouras}) \equiv \neg \exists x. \text{gosta}(x, \text{cenouras})$$

- ▶ *Todo mundo gosta de sorvete* \equiv *não existe alguém que não goste de sorvete*

$$\forall x. \text{gosta}(x, \text{sorvete}) \equiv \neg \exists x. \neg \text{gosta}(x, \text{sorvete})$$

Cálculo de predicados – Proposições (XI)

- Exemplos de parentescos:

- ▶ A mãe de alguém é o ancestral feminino de alguém
 $\forall m \forall c. \text{mãe}(c) = m \equiv \text{feminino}(m) \wedge \text{ancestral}(m, c)$

- ▶ O marido de alguém é o cônjuge masculino de alguém
 $\forall w \forall h. \text{marido}(h, w) \equiv \text{masculino}(h) \wedge \text{cônjuge}(h, w)$

- ▶ Masculino e feminino são categorias disjuntas
 $\forall x. \text{masculino}(x) \equiv \neg \text{feminino}(x)$

- ▶ Ancestral e descendente são relações inversas
 $\forall p \forall c. \text{ancestral}(p, c) \equiv \text{descendente}(c, p)$

- ▶ Avô é um pai do pai de alguém
 $\forall g \forall c. \text{avô}(g, c) \equiv \exists p. \text{pai}(g, p) \wedge \text{pai}(p, c)$

- ▶ Um parente é outro descendente dos ancestrais de alguém
 $\forall x \forall y. \text{parente}(x, y) \equiv x \neq y \wedge \exists p. \text{ancestral}(p, x) \wedge \text{ancestral}(p, y)$

Cálculo de predicados – Sintaxe (I)

- Formalmente, uma linguagem lógica de primeira ordem – notada $L(\mathbf{P}, \mathbf{F}, \mathbf{C}, \mathbf{V})$ – é determinada pela especificação dos seguintes conjuntos de símbolos:
 - ▶ Um conjunto \mathbf{P} de símbolos de predicado
 - ▶ Um conjunto \mathbf{F} de símbolos de função
 - ▶ Um conjunto \mathbf{C} de símbolos de constante
 - ▶ Um conjunto \mathbf{V} de símbolos de variável
- A cada símbolo de predicado e de função é associada uma **aridade**, isto é, o número de argumentos do predicado e da função. Os símbolos de predicado com aridade zero são chamados **símbolos proposicionais**

Cálculo de predicados – Sintaxe (II)

$\langle \text{fórmula} \rangle ::= \langle \text{fórmula-atômica} \rangle \mid \neg(\langle \text{fórmula} \rangle) \mid$
 $(\langle \text{fórmula} \rangle_1 \wedge \langle \text{fórmula} \rangle_2) \mid$
 $(\langle \text{fórmula} \rangle_1 \vee \langle \text{fórmula} \rangle_2) \mid$
 $(\langle \text{fórmula} \rangle_1 \rightarrow \langle \text{fórmula} \rangle_2) \mid$
 $(\forall \langle \text{variável} \rangle. (\langle \text{fórmula} \rangle)) \mid$
 $(\exists \langle \text{variável} \rangle. (\langle \text{fórmula} \rangle)) \mid$

$\langle \text{fórmula-atômica} \rangle ::= V \mid F \mid$
 $\langle \text{predicado} \rangle(\langle \text{termo} \rangle_1, \dots, \langle \text{termo} \rangle_n)$

$\langle \text{termo} \rangle ::= \langle \text{variável} \rangle \mid \langle \text{constante} \rangle \mid$
 $\langle \text{função} \rangle(\langle \text{termo} \rangle_1, \dots, \langle \text{termo} \rangle_n)$

$\langle \text{constante} \rangle ::= a, b, c, d$ e outras palavras iniciadas por minúsculas

$\langle \text{variável} \rangle ::= x, y, z, w$ com ou sem índices

$\langle \text{função} \rangle ::= f, g, h$ e outras palavras iniciadas por minúsculas

$\langle \text{predicado} \rangle ::= P, Q, R$ e outras palavras iniciadas por maiúsculas

Cálculo de predicados – Sintaxe (III)

- Exemplos:

$$Avo(a, c) \leftarrow Pai(a, b) \wedge Pai(b, c)$$

$$\neg(Ama(brutus, cesar))$$

$$Q(b) \leftarrow \forall x.((P(x) \wedge \neg(P(a))))$$

$$\forall x.\exists y.(Gosta(y, x))$$

$$Ama(amelia, z)$$

$$F$$

Cálculo de predicados – Descrevendo fatos em lógica (I)

- Marcos era um homem

Homem(marcos)

- Marcos nasceu em Pompéia

Pompeano(marcos)

- Todos os que nasceram em Pompéia eram romanos

Romano(x) ← ∀x.Pompeano(x)

- César era um soberano

Soberano(cesar)

Cálculo de predicados – Descrevendo fatos em lógica (II)

- Todos os romanos eram leais a César ou então odiavam-no

$$\forall x. \text{Romano}(x) \leftarrow \text{LealA}(x, \text{cesar}) \vee \text{Odeia}(x, \text{cesar})$$

- Todo mundo é leal a alguém

$$\forall x. \exists y. \text{LealA}(x, y)$$

- As pessoas só tentam assassinar soberanos aos quais não são leais

$$\neg \text{LealA}(x, y) \leftarrow \forall x. \forall y. \text{Pessoas}(x) \wedge \text{Soberano}(y) \wedge \text{TentaAssassinar}(x, y)$$

- Marcos tentou assassinar César

$$\text{TentaAssassinar}(\text{marcos}, \text{cesar})$$

Exercícios

Descreva os seguintes fatos em lógica

- ① João gosta de todo o tipo de comida
- ② Maças são comida
- ③ Frango é comida
- ④ Qualquer coisa que alguém coma e não cause sua morte é comida
- ⑤ Paulo come amendoim e ainda está vivo
- ⑥ Susana come tudo o que Paulo come
- ⑦ Carlos só gosta de cursos fáceis
- ⑧ O curso de ciências é difícil
- ⑨ Todos os cursos do departamento de prendas domésticas são fáceis
- ⑩ BK32 é um curso de prendas domésticas

Cálculo de predicados – Representação de conhecimento (I)

- Conhecimento pode ser representado de duas formas:
 - ▶ **explícita:** por meio da formalização de sentenças
 - ▶ **implícita:** por meio de consequência lógica (fatos derivados das sentenças)
- Passos para formalização de sentenças
 - ▶ Identificamos as palavras da sentença que correspondem a conectivos
 - ▶ Identificamos as partes da sentença que correspondem a proposições atômicas e associamos a cada uma delas um símbolo proposicional
 - ▶ Escrevemos a fórmula correspondente à sentença, substituindo suas proposições atômicas pelos respectivos símbolos proposicionais e seus conectivos lógicos pelos respectivos símbolos conectivos
 - ▶ Exemplo:
 - ★ Está chovendo.
 - ★ **Se** está chovendo, **então** a rua está molhada.
 - ★ **Se** a rua está molhada, **então** a rua está escorregadia.

Cálculo de predicados – Representação de conhecimento (II)

- Exemplo:

- ▶ Está chovendo.
- ▶ **Se** está chovendo, **então** a rua está molhada.
- ▶ **Se** a rua está molhada, **então** a rua está escorregadia.

- Vocabulário:

- ▶ a : “está chovendo”
- ▶ b : “a rua está molhada”
- ▶ c : “a rua está escorregadia”

- Formalização (base de conhecimento):

- ▶ $\Delta = \{ a, \quad a \rightarrow b, \quad b \rightarrow c \}$

Exercícios

Formalize as sentenças usando sintaxe da lógica proposicional

- 1 Se Ana é alta e magra, então ela é elegante.
- 2 Se Beto é rico, então ele não precisa de empréstimos.
- 3 Se Caio ama a natureza, então ele ama as plantas e os animais.
- 4 Se Denis jogar na loteria, então ele ficará rico ou desiludido.
- 5 Se faz frio ou chove, então Eva fica em casa e vê TV.

Cálculo de predicados – Forma clausal (I)

- Cálculo de predicados é a base para linguagens de programação lógica
- Lógicas são melhores em sua forma simples: redundância deve ser minimizada
- **Problema:** existem muitas maneiras de definir proposições com o mesmo significado (quantidade grande de redundância)
- Ok para lógicos, mas é um problema sério em sistema automatizado
- Uma forma padrão para proposições é desejável
- A **forma clausal**, relativamente simples de proposições, é uma das formas padrão

Cálculo de predicados – Forma clausal (II)

- Sem perda de generalidade, todas as proposições podem ser expressas em forma clausal
- Uma proposição em forma clausal tem a seguinte sintaxe geral:

$$B_1 \vee B_2 \vee \cdots \vee B_n \leftarrow A_1 \wedge A_2 \wedge \cdots \wedge A_m$$

na qual os A s e B s são termos. O significado é: se todos os A s são verdadeiros, então ao menos um B é verdadeiro

- Características:
 - ▶ Quantificadores existenciais não são necessários
 - ▶ Quantificadores universais são implícitos no uso de variáveis nas proposições atômicas
 - ▶ Nenhum operador, além da conjunção (do lado direito) e da disjunção (do lado esquerdo), é necessário

Cálculo de predicados – Forma clausal (III)

- Todas as proposições de cálculo de predicados podem ser algoritmicamente convertidas para a forma clausal
- Nilsson (1971) prova que isso pode ser feito, e mostra um algoritmo de conversão simples
- Lado direito: **antecedente**; Lado esquerdo: **consequente**
- Exemplos:

$\text{likes}(\text{bob}, \text{trout}) \leftarrow \text{likes}(\text{bob}, \text{fish}) \wedge \text{fish}(\text{trout})$

$\text{father}(\text{louis}, \text{al}) \vee \text{father}(\text{louis}, \text{violet}) \leftarrow$
 $\text{father}(\text{al}, \text{bob}) \wedge \text{mother}(\text{violet}, \text{bob}) \wedge \text{grandfather}(\text{louis}, \text{bob})$

Cálculo de predicados – Construção da base de conhecimento

- 1 Identificar a tarefa
- 2 Agregar conhecimento relevante
- 3 Definir um vocabulário de predicados, funções e constantes
- 4 Codificar o conhecimento geral sobre o domínio
- 5 Codificar uma descrição da instância específica do problema
- 6 Formular consultas ao procedimento de inferência e obter respostas
- 7 Depurar a base de conhecimento

Programação em Prolog (I)

- O Prolog é uma linguagem de programação baseada em lógica de primeira ordem
- Não é padronizada
- Geralmente é interpretado, mas pode ser compilado
- Algumas implementações: SICStus Prolog, Borland Turbo Prolog, GNU Prolog, **SWI-Prolog**, ...
 - ▶ Acessar máquina `alunos.inf.ufsc.br` vias SSH pela rede interna UFSC (ou remotamente via VPN)
 - ▶ O usuário é o número de matrícula (8 dígitos para a graduação)

Programação em Prolog (II)

- Prolog lida com:
 - ▶ Objetos sobre os quais queremos raciocinar (não tem o mesmo sentido que em orientação a objetos – não há métodos ou herança)
 - ▶ Relações entre objetos
 - ▶ Tipo chamado termo que engloba todos os dados e também programas nesta linguagem
- Um programa em Prolog é composto por:
 - ▶ Fatos sobre certos objetos
 - ▶ Regras de inferência
 - ▶ Perguntas sobre os objetos

Programação em Prolog (III)

- Dizemos a Prolog certos fatos e regras
- Em seguida, fazemos perguntas sobre estes fatos e regras
- Exemplo:
 - ▶ Podemos informar sobre irmãs e depois perguntar se Maria e Joana são irmãs
 - ▶ Prolog responderá sim ou não em função do que lhe dissermos
- Prolog faz muito mais do que responder sim ou não
 - ▶ Permite usar o computador como um arcabouço de fatos e regras
 - ▶ Proporciona meios de fazer inferências, indo de um fato a outro, e achando os valores das variáveis que podem levar a conclusões lógicas

Fatos

- Eis alguns exemplos de como se informam fatos a Prolog:

```
gosta(pedro, maria).  
gosta(maria, pedro).  
valioso(ouro).  
mulher(jane).  
possui(jane, ouro).  
pai(pedro, maria).  
entrega(romeu, livro, maria).
```

Observações

- Nomes de relações e objetos iniciam-se com letra minúscula
- O nome da relação vem primeiro, depois vem a lista de objetos separados por vírgula e envolvida em parênteses
- O ponto final é obrigatório ao final do fato
- Terminologia
 - ▶ Relações são predicados e os objetos a que se referem são argumentos
 - ▶ Chamaremos de banco de dados à coleção de fatos e regras que damos a Prolog para resolver um problema

Perguntas (I)

- Uma pergunta em Prolog tem a forma:
`?- possui(maria,livro).`
- Prolog tenta unificar o fato da pergunta com os fatos do banco de dados
- Dois fatos se unificam se têm o mesmo predicado e os mesmos argumentos na mesma ordem
- Se Prolog encontrar um fato que unifica com a pergunta, vai responder “sim”; caso contrário, responderá “não”

Perguntas (II)

- A resposta “não” em Prolog não significa necessariamente que o fato não é verdadeiro, mas simplesmente que Prolog não consegue provar o fato a partir de seu banco de dados

- Considere o banco de dados:

```
humano(socrates).  
humano(aristoteles).  
ateniense(socrates).
```

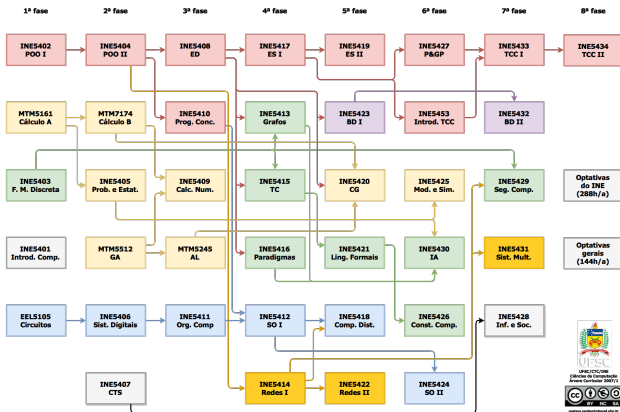
e a pergunta:

```
?- ateniense(aristoteles).
```

- Embora seja verdade que Aristóteles tenha sido ateniense, não se pode provar isto a partir dos fatos dados.

Trabalho 1 – Parte A (entrega pelo Moodle)

- Crie uma base de conhecimento, em Prolog, para a representação da matriz curricular, por fase, e a cadeia de pré-requisitos de um curso da UFSC (verifique seu curso selecionado no [enunciado](#) do Moodle). Exemplo da Ciências da Computação:



Referências

- SEBESTA, Robert W. *Conceitos de Linguagens de Programação*. 5a. Ed. Porto Alegre: Bookman, 2003.
- NILSSON, N. J. *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill, Nova York, Estados Unidos, 1971.
- RUSSELL, S. and NORVIG, P. *Artificial Intelligence: a Modern Approach*, 3rd Edition, Prentice-Hall, 2009 (Cap. 8 e 9).
- Notas de aula do Prof. Silvio L. Pereira.