Paradigmas de Linguagens de Programação

Programação Lógica

Introdução

- Programas lógicos são **declarativos**, e não **procedurais**.
- As **especificações dos resultados desejados** são expressas, não os procedimentos para produzi-los.
- Nas linguagens de programação lógica, os programas são **coleções de fatos e regras**.
- A execução ocorre por meio de **consultas (perguntas)** ao sistema, que responde **consultando fatos e regras**.

"Quem é o pai de Maria?"

- o sistema procura fatos e regras que satisfaçam essa consulta.

Introdução

- A programação lógica difere profundamente das linguagens **imperativas** e **funcionais**:
  - **Imperativas:** descrevem *como fazer* (sequência de comandos).
  - **Funcionais:** descrevem *o que calcular* (funções e expressões).
  - **Lógicas:** descrevem *o que é verdade* (fatos e relações).

Introdução

- A **lógica simbólica** é a base formal das linguagens de programação lógica.
- Programas escritos nesse paradigma são chamados de **programas lógicos**.
- As linguagens baseadas nessa forma são chamadas de **linguagens declarativas**.

Introdução

- A **sintaxe** das linguagens lógicas é bem diferente da de linguagens funcionais ou imperativas.
- A **semântica** também é distinta:
  - Não se baseia em execução sequencial nem em estados de memória.
  - O significado de um programa está **nas afirmações lógicas** que ele representa.

Cálculo de Predicados: Introdução

- Antes de estudar **programação lógica**, é importante entender sua **base teórica**: a **lógica formal**.
- A lógica formal fornece um **método para descrever proposições**, permitindo **verificar sua validade**.
- O objetivo é entender como representar e raciocinar sobre o que é verdadeiro.

Cálculo de Predicados: Introdução

- Uma **proposição** é uma sentença lógica que **pode ser verdadeira ou falsa**.
- Ela consiste em **objetos** e **relacionamentos entre objetos**.
- A **lógica simbólica** foi desenvolvida para:
  - Expressar proposições;
  - Representar relações entre proposições;
  - Descrever como novas proposições podem ser inferidas de outras já conhecidas.
- Exemplo:

  “Socrates é homem” e “Todos os homens são mortais”

  inferimos “Socrates é mortal”.

Cálculo de Predicados: Introdução

- A **lógica formal** e a **matemática** estão intimamente relacionadas.
- Grande parte da matemática pode ser pensada **em termos de lógica**:
  - **Axiomas fundamentais** formam um conjunto inicial de proposições (verdades assumidas).
  - **Teoremas** são proposições **inferidas logicamente** a partir desses axiomas.
- Assim como na matemática, a programação lógica usa **inferência** para derivar novas verdades.

Cálculo de Predicados: Introdução

- A forma de lógica simbólica usada em **programação lógica** é o **cálculo de predicados de primeira ordem**.
- Também chamado simplesmente de **cálculo de predicados**.
- Ele fornece a **base teórica** para linguagens como o **Prolog**.
- Permite representar:
  - **Objetos**,
  - **Relacionamentos** entre objetos,
  - **Regras lógicas** que permitem deduzir novos fatos.

Cálculo de Predicados: Introdução

- Em programação lógica, os **objetos** são representados por **termos simples**, que podem ser:
  - **Constantes:** símbolos que representam objetos fixos.
  - **Variáveis:** símbolos que podem representar diferentes objetos em momentos distinto

Cálculo de Predicados: Introdução

- As proposições mais simples são chamadas de **proposições atômicas**.
- Elas consistem em **termos compostos**, que representam **relações matemáticas**.
- Um **termo composto** é formado por:
  - Um **functor** (nome da relação ou função);
  - Uma **lista ordenada de parâmetros** (elementos da relação).
- Exemplo:

man(jake)

like(bob, steak)

“Jake é homem.” “Bob gosta de bife.”

Cálculo de Predicados: Introdução

- Um termo composto com **um parâmetro** é uma **1-tupla**.
- Com **dois parâmetros**, é uma **2-tupla**, e assim por diante.

- Se adicionarmos mais proposições:

  man(fred)

- então a relação *man* conterá {*jake*, *fred*}.

- Os símbolos *man*, *jake*, *bob* e *steak* são **constantes**.
- Essas proposições **não têm semântica intrínseca,** o significado é dado pela interpretação.

Cálculo de Predicados: Introdução

- Proposições podem ser de dois tipos:
  - **Fatos:** declarações assumidas como verdadeiras.
  - **Consultas:** perguntas feitas ao sistema para verificar se algo é verdadeiro.
- O programa lógico é um **conjunto de fatos e regras**, e a execução é **consultar** esses fatos.

Cálculo de Predicados: Introdução

- **Proposições compostas** combinam duas ou mais proposições atômicas.
- São conectadas por **operadores lógicos**, assim como em linguagens imperativas.

Nome	*Símbolo*	*Exemplo*	*Significado*
negação	$\neg$	$\neg a$	não a
conjunção	$\cap$	$a \cap b$	a e b
disjunção	$\cup$	$a \cup b$	a ou b
equivalência	$\equiv$	$a \equiv b$	a é equivalente a b
implicação	$\supset$	$a \supset b$	a implica b
	$\subset$	$a \subset b$	b implica a

$$a \cap b \supset c$$

$$a \cap \neg b \supset d$$

Cálculo de Predicados: Introdução

- O operador ¬ (negação) tem **precedência mais alta**.
- ∩ e ∪ têm precedência mais alta que ⊃ e ⊂.
- Assim, a expressão:

$$a \cap \neg b \supset d$$

- é equivalente a

$$(a \cap (\neg b)) \supset d$$

"Se a for verdadeiro e b for falso, então d é verdadeiro."

Cálculo de Predicados: Introdução

- Em lógica de predicados, variáveis podem aparecer em proposições, mas **devem ser ligadas a um quantificador**.
- ***X*** é uma variável e o ***P*** é uma proposição

Nome	*Exemplo*	*Significado*
universal	$\forall X.P$	Para todo X, P é verdadeiro.
Existencial	$\exists X.P$	Existe um valor de X tal que P é verdadeiro.

Cálculo de Predicados: Introdução

$$\forall X.\ (\text{mulher}(X) \supset \text{humano}(X))$$
$$\exists X.\ (\text{mãe}(\text{mary}, X) \cap \text{homem}(X))$$

- A primeira proposição diz:

  "Para qualquer X, se X é mulher, então X é humano."

- A segunda:

  "Existe algum X tal que Mary é mãe de X e X é homem."

- Os quantificadores **têm precedência mais alta** que qualquer outro operador lógico.

Cálculo de Predicados: Introdução

- O cálculo de predicados fornece a base para linguagens como Prolog.
- Ele permite representar:
  - Objetos (constantes e variáveis);
  - Relações entre objetos (predicados);
  - Regras lógicas para deduzir novos fatos.
- A execução de um programa lógico é feita por inferência:
  - Dado um conjunto de fatos e regras, o sistema tenta provar se uma proposição (consulta) é verdadeira.

Cálculo de Predicados: Introdução

Forma clausal

- O cálculo de predicados é a **base da programação lógica**.
- Assim como em outras linguagens, **formas mais simples** são melhores, a **redundância deve ser minimizada**.
- Há muitas formas de expressar proposições com o **mesmo significado**, o que causa redundância.
- Para resolver isso, usa-se uma **forma padrão**: a **forma clausal**.
- A forma clausal é uma **forma simplificada de proposições** usada por sistemas automatizados de inferência (como Prolog).

Cálculo de Predicados: Introdução

Forma clausal

- Na lógica, **uma mesma proposição pode ser escrita de várias formas**, todas com o mesmo significado.
- Exemplo:

  “Se chover, então a rua ficará molhada.”

  “A rua não fica molhada a menos que chova.”

  - Ambas expressam a **mesma relação lógica**, mas em **formatos diferentes**.
- Para que sistemas como Prolog consigam inferir, é preciso que todas as proposições estejam em **um formato único e padronizado**.
- Essa padronização é a **forma clausal**:
  - uma maneira **simples, uniforme e sem ambiguidades** de representar proposições.

Cálculo de Predicados: Introdução

Forma clausal

- Uma **proposição em forma clausal** tem a seguinte estrutura geral:

$$B_1 \cup B_2 \cup \ldots \cup B_n \subset A_1 \cap A_2 \cap \ldots \cap A_m$$

- Onde:
  - Os **A's** e **B's** são termos (proposições atômicas).
  - O significado é:

"Se todos os **A's** são verdadeiros, então pelo menos um B é verdadeiro."

Cálculo de Predicados: Introdução

Forma clausal

- Em uma proposição clausal:
  - O **lado direito** (após o símbolo ⊂) é o **antecedente**.
  - O **lado esquerdo** é o **consequente**.
  - Somente **conjunção (∩)** e **disjunção (∪)** são usadas, **nessa ordem**:
    - Disjunções à esquerda, conjunções à direita.
- O consequente é o que **se torna verdadeiro** se o antecedente for verdadeiro.
- Exemplo:

$$\text{likes(bob, trout)} \subset \text{likes(bob, fish)} \cap \text{fish(trout)}$$

“Se Bob gosta de peixe e se truta é um peixe, então Bob gosta de truta.”

Cálculo de Predicados: Introdução

Forma clausal

- Exemplo:

$$\text{father(louis, al)} \cup \text{father(louis, violet)} \subset$$
$$\text{father(al, bob)} \cap \text{mother(violet, bob)} \cap \text{grandfather(louis, bob)}$$

“Se Al é pai de Bob, Violet é mãe de Bob e Louis é avô de Bob, então Louis é pai de Al **ou** pai de Violet. “

Panorama da Programação Lógica

- Linguagens de programação lógica são chamadas de **linguagens declarativas**, pois seus programas consistem em **declarações de fatos e regras**, e não em instruções de controle.
- As declarações são **proposições em lógica simbólica**, e não comandos de atribuição.
- A principal característica dessas linguagens é a **semântica declarativa**, segundo a qual o **significado de uma proposição** é determinado unicamente por sua forma, sem depender do contexto ou da sequência de execução.
- Essa simplicidade torna a semântica declarativa **mais clara** que a das linguagens imperativas.

Panorama da Programação Lógica

- Em linguagens imperativas ou funcionais, o programador descreve **como** o resultado será obtido.
- Em linguagens lógicas, o programador descreve **o que** deve ser verdadeiro.
- Assim, programas em linguagens lógicas são **não procedurais,** descrevem o problema, e não o processo para resolvê-lo.
- O sistema lógico é responsável por determinar **como** o resultado será computado, aplicando inferência e resolução.

Prolog

- **Prolog** vem de *“PROgramming in LOGic”*: Programação em Lógica.
- É a **primeira linguagem de programação lógica** amplamente usada.
- Baseada em **lógica simbólica** e **inferência automática** de fatos e regras.
- Objetivo inicial: permitir que o computador **“raciocinasse” logicamente** a partir de um conjunto de proposições.

Prolog

- Criada no **início dos anos 1970** por **Alain Colmerauer** e **Philippe Roussel**, na **Universidade de Aix-Marseille (França)**, com colaboração de **Robert Kowalski** da **Universidade de Edimburgo (Escócia)**.
- Colmerauer e Roussel estudavam **processamento de linguagem natural**, enquanto Kowalski focava em **lógica matemática e prova de teoremas**.
- A colaboração entre os grupos de Marselha e Edimburgo originou o **Prolog**.

Prolog

- Após 1975, o desenvolvimento seguiu de forma independente entre França e Reino Unido, dando origem a **dialetos diferentes** de Prolog.
- Em **1981**, o governo japonês lançou o projeto **FGCS: Fifth Generation Computer Systems**, com o objetivo de criar **máquinas inteligentes** baseadas em **Prolog e IA simbólica**.
- O projeto despertou grande interesse mundial em **inteligência artificial e programação lógica**.

Prolog

- Apesar do entusiasmo inicial, o **projeto FGCS foi encerrado** na década seguinte.
- A promessa de máquinas inteligentes não se concretizou, mas o Prolog influenciou fortemente a **pesquisa em IA, raciocínio automático e sistemas especialistas**.
- Levou ao declínio no interesse e no uso de Prolog, apesar de ainda ser usado.

Sintaxe básica de Prolog

Elementos fundamentais

- **Constantes:** nomes que começam com **letra minúscula**
  - Ex: maria, chocolate, carro.

- **Variáveis:** começam com **letra maiúscula** ou _ (underscore).
  - Ex: X, Pessoa, _Temp.

- **Predicados:** representam relações lógicas entre termos.
  - Ex: pai(bill, jake).

“Bill é pai de Jake.”

Sintaxe básica de Prolog

Fatos

- Descrevem o que é verdadeiro no contexto:

```
female(mary).
father(bill, jake).
```

Sintaxe básica de Prolog

Regras

- Descrevem relações **condicionais** (se-então):

```
parent(X, Y) :- father(X, Y).
parent(X, Y) :- mother(X, Y).
```

"X é pai/mãe de Y **se** X é pai de Y."

- O símbolo **:-** corresponde à **implicação lógica (⊂)** da forma clausal.

Sintaxe básica de Prolog

- Prolog tenta **provar** uma consulta a partir dos fatos e regras.
- Usa **inferência por resolução**: busca fatos que tornem a consulta verdadeira.

Paradigmas de Linguagens de Programação

Programação Lógica