Unidade 2

Lógica de Predicados

Objetivos:

- Conhecer a linguagem formal Lógica de Predicados, assim como gerar fórmulas bem formadas nessa linguagem, a atribuição de valor verdade às fórmulas envolvendo quantificadores, algumas implicações e equivalências lógicas que podem ser empregadas durante o processo de raciocínio.
- Aplicar o conhecimento adquirido nos processos de representação de programas em Lógica de Predicados e de raciocínio automático na obtenção de respostas para consultas realizadas aos programas.

Capítulo 1

Linguagem lógica de predicados

Introdução

Uma **Teoria** consiste de um conjunto de proposições acerca de um universo de discurso ***(U)***. Considere o exemplo abaixo:

Catálogo:

p_1. *Soma.for é um programa FORTRAN*

p_2. *Soma.pas é um programa PASCAL*

p_3. *Soma.pro é um programa PROLOG*

p_4. *Todo programa FORTRAN é um programa procedimental*

p_5. *Todo programa PASCAL é um programa procedimental*

p_6. *Todo programa PROLOG é um programa declarativo*

U *= { Soma.for. Soma.pas, Soma.pro, FORTRAN, PASCAL, PROLOG}*

Um **Sistema Formal** consiste de uma **linguagem formal** e de uma **abstração adequada para os princípios** usados para decidir quando uma proposição é conseqüência lógica de outras.

A **Lógica Proposicional** é um exemplo de sistema formal.

Nesse contexto, a **Lógica de Predicados de Primeira Ordem** é uma extensão da Lógica Proposicional que pode ser caracterizada como um **sistema formal** apropriado **à definição de teorias** acerca de diversos universos de discurso.

A **Linguagem Lógica de Predicados** permite a representação de proposições que não podem ser representadas razoavelmente na Linguagem Lógica Proposicional.

Exemplo:

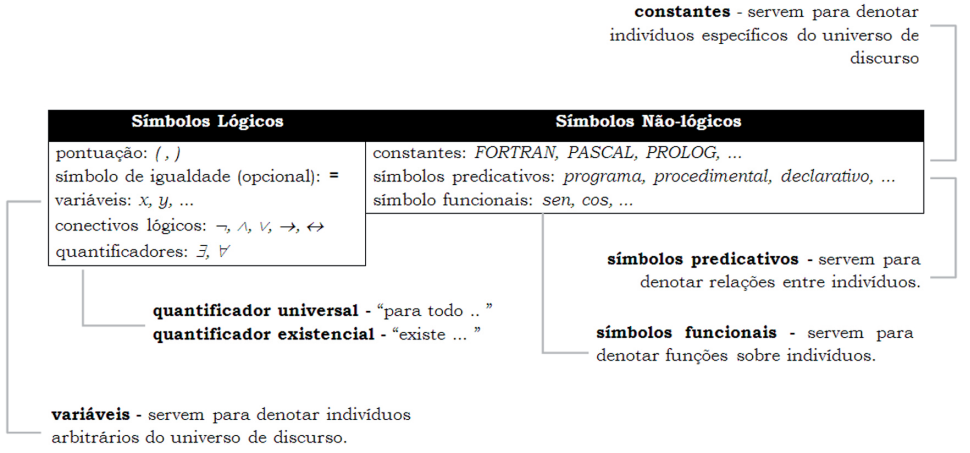
Proposição	Lógica Proposicional	Lógica de Predicados
p_1. *Sócrates é um homem*	p	*homem(Sócrates)*
p_2. *Platão é um homem*	q	*homem(Platão)*
p_3. *Todos os homens são mortais*	r	$\forall x$:(*homem*(x) $\rightarrow$ *mortal*(x))

A **Resolução para Lógica de Predicados** permite que se decida se uma proposição é conseqüência lógica de outras.

Linguagem Lógica de Predicados

O **Alfabeto** ***(A)*** da Linguagem Lógica de Predicados consiste de:

- uma série de **símbolos lógicos**:
  - variáveis,
  - conectivos lógicos,
  - quantificadores,
  - pontuação e
  - símbolo de igualdade;
- uma série de **símbolos não-lógicos**:
  - constantes,
  - símbolos predicativos e
  - símbolos funcionais.



Um **Termo** particular da linguagem pode ser:

- uma variável ou
- uma constante ou
- uma expressão da forma $f(t_1, t_2, t_m)$, onde

  f - é um símbolo funcional admitindo m argumentos e

  $t_1, t_2, ..., t_m$ - são termos.

As **Regras Sintáticas** da linguagem definem as sentenças da linguagem como sendo **fórmulas** de dois tipos:

- fórmula atômica da forma $p(t_1, t_2, t_n)$, onde

  p - é um símbolo predicativo admitindo n argumentos e

  $t_1, t_2, ..., t_n$ - são termos;
- expressão obtida compondo-se fórmulas
  - através dos conectivos lógicos ou
  - prefixando-se fórmulas com quantificadores.

Considere o exemplo a seguir:

Termos - permitem denotar indivíduos do domínio de discurso diretamente através de seus nomes ou indiretamente através de funções.

Termos	Fórmulas
variável: *x, y, ...*	fórmula atômica: *programa(Soma.for, FORTRAN), procedimental(x), ...*
constantes: *0, 1, Soma.for...*	fórmula: *programa(Soma.for, FORTRAN) ∧ programa(Soma.pas, PASCAL)*
expr. func.: *sen(x), cos(y), ...*	fórmula: *∀x:(programa(x, FORTRAN) → procedimental(x)), ...*

Fórmulas (não-atômicas) - permitem expressar propriedades das relações entre indivíduos do universo de discurso.

Fórmulas atômicas - permitem expressar relações entre indivíduos do universo de discurso e as propriedades destes.

A **Linguagem Lógica de Predicados** é o **conjunto de fórmulas bem formadas** sobre ***A***.

As **Regras Semânticas** da linguagem capturam o **significado das fórmulas** definidas sobre o alfabeto associando um dos valores verdade *V* ou *F*.

Veja o exemplo abaixo:

Universo	Alfabeto	Significado
U = *conj. reais*	***A*** = *{maior, x, y, ... } ∪* ***U***	*maior(x, y) = V* sss *(x, y) ∈ {(m,n) ∈* ***U*** *x* ***U*** *| m > n}* *maior(2, 1) = V* *maior(1, 1) = F*

Capítulo 2

Quantificadores

Uma **Proposição** é uma sentença declarativa que é verdadeira ou falsa.

Exemplo:

Proposição	Valor Verdade
"2 < 3"	Verdadeira

Uma **Função Proposicional** é uma sentença declarativa contendo uma ou mais variáveis que se torna uma proposição quando valores particulares (interpretações) são atribuídos às variáveis.

O **Domínio de Discurso** de uma função proposicional é o conjunto de valores possíveis para as variáveis da função. Supondo-se *D* o domínio de discurso de uma função proposicional *p*, podemos tornar *p* uma proposição **substituindo** vários membros de *D* em *p*

Considere o seguinte exemplo:

F. Prop.	$x \in D_1 = \{1,2,3\}$	$y \in D_2 = \{2,4\}$	Prop.	Representação	V. Verdade
"x < 3"	*1*	-	*"1 < 3"*	*p(1)*	verdadeira
"x < 3"	*3*	-	*"2 < 3"*	*p(3)*	falsa
"x < y"	*2*	4	*"2 < 4"*	*q(2, 4)*	verdadeira
"x < y"	*3*	2	*"3 < 2"*	*q(3, 2)*	falsa

Quantificando a função proposicional *p*.

Maneiras de se quantificar uma função proposicional:

- prefaciar a função proposicional com ***"para todo x em D"*** e
- prefaciar a função proposicional com ***"existe um x em D tal que"***

Veja o exemplo:

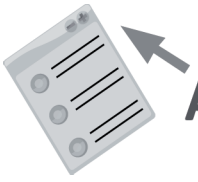
F. Prop.	Representação	Proposição	Representação
"x < 3"	*p(x)*	*"para todo x em D_1, p(x)"*	$\exists x{:}p(x)$
"x < 3"	*p(x)*	*"existe x em D_1 tal que p(x)"*	$\forall x{:}p(x)$
"x < y"	*q(x, y)*	*"para todo x em D_1, existe y em D_2 tal que q(x, y)"*	$\forall x{:}\exists y{:}q(x, y)$
"x < y"	*q(x, y)*	*"existe x em D_1 tal que para todo y em D_2, q(x, y)"*	$x{:}\forall y{:}q(x, y)$

O **Valor Verdade** de uma função proposicional quantificada é determinado da seguinte forma:

- Se *p(x)* é verdadeira para **toda interpretação de x em D**
  então $\forall x:p(x)$ é verdadeira
  senão $\forall x:p(x)$ é falsa
- Se *p(x)* é verdadeira para **pelo menos uma interpretação de x em D**
  então $\exists x:p(x)$ é verdadeira
  senão $\exists x:p(x)$ é falsa

Observação: valor verdade de uma função proposicional depende do domínio *D* utilizado

- Se *D* é **finito**, com elementos $x_1, x_2, ..., x_n$,
  então $\forall x:p(x)$ **é equivalente a** $p(x_1) \wedge p(x_2) \wedge ... \wedge p(x_n)$ e
  $\exists x:p(x)$ **é equivalente a** $p(x_1) \vee p(x_2) \vee ... \vee p(x_n)$
- Se *D* é **vazio**, não contém elementos,
  então $\forall x:p(x)$ **é verdadeira** e
  $\exists x:p(x$ **é falsa**



ATIVIDADES DE AVALIAÇÃO

1. Seja x $\in$ *D = {1, 2, 3, 4}*, *y* $\in$ *S = {-1, 0, 1, 2}*, *p(x)* é *"x < 3"* e *q(y)* é *"y < 3"*. Verifique o valor verdade das seguintes proposições: $\forall x:p(x)$, $\forall y:q(y)$, $x:p(x)$, $\exists y:q(y)$.

2. Seja x $\in$ *D* o conjunto de matemáticos acima de três metros de altura, *p(x) é "x gosta de chocolate"*. Verifique o valor verdade das seguintes proposições: $\forall x:p(x)$, $\exists x:p(x)$.

Negação de funções proposicionais quantificadas

Funções proposicionais quantificadas com "*para todo*" e "*existe*" estão relacionadas através do conectivo de negação, da forma como se segue:

$\neg(\forall x:p(x))$ $\exists x:\neg p(x)$

$\neg(\exists x:p(x)) \leftrightarrow \forall x:\neg p(x)$

Exemplo:

$\neg(\forall x:(p(x) \rightarrow q(x))) \leftrightarrow \exists x:(p(x) \wedge \neg q(x))$

$\neg(\exists x:(p(x) \wedge q(x))) \leftrightarrow \forall x:(\neg p(x) \vee \neg q(x))$

Funções proposicionais quantificadas em linguagem natural

Em linguagem natural, as funções proposicionais quantificadas são frequentemente utilizadas de diversas formas, como mostrado nos exemplos a seguir:

- ***"Existe um inteiro tal que seu quadrado é nove"***

  D = conjunto dos inteiros

  p(x) é "$x^2 = 9$"

  $\exists x: p(x)$

- ***"Se f é diferenciável então f é contínua"***

  D = conjunto de funções

  p(f) é "f é diferenciável"

  q(f) é "f é contínua"

  $\forall f:(p(f) \rightarrow q(f))$

- ***"Todo estudante de lógica entende quantificadores"***

  D = conjunto de estudantes

  p(x) é "x é estudante de lógica"

  q(x) é "x entende quantificadores"

  $\forall x:(p(x) \rightarrow q(x))$

- ***"Alguns estudantes de lógica entendem quantificadores"***

  D = conjunto de estudantes

  p(x) é "x é estudante de lógica"

  q(x) é "x entende quantificadores"

  $\exists x:(p(x) \wedge q(x))$

Observações:

"Todo p é um q" representado por $\forall x:(p(x) \rightarrow q(x))$

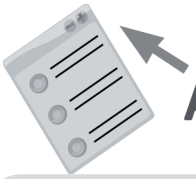
"Algum p é um q" representado por $\exists x:(p(x) \wedge q(x))$

Negação de funções proposicionais quantificadas em português

A relação de negação existente entre os quantificadores de funções proposicionais quantificadas é frequentemente usada em linguagem natural na geração de sentenças logicamente equivalentes, como mostrado no exemplo abaixo.

- *"Todo estudante de lógica entende quantificadores"*

Representação em Lógica de Predicados	$\forall x:(p(x) \rightarrow q(x))$
Negação da representação	$\exists x:(p(x) \wedge \neg q(x))$
Significado da representação em português	*"Alguns estudantes de lógica não entendem quantificadores"*



ATIVIDADES DE AVALIAÇÃO

1. Encontre uma negação para cada uma das proposições:

a) Existe um inteiro x tal que 4 = x + 2.

b) Para todo inteiro x, 4 = x + 2.

c) Todo estudante gosta de lógica.

d) Alguns estudantes não gostam de lógica.

e) Nem um homem é uma ilha.

f) Todo mundo que estuda lógica gosta.

g) Toda pessoa tem uma mãe.

2. Seja *D* o conjunto dos números naturais (isto é, *D = {1, 2, 3, 4, ...}*), *p(x) é "x é par"*, *q(x) é "x é divisível por 3"* e *r(x) é "x é divisível por 4"*. Expresse em português, determine valores verdade e dê uma negação para cada uma das proposições.

a) $\forall x:p(x)$.

b) $\forall x:(p(x) \vee q(x))$.

c) $\forall:(p(x) \rightarrow q(x))$.

d) $\forall:(p(x) \vee r(x))$.

e) $\forall x:(p(x) \wedge q(x))$.

f) $\exists x:r(x)$.

g) $\exists x:(p(x) \wedge q(x))$.

h) $\exists x:(p(x) \rightarrow q(x))$.

i) $\exists x:(q(x) \rightarrow q(x + 1))$.

Sentenças declarativas que envolvem mais de um quantificador

É importante destacar que uma dada função declarativa pode ser quantificada simultaneamente por mais de um quantificador.

Exemplos:

"Para todo inteiro par n, existe um inteiro k tal que n = 2k"

"Para todo y em B, existe um x em A tal que y = f(x)"

No entanto, é importante perceber que a ordem na qual os quantificadores estão dispostos interfere na interpretação lógica da sentença, como mostrado abaixo.

- Se $S = \{x_1, x_2, ..., x_n\}$ e $T = \{ y_1, y_2, ..., y_m\}$
  então $\forall x:(\exists y:p(x,y))$ não é equivalente a $\exists y:(\forall x:p(x,y))$.

Exemplo:

- Se $S = T = \{1, 2\}$ e $p(x,y)$ é "$x = y$"

  então $\forall x:(\exists y:p(x,y))$ é uma proposição verdadeira e

  $\exists y:(\forall x:p(x,y))$ é uma proposição falsa

ATIVIDADES DE AVALIAÇÃO

1. Seja $S = T =$ *conjunto de todas as pessoas* e $p(x,y)$ é *"y é mãe de x"*. Qual o valor verdade de $\forall x:(\exists y:p(x,y))$ e $\exists y:(\forall x:p(x,y))$?

2. Seja a proposição: *"Para todo cachorro preto no sofá existe uma pulga no carpete tal que se o cachorro é preto então a pulga pica o cachorro".*

a) Qual a representação da proposição em Linguagem Lógica de Predicados?

b) Qual a negação da proposição em Linguagem Lógica de Predicados?

c) Qual a negação da proposição em português?

Equivalências lógicas

Além da equivalência lógica dada pela negação entre os quantificadores "para todo" e "existe", outras equivalências existem, como apresentado abaixo:

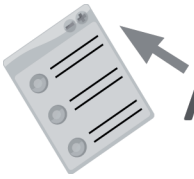
- $\exists x:(\exists y:p(x,y)) \leftrightarrow \exists y:(\exists x:p(x,y))$
- $\forall x:(\forall y:p(x,y)) \leftrightarrow \forall y:(x:p(x,y))$
- $((\exists x:p(x)) \vee (\exists x:q(x))) \leftrightarrow \exists x:(p(x) \vee q(x))$
- $((\forall x:p(x)) \wedge (\forall x:q(x))) \leftrightarrow \forall x:(p(x) \wedge q(x))$

Implicações lógicas

Além dessas equivalências lógicas, algumas implicações lógicas podem ser desenhadas.

- $\exists y:(\forall x:p(x,y)) \rightarrow \forall x:(\exists y:p(x,y))$
- $((\forall x:p(x)) \vee (\forall x:q(x))) \rightarrow \forall x:(p(x) \vee q(x))$
- $\exists x:(p(x) \wedge q(x)) \rightarrow ((\exists x:p(x)) \wedge (x:q(x)))$
- $\forall x:(p(x) \rightarrow q(x)) \rightarrow ((\forall x:p(x)) \rightarrow (\forall x:q(x)))$

As **idéias e métodos de análise** utilizados para declarações envolvendo dois quantificadores podem ser estendidos para **três (ou mais) quantificadores.**



ATIVIDADES DE AVALIAÇÃO

1. Seja *S = conjunto de inteiros, p(x)* é *"x é par"* e *q(x)* é *"x é impar"*.

Verifique as duas últimas equivalências lógicas acima.

2. Seja *S = conjunto de inteiros, p(x)* é *"x é par"* e *q(x)* é *"x é impar"*.

Verifique as três últimas implicações lógicas acima.

3. Traduzir as seguintes sentenças em formas simbólicas, indicando escolhas apropriadas para os domínios:

a) Para todo inteiro par n existe um inteiro k tal que n = 2k.

b) Para toda linha l e todo ponto p não em l existe uma linha l' que passa por p que é paralela a l.

c) Para todo y em B existe um x em A tal que f(x) = y.

d) Para todo x em G existe um x' em G tal que xx' = e.

e) Se todo inteiro é impar então todo inteiro é par.

f) Todo mundo ama alguém alguma vez.

g) Para todo inteiro n existe outro inteiro maior que 2n.

h) A soma de dois inteiros pares é par.

4. Ache uma negação em Português para todas as proposições no exercício acima.

5. Considere que *p*(*x*,*y*) representa *"x + 2 > y"* e *D o conjunto dos números naturais* (*D= {1, 2, 3,...}*). Escreva em palavras e atribua valores verdade para

a) $\forall x:(\exists y:p(x,y))$.

b) $\exists x:(y\ p(x,y))$.

c) $\forall x:(\forall y:p(x,y))$.

d) $\exists x:(\exists y:p(x,y))$.

e) $\forall y:(\exists x:p(x,y))$.

f) $\exists y:(\forall x:\ p(x,y))$.

6. Considere a seguinte proposição:

"Para toda galinha no galinheiro e para toda cadeira na cozinha existe uma frigideira no armário tal que se o ovo da galinha está na frigideira então a galinha está a dois metros da cadeira".

a) Traduza em forma simbólica.

b) Expresse sua negação em símbolos e em Português.

Capítulo 3

Representação do conhecimento e programação em lógica

A Programação em Lógica de Predicados é composta pelos seguintes elementos:

- Um **programa** é uma teoria formalizada em Linguagem Lógica de Predicados, isto é, uma coleção finita de fórmulas em Linguagem Lógica de Predicados;
- Uma **consulta** é qualquer fórmula em Linguagem Lógica de Predicados exprimindo as condições a serem satisfeitas por uma resposta correta;
- Uma **resposta** correta é um indivíduo do universo de discurso ***U***; e
- Um **método de busca** que pode ser utilizado para obtenção de respostas é a Resolução para Lógica de Predicados.

Exemplo:

Programa em Lógica de Predicados	Consulta em Lógica de Predicados
F_1. *programa(Soma.for, FORTRAN)*	F_6. $\exists x$:*procedimental(x)*
F_2. *programa(Soma.pro, PROLOG)*	
F_3. $\forall x$:(*programa(x, FORTRAN)* $\rightarrow$ *procedimental(x)*)	
F_4. $\forall x$:(*programa(x, PASCAL)* $\rightarrow$ *procedimental(x)*)	
F_5. $\forall x$:(*programa(x, PROLOG)* $\rightarrow$ *declarativo(x)*)	

Exemplos do uso da Linguagem Lógica de Predicados como linguagem de programação

Programa em Linguagem natural e sua representação em Linguagem Lógica de Predicados

1. *"Marcos era um homem"*
   homem(Marcos)
2. *"Marcos Nasceu em Pompéia"*
   pompeano(Marcos)
3. *"Todos os que nasceram em Pompéia eram romanos"*
   $\forall x$:(*Pompeano(x)* $\rightarrow$ *Romano(x)*)
4. *"César era um soberano"*
   soberano(César)

5. *"Todos os romanos eram leais a César ou então odiavam-no*
   $\forall x:(Romano(x) \rightarrow (leal\_a(x,César) \vee odeia(x,César)))$
6. *"Marcos tentou assassinar César"*
   tentou_assassinar(Marcos,César)
7. *"As pessoas só tentam assassinar soberanos aos quais não são leais"*
   $\forall x:(\forall y:((pessoa(x) \wedge soberano(y) \wedge tentou\_assassinar(x,y)) \rightarrow \neg leal\_a(x,y)))$
8. *"Todo mundo é leal a alguém"*
   $\forall x:(\exists y:leal\_a(x,y))$
9. *"Todos os homens são pessoas"*
   $\forall x:(homem(x) \rightarrow pessoa(x))$

Consulta em Linguagem natural e sua representação em Linguagem Lógica de Predicados:

10. *"Marcos era leal a César?"*
    leal_a(Marcos,César) ou
    $\neg leal\_a(Marcos,César)$

Exemplo de um **método de busca de respostas**: **raciocínio a partir do objetivo para trás** - a fim de responder à consulta, isto é, a fim de comprovar o objetivo, utilizamos "modus ponens" como fundamento e transformamos o objetivo principal (consulta) em um subobjetivo (ou, possivelmente, em um conjunto de subobjetivos) que possa, por sua vez, ser transformado e assim por diante, até todos os subobjetivos gerados serem satisfeitos (ou seja, até todas as variáveis geradas no processo de raciocínio serem substituídas por constantes apropriadas).

Resposta: Marcos não era leal a César.

$\neg leal\_a(Marcos,César)$
|
(7, substituição)
$pessoa(Marcos) \wedge$
$soberano(César) \wedge$
$tentou\_assassinar(Marcos,César)$
|
(4)
$pessoa(Marcos) \wedge$
$tentou\_assassinar(Marcos,César) \wedge$
|
(6)
$pessoa(Marcos)$
|
(9)
$homem(Marcos)$
|
(1)

Aspectos importantes que precisam ser consideradas no processo de conversão de frases em linguagem natural para Linguagem Lógica de Predicados:

- Frases em linguagem natural são **ambíguas**, portanto, pode ser difícil escolher uma representação correta.

  Exemplo:

  Declaração 5:

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

  $\forall x:(Romano(x) \rightarrow (leal\_a(x,César) \vee odeia(x,César)))$ ou

  $\forall x:(Romano(x) \rightarrow ((leal\_a(x,César) \vee odeia(x,César)) \wedge \neg(leal\_a(x,César) \vee odeia(x,César)))$

- Para se usar um conjunto de declarações eficientemente, normalmente é necessário ter acesso a um outro conjunto de declarações que representam **fatos** que as pessoas consideram **óbvios** demais para serem declarados.

  Exemplo:

  Declaração 9:

  "Todos os homens são pessoas"

  $\forall x:(homem(x) \rightarrow pessoa(x))$

- A relação **elemento pertence conjunto** pode ser representada por uma fórmula atômica da forma *conjunto(elemento)*.

  Exemplo:

  Declaração 2: *"Marcos Nasceu em Pompéia"* - Elemento Marcos pertence ao conjunto dos pompeanos

  Pompeano(Marcos) - pertence(Marcos,Pompeanos)

  Declaração 4: *"César era um soberano"* - Elemento César pertence ao conjunto dos soberanos

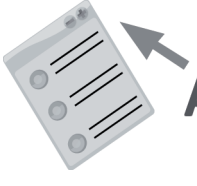
  soberano(César) - pertence(César, Soberanos)

- A relação **conjunto1 contido conjunto2** pode ser representada por uma fórmula da forma $\forall x:(conjutno1(x) \rightarrow conjunto2(x))$.

  Exemplo:

  Declaração 3: *"Todos os que nasceram em Pompéia eram romanos"* - Conjunto de pompeanos contido no de romanos

  $\forall x:(Pompeano(x) \rightarrow Romano(x))$ - *contido(Pompeanos,Romanos)*



ATIVIDADES DE AVALIAÇÃO

1. Represente as seguintes sentenças por fbf's em lógica de predicados:

a) "Um sistema de computador é inteligente se pode realizar uma tarefa que, se realizada por um humano, requer inteligência".

b) "Uma fórmula cujo conectivo principal é uma → é equivalente a alguma fórmula cujo principal conectivo é um ∨".

c) "Se a entrada para o algoritmo da unificação é um conjunto de expres sões unificáveis, a saída é o mgu (unificador mais geral); se a entrada é um conjunto de expressões não unificáveis, a saída é fracasso".

d) "Se não se pode dizer um fato a um programa, então o programa não pode aprender o fato.

2. Traduza as seguintes fbf's em Lógica de Predicados para o português.

a) $\forall x:(duvida(x) \rightarrow perde(x))$.

b) $\neg\exists x:(homem\_negócios(x) \wedge gosta(x,vida\_noturna))$.

c) $\neg\forall x:(brilhante(x) \rightarrow ouro(x))$.

d) $\forall x:(\forall y:(\forall z:(mais\_rápido(x,y) \wedge mais\_rápido(y,z)) \rightarrow mais\_rápido(x,z)))$.

3. O que está errado com o seguinte argumento?

- *"Os homens estão amplamente distribuídos pela Terra".*
- *"Sócrates é um homem".*
- *"Portanto, Sócrates está amplamente distribuído pela Terra"*

4. Como é que os fatos representados pela sentença acima devem ser representados na lógica para que este problema não ocorra.

5. Expresse as seguintes sentenças em Lógica de Predicados. Use somente o objeto simbólico *Erasmus* e as relações simbólicas *homem(x)*, *mulher(x)*, *vegetariano(x)*, *Açougueiro(x)*, *Gosta(x,y)*, *=(x,y)*.

a) Erasmus não gosta de nenhum vegetariano.

b) Erasmus não gosta de todo vegetariano.

c) Nenhum homem é açougueiro e vegetariano.

d) Todos os homens gostam de vegetarianos, exceto os açougueiros.

e) Alguns vegetarianos não gostam de todos os açougueiros.

f) Os únicos vegetarianos são homens que gostam de vegetarianos.

g) Homens só gostam de mulheres que não são vegetarianas.

h) Nenhuma mulher gosta de qualquer homem que não gosta de todos os vegetarianos.

i) Todo homem gosta de uma mulher diferente.

6. Traduza as seguintes fbf's em Lógica de Predicados para o português.

a) $\forall x:(duvida(x) \rightarrow perde(x))$.

b) $\neg\exists x:(homem\_negócios(x) \wedge gosta(x,vida\_noturna))$.

c) $\neg\forall x:(brilhante(x) \rightarrow ouro(x))$.

d) $\forall x:(\forall y:(\forall z:(mais\_rápido(x,y) \wedge mais\_rápido(y,z)) \rightarrow mais\_rápido(x,z)))$.

Capítulo 4

Funções e Predicados Computáveis e a Noção de Igualdade

Considerando que estamos interessados nos aspectos de construção de programas provadores automáticos, capazes de encontrar respostas para consultas, as funções computáveis são necessárias quando o programa precisar avaliar a verdade de predicados que possuam expressões funcionais como termos. Por exemplo:

"2 + 3 é maior que 1"

maq(+(2,3),1))

Neste caso, o programa provador deve chamar uma **função computável**, capaz de computar o valor da expressão *2+3* e, em seguida, avaliar o predicado *maq(5,1)*.

No programa em lógica apresentado anteriormente, utilizamos **predicados isolados** (para diferenciar de predicados computáveis) para a representação de fatos simples, e uma combinação destes predicados para a representação de relações entre fatos. Por exemplo:

6. *"Marcos tentou assassinar César"*
   tentou_assassinar(Marcos,César)

9. *"Todos os homens são pessoas"*

   $\forall x:(homem(x) \rightarrow pessoa(x))$

A representação por meio de predicados isolados deve ser empregada quando o número de fatos não for muito grande. Se, por exemplo, desejarmos representar fatos que expressem relações do tipo *maior que*, em um conjunto composto por um número grande de elementos numéricos,

"um é maior que zero"
maq(1,0)

"dois é maior que um"
maq(2,1)

"três é maior que 2"
maq(3,2)

...

Deveremos empregar um **predicado computável**. Assim, em vez do programa provador automático procurar pelo predicado explicitamente no conjunto de fórmulas disponíveis (ou tentar deduzir o predicado através de mais raciocínio) para a obtenção de respostas, ele deve invocar um procedimento e este avaliará o predicado.

A **noção de igualdade** permite que indivíduos iguais do universo de discurso sejam substituídos uns pelos outros sempre que este procedimento parecer útil durante o processo de busca de respostas. Por exemplo, a representação da sentença abaixo, utilizando a noção de igualdade,

"Estamos agora em 1998"
agora = 1998

indica que sempre que a constante *agora* aparecer como termo de algum predicado, ela poderá ser substituída pela constante *1998*.

Exemplos que mostram a utilidade das idéias de funções e predicados computáveis e a noção de igualdade

Programa em Linguagem natural e sua representação em Linguagem Lógica de Predicados

1. "*Marcos era um homem*"
   homem(Marcos)
2. *"Marcos Nasceu em Pompéia"*
   pompeano(Marcos)
3. *"Marcos nasceu em 40 d.C."*
   nasceu(Marcos,40)
4. *"Todos os homens são mortais"*
   $\forall x:(homem(x) \rightarrow mortal(x))$
5. *"Todos os habitante de Pompéia morreram quando o vulcão entrou em erupção em 79 d.C."*
   $\forall x:(Pompeano(x) \rightarrow morreu(x,79))$
   entrou_em_erupção(vulcão,79)
6. *"Nenhum mortal vive mais que 150 anos"*
   $\forall x:(\forall t_1:(\forall t_2:((mortal(x) \wedge nasceu(x,t_1) \wedge \mathbf{maq}(-(t_2,t_1),150)) \rightarrow morto(x,t_2))))$
7. *"Estamos agora em 1998"*
   agora **=** *1998*
8. *"Estar morto significa não estar vivo"*
   $\forall x:(\forall t:((morto(x,t) \rightarrow \neg vivo(x,t)) \wedge (\neg vivo(x,t) \rightarrow morto(x,t)))$
9. *"Se alguém morre então ele está morto em todos os momentos posteriores"*
   $\forall x:(\forall t_1:(\forall t_2:((morreu(x,t_1) \wedge \mathbf{maq}(t_2,t_1)) \rightarrow morto(x,t_2))))$

Consulta em Linguagem natural e sua representação em Linguagem Lógica de Predicados:

10. *"Marcos está vivo?"*
    vivo(Marcos,agora) ou $\neg vivo(Marcos,agora)$

Método de busca de respostas: Raciocínio a partir do objetivo para trás.

Resposta 1: Marcos não está vivo (porque foi morto pelo vulcão).

$\neg vivo(Marcos,agora)$

(8, substituição)

$morto(Marcos,agora)$

(9, substituição)

$morreu(Marcos,t_1) \wedge$
$maq(agora,t_1)$

(5, substituição)

$Pompeano(Marcos)$
$maq(agora,79)$

(2)

$maq(agora,79)$

(7)

$maq(1998,79)$

(calcular *maq*)

nil

Resposta 2: Marcos não está vivo (senão estaria com mais de 150 anos).

$\neg vivo(Marcos,agora)$

(8, substituição)

$morto(Marcos,agora)$

(6, substituição)

$mortal(Marcos) \wedge$
$nasceu(Marcos,t_1) \wedge$
$maq(-(agora,t_1),150)$

(4, substituição)

$homem(Marcos) \wedge$
$nasceu(Marcos,t_1) \wedge$
$maq(-(agora,t_1),150)$

(1)

$nasceu(Marcos,t_1) \wedge$
$maq(-(agora,t_1),150)$

(3, substituição)

$maq(-(agora,40),150)$

(7, substituição)

$maq(-(1998,40),150)$

(computar -)

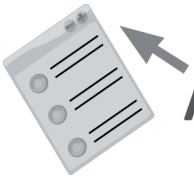
$maq(1958,150)$

(computar *maq*)

nil

Observações:

- A representação de fatos que adquirem gradualmente com a experiência pode diminuir o número de etapas para a consulta ser respondida.
- Um bom método de busca de respostas deve ser capaz de:
  - Determinar que existe um casamento entre dois predicados
    Ex: *morto(Marcos,agora)* casa com *morto(x,t_2),*
  - Garantir substituições uniformes em toda a prova r
  - Aplicar *modus ponens*.
- Do ponto de vista computacional, precisamos de um método de busca de respostas capaz de executar em uma única operação a série de processos envolvidos no raciocínio com declarações em Lógica de Predicados.



ATIVIDADES DE AVALIAÇÃO

1. Usando o primeiro programa em lógica desta seção responda à consulta: *"Marcos odiava César?"*

2. Considere as seguintes sentenças:

- *"João gosta de todo tipo de comida".*
- *"Maçãs são comidas".*
- *"Frango é comida"*
- *"Qualquer coisa que alguém coma e que não cause sua morte é comida".*
- *"Paulo come amendoim e está vivo".*
- *"Susana come tudo o que Paulo come"*

Traduza as sentenças em fbf's em Lógica de Predicados e responda a consulta abaixo, usando o raciocínio para trás:

- *"João gosta de amendoim?"*