Inteligência Artificial
Inferência com Lógica de Predicados
Prof^{o} Wagner Toscano

PMR/POLI/USP, São Paulo, Brasil
wt@wagnertoscano.eti.br

31 de outubro de 2009

Introdução

Existem diversos tipos de argumentos que não podem ser expressos em lógica proposicional. Como por exemplo, o argumento válido:

Sócrates é homem.
Todo homem é mortal.
Logo, Sócrates é mortal.

Porém aplicando lógica proposicional a forma do argumento seria:

$$\{p, q\} \models r$$

e não haveria como demonstrar que a conclusão é uma conseqüência lógica das premissas p e q. Isso ocorre porque a validade do argumento depende da semântica da palavra **Todo**, que não é considerada na lógica proposicional.

Para tratar esse tipo de argumento a lógica proposicional foi estendida, com a criação da lógica de predicados.

A **Lógica de predicados** é um sistema lógico formado por um conjunto de fórmulas e um conjunto de regras de inferência. As fórmulas são sentenças que pertencem à uma linguagem formal. Cada fórmula pode ser associada a um valor verdade (Verdadeiro ou Falso).

A transformação de sentenças declarativas (em que não haja paradoxo), com a utilização de conectivos lógicos, permitem construir proposições que poderão ser implementadas em uma linguagem lógica, com o objetivo de poderem ser inferidas.

Porém, a transformação de sentenças como:

Todos os amigos de Carlos são amigos de Jonas.
Pedro não é amigo de Jonas.
Logo, Pedro não é amigo de Carlos.

ou

Todos os humanos são racionais.
Alguns animais são humanos.
Portanto, alguns animais são racionais.

Possuem palavras que denotam quantidades não precisas, como: todos, alguns, qualquer, nenhum etc. Dessa forma para que se possa traduzir essas sentenças para uma proposição a ser implementada, é necessário introduzir novos símbolos que auxiliarão na conversão. Na tabela 1, para recordação também, são apresentados os símbolos já conhecidos além dos novos símbolos.

Símbolo	Descrição
conectivos	auxiliam no relacionamento das proposições
parênteses	agrupam objetos relacionados
variáveis	representam objetos que não estão identificados no Universo considerado
constantes	representam objetos identificados no universo
símbolos de predicados	representam propriedades ou relações entre objetos do universo
quantificadores	representam o relacionamento de uma quantidade indefinida de objetos
termos	é o nome genérico de variáveis e constantes

Tabela 1: Símbolos do Cálculo de Predicados

Uma regra de inferência é uma regra sintática que, quando aplicada repetidamente a uma ou mais fórmulas verdadeiras, gera apenas novas fórmulas verdadeiras. As regras de inferência fornecem uma estrutura dedutiva à lógica.

Formalmente, a lógica de predicados é denominada Linguagem Lógica de Primeira Ordem quando é determinada a especificação dos seguintes conjuntos:

- Um conjunto P de Símbolos de Predicados:

$$\begin{array}{lcl} \neg & = & \textit{not}\text{, negação, não} \\ \wedge & = & \text{e, } \textit{and} \\ \vee & = & \text{ou , } \textit{or} \\ \forall x & = & \text{para qualquer x} \\ \exists x & = & \text{existindo x} \end{array}$$

  Os 2 últimos símbolos são denominados de quantificadores. Note que os símbolos: $\rightarrow, \leftrightarrow, \bigodot$ e $\bigoplus$, são representados pelos símbolos de predicados, exemplo:

$$\begin{array}{lcl} a \rightarrow b & = & \overline{a} + b \\ a \leftrightarrow b = \bigodot & = & a.b + \neg a.\neg b \\ \bigoplus = \neg bigodot & = & \neg a.b + a.\neg b \end{array}$$

- Um conjunto F de Símbolos de Função, como por exemplo: g, f, $g1$, $f1$, $\ldots$;
- Um conjunto C de Símbolos de Constantes, como por exemplo: G, F, $G1$, $G2$, $\ldots$; e
- Um conjunto V de Símbolos de Variável: x, $x1$, y, $y1$, $dots$.

1 Quantificadores

A seguir uma relação de proposições e suas respectivas conversões para a simbologia de cálculo de predicados:

- Maria é inteligente.
  inteligente("Maria").
- Alguém gosta de Maria.
  gosta(x,"Maria"). ou (∃x) gosta(x,"Maria").
  Se existe alguém, esse alguém gosta de Maria.
- Todos os amigo de Carlos são amigos de Jonas.
  (∀x) (amigo(x,"Carlos") → amigo(x,"Jonas")).
  Qualquer que seja a pessoa, se essa pessoa é amiga de Carlos, logo ela é amiga de Jonas.
- Todos os humanos são racionais.
  (∀x) (humano(x) → racional(x)).
  Qualquer coisa que seja humano, essa coisa é racional.

Para os silogismos (argumento) a seguir:

- Todos os amigos de Carlos são amigos de Jonas.
  Pedro não é amigo de Jonas.
  Logo, Pedro não é amigo de Carlos.

  Em cálculo de predicados tem-se:
  (∀x)(amigo(x,"Carlos") → amigo(x,"Jonas")).
  ¬ amigo("Pedro","Jonas").
  ¬ amigo("Pedro","Carlos").
- Todos os humanos são racionais.
  Alguns animais são humanos.
  Portanto, alguns animais são racionais.

  Em cálculo de predicados tem-se: (∀x)(humano(x) → racional(x).
  (∃x)(animal(x) ∧ humanos(x))
  (∃x)(animal(x) ∧ racional(x))

1.1 Negação de Quantificadores

Considerando a fórmula: $(\forall x)(predicado(x))$, e
o conjunto universo $S = a, b, c$, podemos afirmar que:

$$(\forall x)(predicado(x) \leftrightarrow predicado(a) \wedge predicado(b) \wedge predicado(c).$$

e a negação:

$$\neg(\forall x)(predicado(x) \leftrightarrow \neg(predicado(a) \wedge predicado(b) \wedge predicado(c)) \leftrightarrow \neg predicado(a) \vee \neg' predicado(b) \vee \neg predicado(c).$$

Que pode ser traduzido como: Existe um objeto em S tal que $\neg predicado(x)$, ou

$$\neg(\forall x)predicado(x) \leftrightarrow (\exists x)\neg predicado(x)$$

A seguir, têm-se um exemplo de aplicação:

wt@wagnertoscano.eti.br

Linguagem Natural	Lógica de 1^a Ordem
Pedro é um menino.	menino(Pedro).
Maria gosta de doces.	$\forall x : doce(x) \rightarrow gosta(Maria, x)$.
Quindim é um doce.	doce(Quindim).
Mimi é um gato pequeno.	gato(Mimi). pequeno(Mimi).
Rex é um cão grande.	cão(Rex). grande(Rex).
Cães e gatos, se pequenos, são animais domésticos.	$\forall x : (gato(x) \vee cao(x)) \wedge pequeno(x) \rightarrow domestico(x)$.

Logo a representação do argumento

Sócrates é homem.
Todo homem é mortal.
Logo, Sócrates é mortal.

Pode ser representada por:

$\forall$ x: homem(x) $\rightarrow$ mortal(x).

2 Semâtica

Como na lógica proposicional, o significado das fórmulas na lógica de predicados depende da interpretação dos símbolos. A interpretação consiste de:

- um conjunto $D \neq \emptyset$, denominado domínio da interpretação;
- um mapeamento que associa cada predicado a uma relação em D;
- um mapeamento que associa cada variável ou função a um elemento em D e
- um mapeamento que associa cada constante a um elemento fifixo em D.

O quantificador universal ($\forall$) corresponde a uma conjunção e o quantificador existencial ($\exists$) corresponde a uma disjunção. Como por exemplo, supondo:

$$D = \{a, b, c\} \leftrightarrow \forall x p(x) \leftrightarrow p(a) \wedge p(b) \wedge p(c)$$

$$D = \{a, b, c\} \leftrightarrow \exists x q(x) \leftrightarrow q(a) \vee q(b) \vee q(c)$$

Lembrando que:

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$$

Logo:

$$\neg\forall x p(x) \equiv \exists x \neg p(x).$$

De forma análoga:

$$\neg\exists x p(x) \equiv \forall x \neg p(x).$$

Enunciado categórico

São denominados enunciados categóricos, os enunciados que aparecem freqüentemente na Lógica Clássica. Sendo 4 os mais comuns denominados de A, E, I e O

A : Universal afirmativa: "Todo P é Q"
$(\forall x)(P(x) \rightarrow Q(x))$
Todos os homens são mortais.

E : Universal negativa: “Nenhum P é Q” ou “Todo P não é Q”
$(\forall x)(P(x) \rightarrow \neg Q(x))$
Nenhum home é extraterrestre.

I : Particular afirmativa: “Algum P é Q”
$(\exists x)(P(x) \wedge Q(x))$
Alguns homen são cultos.

O : Particular negativa: “Algum P não é Q”
$(\exists x)(P(x) \wedge \neg Q(x))$
Alguns homens não são honestos.

2.1 Representação no diagrama de Venn

São utilizados para interpretação e validação de argumentos, cujas as premissas e conclusões são enunciados categóricos do tipo **A**, **E**, **I**, **O**. Porém, não devem ser considerados instrumentos de prova infalível.

A construção é simples de um diagrama de Venn é simples:

1. Cada círculo representa uma classe de objeto que quando em branco indica ausência de informação a respeito do conjunto.
2. Círculo hachurado ou região de um círculo hachurada, representa região vazia de elementos.
3. Círculo ou região de um círculo com X representa região não vazia de elementos.

Na figura 1 é apresentada a representação gráfica de um diagrama de Venn, tendo como referência o predicado q, representando “quebrado”, no conjunto Q.

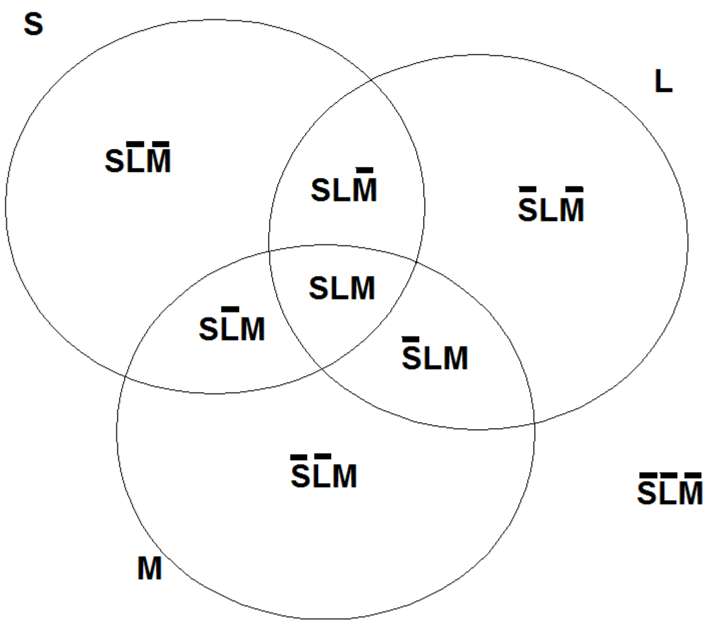

Figura 1: Elementos de representação de um diagrama de Venn

Reconhecer o tipo de uma sentença facilita a tradução para a lógica de predicados. Como exemplo, têm-se:

- Toda cobra é venenosa: $\forall x : cobra(x) \rightarrow venenosa(x)$.
- Os remédios são perigosos: $\forall x : remedio(x) \rightarrow perigoso(x)$.
- Nenhuma bruxa é bela: $\forall x : bruxa(x) \rightarrow \neg bela(x)$.
- Não existe bêbado feliz: $\forall x : bebado(x) \rightarrow \neg feliz(x)$.
- Algumas pedras são preciosas: $\exists x : pedra(x) \wedge preciosa(x)$.
- Existem plantas que são carnívoras: $\exists x : planta(x) \wedge carnivora(x)$.
- Alguns políticos não são honestos: $\exists x : politico(x) \wedge honesto(x)$.
- Existem aves que não voam: $\exists x : ave(x) \wedge \neg voa(x)$.

3 Exercícios

Formalize as sentenças a seguir:

1. Tudo que sobe, desce.
2. Nenhum leão é manso.
3. Todo circo tem palhaço.
4. Toda pedra preciosa é cara.
5. Nenhum homem é infalível.
6. Alguns escritores são cultos.
7. Ninguém gosta de impostos.
8. Existem impostos que não são bem empregados.

Equivalência de senteças

- “Nem tudo que brilha é ouro.”

  Se nem tudo que brilha é ouro, então significa que exite alguma coisa que brilha e não é ouro. Logo:

  $\neg\forall x : brilha(x) \rightarrow ouro(x)$, ou
  $\exists x : brilha(x) \wedge \neg ouro(x)$.

  Comprovação

  $$\begin{aligned} &\neg\forall x : brilha(x) \rightarrow ouro(x) \\ &\equiv \neg\forall x : \neg brilha(x) \vee ouro(x) \\ &\equiv \exists x\neg : \neg brilha(x) \vee ouro(x) \\ &\equiv \exists x : brilha(x) \wedge \neg ouro(x) \end{aligned}$$

- “Nem todo ator americano é famoso.”

  $\neg\forall x : ator(x) \wedge americano(x) \rightarrow famoso(x)$.

  Se nem todo ator americano é famoso, então existe ator americano que não é famoso.

  $\exists x : ator(x) \wedge americano(x) \wedge \neg famoso(x)$

Comprovação

$$\begin{aligned}
&\neg\forall x : ator(x) \wedge americano(x) \rightarrow famoso(x)\\
&\equiv \neg\forall x : \neg(ator(x) \wedge americano(x)) \vee famoso(x)\\
&\equiv \neg\forall x : \neg ator(x) \vee \neg americano(x) \vee famoso(x)\\
&\equiv \exists x\neg : \neg ator(x) \vee \neg americano(x) \vee famoso(x)\\
&\equiv \exists x : ator(x) \wedge americano(x) \wedge \neg famoso(x)
\end{aligned}$$

Exercícios: verificar as sentenças equivalentes

Nem toda estrada é perigosa e Algumas estradas não são perigosas.
Nem todo bêbado é fumante e Alguns bêbados são fumantes
.

Inferência Lógica

Inferir conclusões corretas, a partir de um conjunto de premissas, Considerando o argumento:

$$\{homem(socrates), \forall X[homem(X) \rightarrow mortal(X)\} \models mortal(socrates)$$

Tendo X como uma variável universal, sendo $X = socrates$, normalizando, tem-se:

(1)	$homem(socrates)$	α
(2)	$\neg homem(socrates) \vee mortal(socrates)$	$\neg\alpha \vee \beta$
(3)	$\neg mortal(socrates)$	conclusão negada
(4)	$\neg home(socrates)$	RES(2,3)
(5)	$\square$	contradição (1,4)

A instanciação universal, a qual permite substituir uma variável por uma constante qualquer do domínio, não funciona corretamente para todos os argumentos. Por exemplo:

Toda árvore dá um fruto.
$\forall X[arvore(X) \rightarrow \exists Y[fruto(Y, X)]]$

Substituindo X por uma constante 'Mangueira':

Toda árvore com nome 'Mangueira' dá um fruto.
$arvore(Mangueira) \rightarrow \exists Y[fruto(Y, Mangueira)]$.

Se o antecedente ($arvore(Mangueira)$) for verdade, o conseqüênte ($\exists Y[fruto(Y, Mangueira)]$.) também o é. Aplicando o conceito da tabela da verdade de implicação ($A \rightarrow B$), se o antecedente for falso, o conseqüênte continua verdadeiro.

Se a interpretação for:

Toda arvore dá uma fruta chamada Laranja.
$\forall X[arvore(X) \rightarrow [fruta(Laranja, X)]]$

O significado foi alterado, porque Y depende do valor de X.

Uma forma de eliminar essa ambiguidade é aplicando o processo denominado "Skolemização". Esse processo consiste em identificar um valor correto para substituir a variável existencial, essa substituição é dada pela permuta por uma função distinta, em que os argumentos são as variáveis universais.

$\forall X[arvore(X) \rightarrow \exists Y[fruto(Y, X)]]$.
$\forall X[arvore(X) \rightarrow \exists Y[fruto(nasce(X), X)]]$

O significado a sentença continua original

$$\forall X, Y[p(X, Y) \to \exists Z \forall W[q(Z, X) \land q(W, Z)]]$$

Sendo $Z = f(X, Y)$, tem-se:

$$\forall X, Y[p(X, Y) \to \exists f(X, Y) \forall W[q(f(X, Y), X) \land q(W, f(X, Y))]]$$

3.1 Exercícios

1. Todo cão é fiel ao seu dono.
2. Existe um lugar onde todos são felizes.

4 Unificação

Unificação é o processo de substituir uma variável por uma constante, uma variável ou uma função. Dessa forma as fórmulas atômicas idênticas podem ser canceladas. Por exemplo:

gosta(ana,X), gosta(Y,Z) → Y=ana e X=Z ⊨ gosta(Y,X).

ama(deus,Y), ama(X, filho(X)) → X = deus e Y = filho(deus). ⊨ ama(X,filho(X)).

Porém:

igual(X,X), igual(bola, bala), não podem ser unificadas, X = bola → igual(bola, bola).

Sendo bola ≠ bala, não é possível efetuar a substituição que torne as fórmulas atômicas idênticas. A substituição, também, não é possível quando a variável é substituída por uma função que tenha a própria variável ($X = f(X)$).

4.1 Exercícios

1. cor(sapato(X), branco) e cor(sapato(suspeito), Y)
2. mora(X, casa(mae(X))) e mora(joana, Y)
3. primo(X, Y) e prima(A,B)
4. ponto(X, 2,Z) e ponto(1,W)
5. p(f(Y), Y,X) e p(X, f(a), f(Z))

Raciocínio automatizado

Um dos principais algoritmos para raciocínio automatizado é denominado SLD-resolução. Trata-se de um método computacional de refutação que apresenta as seguintes características:

1. Restringe-se à uma classe de fórmulas denominada cláusulas de Horn.
2. Emprega resolução e unificação como regra de inferência.
3. Adota uma estratégia de busca em profundidade para controlar as inferências.
4. Introduz o conceito de predicado computável.
5. Introduz o conceito de negação por falha finita.

Cláusulas de Horn

Cláusulas de Horn são fórmulas da forma $\phi \leftarrow \phi_1, \ldots \phi_n$, sendo $n \geq 0$, em que o literal ϕ é uma conclusão e os literais ϕ_i são condições.

$$\phi \leftarrow \begin{cases} n = 0 & fato \\ n > 0 & regra \\ n = \emptyset & \texttt{contradição}. \end{cases}$$

$$\emptyset \leftarrow \phi_1, \ldots, \phi_n, \text{é} umaconsulta.$$

4.2 Inferência com cláusulas de Horn

As inferências são realizadas sempre entre uma **consulta e um fato** ou entre uma **consulta e uma regra**, sendo que o resultado da inferência, denominado resolvente, é sempre uma consulta ou a clásula vazia.

Caso 1 : se o fato α_0 unifica-se com o primeiro literal da consulta $\leftarrow \beta_1, \ldots, \beta_n$, sob o conjunto de substituições θ, então a resolvente será a consulta:

$$\beta'_2, \beta'_n, \texttt{ em que } \beta'_i \texttt{ é uma instância de } \beta_i, \texttt{ sob a subtituição } \theta \quad (1)$$

Caso 2 : se a conclusão da regra $\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_n$, sob o conjunto de substituição θ, então a resolvente será a consulta:

$$\leftarrow \alpha'_1, \ldots, \alpha'_m, \beta'_2, \beta'_n, \texttt{ em que } \alpha'_i \texttt{ é uma instância de } \alpha_i, e\beta'_i, \texttt{ sob a substituição } \theta. \quad (2)$$

A função unifica(α, β, γ) tenta unificar a cláusula α com a consulta β, resultando na nova consulta γ (que pode ser a cláusula vazia). Caso a unificação seja bem sucedida, essa função devolve verdade; caso contrário, devolve falso.

4.3 Algoritmo de busca SLD-Resolução

O algoritmo SLS-Resolução controla a aplicação da regra de inferência, realizando uma busca em profundidade. Ao derivar a cláusula vazia, o algoritmo sinaliza sucesso e apresenta como solução a composição das substituições efetuadas no caminho percorrido até a cláusula vazia.

Ao atingir um ponto em que a consulta não pode ser unificada com nenhuma cláusula, o algoritmo sinaliza fracasso e tenta retroceder na busca. Sejam Δ um conjunto de cláusulas de Horn (programa lógico) e β uma consulta. O algoritmo a seguir responde à consulta com base em Δ:

SLD-Resolução(ϕ)

1. se ϕ é a cláusula vazia então devolva sucesso
2. para cada cláusula (de cima para baixo) faça
3. se unifica(ϕ) e SLD-Resolução(ϕ) =sucesso então
4. exiba a composição das substituições efetuadas
5. caso contrário devolva fracasso

Exemplo :
Considere as seguintes cláusulas

Saudável :
bebe(zé, pinga).
bebe(manuel, água).
vivo(manuel).
saudável(X):- bebe(Y,X), vivo(Y).
Para descobrir o que é saudável, entra-se com a consulta: **?- saudavel(R)**.

$\sqrt{[0]} \leftarrow saudavel(R)$

saudavel(R) ← bebe(Y,R), vivo(Y) {X=R}

[1] ← bebe(Y,R), vivo(Y)

bebe(zé,pinga) {Y=zé,R=pinga}	bebe(manuel,água) {Y=manuel,R=água}
[2] ← vivo(zé)	[3] ← vivo(manuel)
fracasso	vivo(manuel) ← {}
[4] ←	sucesso

Nessa árvore de refutação, as resolventes estão numeradas na ordem em que são geradas pelo algoritmo SLD-Resolução. O primeiro caminho percorrido termina com fracasso, então o algoritmo retrocede e tenta outro caminho. Ao encontrar a cláausula vazia, ele sinaliza sucesso e exibe a resposta derivada das substituições efetuadas no caminho percorrido até a cláusula vazia. No caso do exemplo, a resposta encontrada foi R = água.

O que Ana Fala :
nasceu(ana, brasil).
nasceu(yves, france).
idioma(brasil, portugues).
idioma(franca, frances).
estudou(ana, frances).
fala(A,C):- nasceu(A,B), idioma(B,C).
fala(D,E):- estudou(D,E.).
Para descobrir que idiomas Ana fala, aplica-se a consulta: **?- fala(ana,R)**.

[0] ← fala(ana,R)

fala(ana,R) ← nasceu(ana,B), idioma(B,R) {A=ana,C=R}	fala(ana,R) ← estudou(ana,R) {D=ana,E=R}
[1] ← nasceu(ana,B), idioma(B,R)	[3] ← estudou(ana,R)
nasceu(ana,brasil) {B=brasil}	estudou(ana,frances) {R=frances}
[2] ← idioma(brasil,R)	sucesso
idioma(brasil,portugues) {R=portugues}	
sucesso	

Para essa consulta, o algoritmo é bem sucedido nos dois caminhos percorridos e apresenta R = portugues e R = frances como respostas. Veja como isso faz sentido: declaramos no programa lógico que uma pessoa fala um idioma se ela nasce num país onde se fala esse idioma ou se ela estuda esse idioma. Então, como Ana nasceu no Brasil, o algoritmo conclui que Ana fala português. Além disso, como Ana estudou francês, o algoritmo conclui que Ana também fala francês. Vejamos, agora, as respostas para a consulta **?- fala(yves,R)**:

[0] ← fala(yves,R)

fala(yves,R) ← nasceu(yves,B), idioma(B,R) {A=yves,C=R}

[1] ← nasceu(yves,B), idioma(B,R)

nasceu(yves,franca) {B=franca}

[2] ← idioma(franca,R)

idioma(franca,frances) {R=frances}

sucesso

fala(yves,R) ← estudou(yves,R) {D=yves,E=R}

[3] ← estudou(yves, R)

fracasso

Bem, como Yves não estudou nenhum idioma, era de se esperar que ele falasse apenas francês, que é a sua língua nativa. Portanto, o algoritmo encontrará apenas uma resposta: R = frances.

Exercícios

1. %Base de fatos
   gosta(ary, eva) ←
   gosta(ivo, ana) ←
   gosta(ivo, eva) ←
   gosta(eva, ary) ←
   gosta(ana, ary) ←
   namora(A,B) ← gosta(A,B), gosta(B,A).

   Demonstre como o algoritmo Sld-Resolução responde nas seguintes consultas:

   (a) Ivo gosta de quem?

   (b) Quem gosta de Ary?

   (c) Ivo namora com Eva?

   (d) Ary namora com quem?

   (e) Eva namora com Ary?

2. %Base de fatos
   pai(adao, cain) ←
   pai(adao, abel) ←
   pai(adao, seth) ←
   pai(seth, enos) ←
   avo(X,Z) ← pai(X, Y), pai(Y,Z)

   Demonstre como o algoritmo Sld-Resolução responde nas seguintes consultas:

   (a) Quem é pai de Abel?

   (b) Adão é pai de quem?

   (c) Quem é avô de Enos?

   (d) Seth é avô de alguém?

3. Demonstre, com base no programa a seguir, o algoritmo Sld-Resolução encontra três respostas para consulta $\leftarrow$ irmao(cain,R).

   (a) pai(adao, cain) $\leftarrow$
   (b) pai(adao, abel) $\leftarrow$
   (c) pai(adao, seth) $\leftarrow$
   (d) irmao(X, Y) $\leftarrow$ pai(Z,X), pai(Z, Y)

Referências

AMBL87 AMBLE, T. **Logic Programming and Knowledge Engineering**, Addison-Wesley, 1987.

BRAT90 BRATKO, I. **Prolog - Programming for Artificial Intelligence**, Addison-Wesley, 1990.

GENE88 GENESERETH, M. R. and NILSSON, N. J. **Logical Fundations of Artificial Intelligence**, Morgan Kaufmann Publishers, 1988.

REIT78 REITER, R. **On Closed Word Data Bases**, In H. Gallaire and J. Minker, *Logic and Data Bases*, pages 55-76, Plenum Press, NY, 1978.

RICH95 RICH, E. and KNIGHT, K. **Inteligência Artificial**, 2a ed., Makron Books, 1995.

RUSS95 RUSSELL, S. and Norvig, P. **Artificial Intelligence - a modern approach**, Prentice-Hall, 1995.

STER86 STERLING, L. and SHAPIRO, E. **The Art of Prolog**, MIT Press, 1986.