INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

PARTE 2. LÓGICA DE PREDICADOS

2.0. ASPECTOS DA LÓGICA

2.0.0. Introdução

A Inteligência Artificial (IA) deve ter mecanismos para a representação de fatos. A abordagem mais comum para este fim é usar a linguagem da *lógica*. Outros formalismos serão discutidos depois. A prova de teoremas era um dos primeiros domínios nos quais as técnicas de IA eram exploradas. Vai-se usar neste capítulo os seguintes símbolos lógicos padrões: "$\rightarrow$" (implicação), "$\neg$" (negação), "$\vee$" (disjunção), "$\wedge$" (conjunção), "$\forall$" (quantificação universal = "para todos") e "$\exists$" (quantificação existencial = "existe").

2.0.1. Representando fatos simples através da lógica

Para representar o conhecimento do mundo que um sistema de IA necessita, explora-se o uso da lógica proposicional. Vai-se representar os fatos do mundo real através das *fórmulas bem formadas ("fbf's")* ou *proposições lógicas*, como mostrado abaixo:

Está chovendo.
CHOVENDO
Está ensolarado.
ENSOLARADO
Se está chovendo, então não está ensolarado.
CHOVENDO ® ØENSOLARADO

Se se quiser representar o fato óbvio de que:
"Sócrates é um homem"

Pode-se escrever:
SOCRATESHOMEM

Mas se se quiser representar
"Platão é um homem"

seria necessário escrever algo como:
PLATAOHOMEM

que seria uma asserção totalmente isolada, não sendo possível concluir sobre similaridades entre Sócrates e Platão. Seria melhor, portanto, escrever estes fatos como:

HOMEM(SOCRATES)
HOMEM(PLATAO)

já que a estrutura da representação reflete a estrutura do conhecimento. Veja a dificuldade em representar a sentença

"Todos os homens são mortais"

Poder-se-ia representá-la por

MORTALHOMEM

Mas esta forma não captura o relacionamento existente entre um indivíduo sendo um homem e um indivíduo sendo mortal. Para fazê-lo, necessita-se de variáveis e quantificação, a menos que se deseje escrever comandos separados sobre a mortalidade de todos os homens conhecidos.

Vai-se explorar o uso da lógica de predicados como uma forma de representar conhecimento. Considere o seguinte conjunto de sentenças:

1. Marco era um homem.
2. Marco era um pompeiano.
3. Todos os pompeianos eram romanos.
4. César era um soberano.
5. Todos os romanos ou eram leais a César ou o odiavam.
6. Todos são leais a alguém.
7. As pessoas somente tentam assassinar soberanos aos quais elas não são leais.
8. Marco tentou assassinar César.

Os fatos descritos por estas sentenças podem ser representados como um conjunto de fbf's na lógica de predicados como se segue:

1. Marco era um homem.
   homem(Marco)

Esta representação captura o fato crítico de Marco ser um homem. Não há, entretanto, a informação de tempo verbal. Mas para este exemplo, isto não é relevante.

2. Marco era um pompeiano.
   pompeiano(Marco)
3. Todos os pompeianos eram romanos.
   $\forall x: pompeiano(x) \rightarrow romano(x)$
4. César era um soberano.
   soberano(Cesar)
5. Todos os romanos ou eram leais a César ou o odiavam.
   $\forall x: romano(x) \rightarrow leal(x,Cesar) \vee odiar(x,Cesar)$
6. Todos são leais a alguém.
   $\forall x: \exists y: leal(x,y)$
7. As pessoas somente tentam assassinar soberanos aos quais elas não são leais.

$\forall x{:}\forall y{:}\ pessoa(x) \wedge tentarassassinar(x,y) \rightarrow \neg leal(x,y)$

8. Marco tentou assassinar César.

tentarassassinar(Marco,Cesar)

Através deste breve exemplo, foi possível perceber que não é tão simples passar sentenças do Português para a forma de comandos lógicos. Suponha que se deseje usar estes comandos para responder à questão

"Marco era leal a César?"

Parece que usando 7 e 8, dá para concluir que Marco não era leal a César (ignorando a distinção entre passado e presente). Agora, vai-se tentar produzir uma prova formal, raciocinando "para trás" (*backward*), a partir da meta desejada:

¬leal(Marco,Cesar)

A fim de provar a meta, necessita-se de regras de inferência para transformá-la em outra meta (ou possivelmente, um conjunto de metas) que pode, por sua vez, ser transformada, e assim por diante, até que não haja mais metas insatisfeitas. A figura 1 mostra uma tentativa para produzir uma prova da meta, reduzindo o conjunto de metas até o conjunto vazio.

O problema é que, ainda que se saiba que Marco era um homem, não há maneira de concluir que Marco era uma pessoa. Precisa-se adicionar a representação de um outro fato ao sistema:

¬leal(Marco,Cesar)
↑ (7, substituição)
pessoa(Marco) ∧
soberano(Marco) ∧
tentarassassinar(Marco,Cesar)
↑ (4)
pessoa(Marco)
tentarassassinar(Marco,Cesar)
↑ (8)
pessoa(Marco)

Fig. 1. Uma tentativa de prova para "¬leal(Marco,Cesar)"

9. Todos os homens são pessoas.

$\forall x{:}\ homem(x)$ ® $pessoa(x)$

Agora dá para satisfazer a última meta e produzir uma prova que Marco não era leal a César.

Deste exemplo simples, pode-se perceber três pontos importantes, na conversão de sentenças do português em comandos lógicos:

- Muitas sentenças do português são ambíguas (por exemplo, 5, 6 e 7 acima). A escolha da interpretação correta pode ser difícil.

- Existe freqüentemente uma escolha de como representar o conhecimento. Representações simples são desejáveis mas elas podem impedir certos tipos de raciocínio.

- Mesmo em situações muito simples, um conjunto de sentenças não parece conter toda a informação necessária para raciocinar sobre o tópico em questão. Para ser capaz de usar um conjunto de comandos efetivamente, é muitas vezes necessário ter acesso a um outro conjunto de comandos que representam fatos considerados óbvios demais para mencionar.

Um outro problema surge em situações onde não se conhece de antemão quais comandos deduzir. No exemplo apresentado, o objeto era responder a questão "Marco era leal a César?" Como um programa poderia decidir se ele deveria tentar provar

leal(Marco,Cesar)

ou

¬*leal(Marco,Cesar)*

2.0.2. Funções e predicados computáveis

No exemplo visto, todos as fatos simples eram expressos como combinações de predicados individuais, tais como:

tentarassassinar(Marco,Cesar)

Isto é interessante se o número de fatos não é muito grande ou se os fatos são suficientemente desestruturados. Mas suponha que se deseja expressar fatos simples, tais como os seguintes relacionamentos "maior-que" e "menor-que":

ma(1,0)	*me*(0,1)
ma(2,1)	*me*(1,2)
ma(3,2)	*me*(2,3)
	...

É óbvio que não é desejável ter que escrever a representação de cada um destes fatos individualmente. Neste caso é útil ampliar a representação usando os *predicados computáveis*.

É também interessante ter *funções computáveis* assim como os predicados. Então pode-se ser capaz de calcular a verdade de

ma(2+3,1)

Para isto, é necessário primeiro calcular o valor da função *mais*, dados os argumentos 2 e 3, e então enviar os argumentos 5 e 1 a *ma*.

O próximo exemplo mostra como estas idéias de funções e predicados computáveis podem ser úteis. Ele também faz uso da noção de igualdade e permite que objetos iguais sejam substituídos um pelo outro quando isto for interessante durante uma prova.

Considere o seguinte conjunto de fatos, novamente envolvendo Marco:

1. Marco era um homem.
   homem(Marco)
2. Marco era um pompeiano.
   pompeiano(Marco)
3. Marco nasceu em 40 D.C.
   *nascer(Marco,*40*)*
4. Todos os homens são mortais.
   $\forall x$: *homem(x)* $\rightarrow$ *mortal(x)*
5. Todos os pompeianos morreram quando o vulcão entrou em erupção em 79 D.C.
   *erupcao(vulcao,*79*)* $\wedge$ $\forall x$: [*pompeiano(x)* ® *morreu(x,*79*)*]
6. Nenhum mortal vive mais de 150 anos.
   $\forall x$:$\forall t1$:$\forall t2$: *mortal(x)* $\wedge$ *nasceu(x,t1)* $\wedge$ *ma(t2-t1,150)* $\rightarrow$ *morto(x,t2)*
7. Agora é 1997.
   agora = 1997
8. Vivo significa não morto.
   $\forall x$:$\forall t$:[*vivo(x,t)* $\rightarrow$ $\neg$*morto(x,t)*] $\wedge$ [$\neg$*morto(x,t)* $\rightarrow$ *vivo(x,t)*]
9. Se alguém morre, então ele está morto para sempre.
   $\forall x$:$\forall t1$:$\forall t2$: *morreu(x,t1)* $\wedge$ *ma(t2,t1)* $\rightarrow$ *morto(x,t2)*

1. *homem(Marco)*
2. *pompeiano(Marco)*
3. *nascer(Marco,*40*)*
4. $\forall x$: *homem(x)* $\rightarrow$ *mortal(x)*
5. *erupcao(vulcao,*79*)*
6. $\forall x$: [*pompeiano(x)* ® *morreu(x,*79*)*]
7. $\forall x$:$\forall t1$:$\forall t2$: *mortal(x)* $\wedge$ *nasceu(x,t1)* $\wedge$ *ma(t2-t1,150)* $\rightarrow$ *morto(x,t2)*
8. *agora* = 1997
9. $\forall x$:$\forall t$:[*vivo(x,t)* $\rightarrow$ $\neg$*morto(x,t)*] $\wedge$ [$\neg$*morto(x,t)* $\rightarrow$ *vivo(x,t)*]
10. $\forall x$:$\forall t1$:$\forall t2$: *morreu(x,t1)* $\wedge$ *ma(t2,t1)* $\rightarrow$ *morto(x,t2)*

Fig. 2. Um conjunto de fatos sobre “Marco”.

Um conjunto de todos os fatos representados está na figura 2. (A numeração mudou um pouco, pois a sentença 5 foi quebrada em duas partes.) Agora pode-se tentar responder à questão "Marco está vivo?" provando:

$\neg$*vivo(Marco,agora)*

As provas são mostradas nas figuras 3 e 4. O termo *nil* no final de cada prova indica que a lista de condições restantes a serem provadas está vazia e, portanto, a prova foi bem sucedida. Note que nestas provas quando um comando da forma:

$a \wedge b \rightarrow c$

foi usado, *a* e *b* são estabelecidos como submetas independentes. Pode-se satisfazer a meta

nasceu(Marco,t1)

usando o comando 3 fazendo a ligação de *t1* a 40, mas deve-se também ligar *t1* a 40 em
ma(agora-t1,150*)*

já que os dois *t1*'s são a mesma variável no comando 4, da qual as duas submetas vieram. Um bom procedimento de prova computacional deve incluir uma forma de determinar que uma unificação existe e também uma forma de garantir substituições uniformes durante a prova.

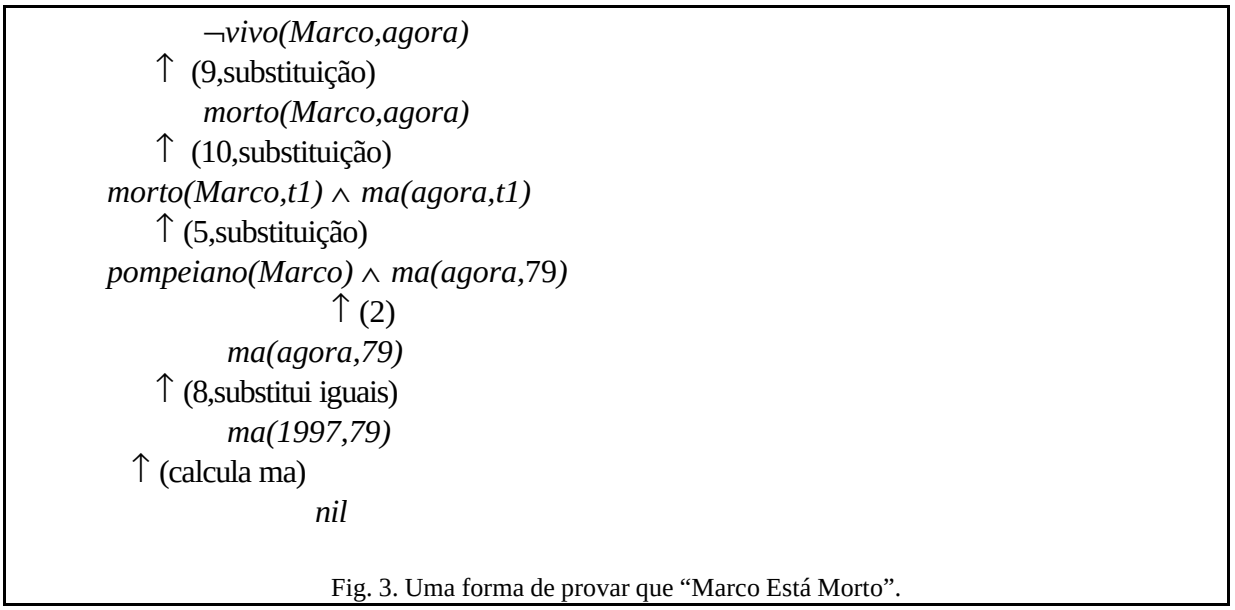
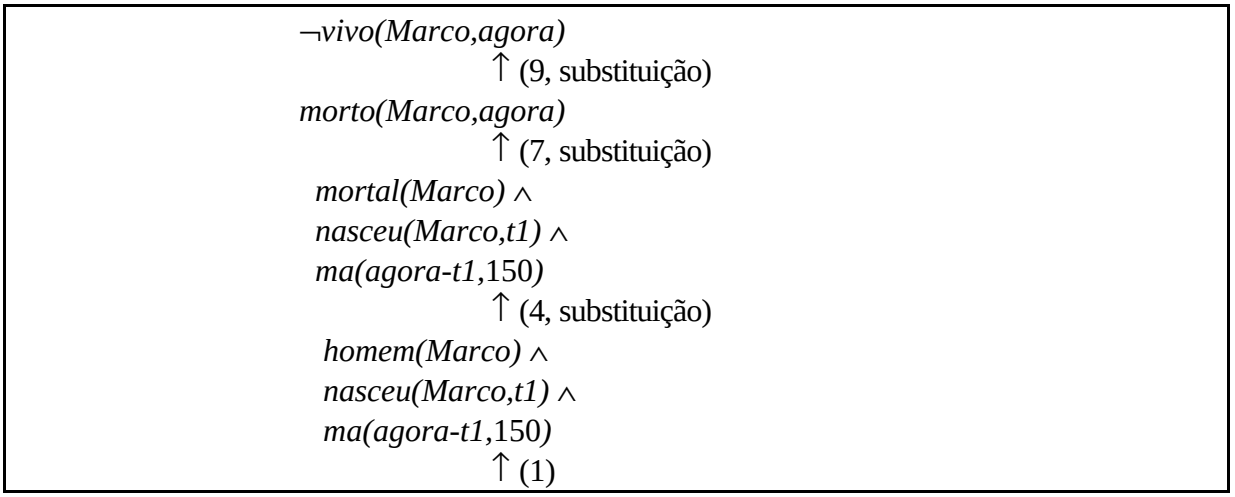


Fig. 3. Uma forma de provar que “Marco Está Morto”.



nasceu(Marco,t1) ∧
ma(agora-t1,150*)*
↑ (3)
ma(agora-40,150*)*
↑ (8)
*ma(*1997-40,150*)*
↑ (calcula subtração)
*ma(*1957,150*)*
↑ (calcula ma)
nil

Fig. 4. Uma outra forma de provar que "Marco Está Morto".

Olhando as provas mostradas, duas coisas devem ficar claras:

- Mesmo conclusões simples podem precisar de muitos passos para provar.

- Vários processos, tais como unificação, substituição e aplicação de *modus ponens* são envolvidas na produção de uma prova.

2.1. LÓGICA SENTENCIAL OU CÁLCULO PROPOSICIONAL

A motivação para se estudar lógica num curso de Inteligência Artificial é de usar esta linguagem artificial como uma ferramenta para fazer dedução lógica em base de conhecimento.

2.1.1. Sintaxe das linguagens proposicionais

Dado um alfabeto α, uma *cadeia* sobre α é uma seqüência de símbolos de α. Uma *linguagem* sobre α é um conjunto de cadeias de α.

Definição 1.1: Um *alfabeto proposicional* α consiste de

símbolos lógicos:
pontuação: (,)
conectivos: ¬ (negação)
∧ (conjunção)
∨ (disjunção)
→ (implicação)
↔ (bi-implicação ou bi-condicional)

símbolos não-lógicos: um conjunto finito **P** de *símbolos proposicionais* diferentes dos símbolos lógicos. Ex. P, Q, etc.

Todas as definições a seguir têm por base o alfabeto proposicional α.

Definição 1.2: O conjunto de *fórmulas proposicionais* (ou simplesmente *fórmulas*) é o menor conjunto de cadeias satisfazendo às seguintes condições:

(i) todo símbolo proposicional é uma fórmula;
(ii) se P e Q são fórmulas, então ($\neg$ P), (P $\wedge$ Q), (P $\vee$ Q), (P $\rightarrow$ Q) e (P $\equiv$ Q) também são fórmulas.

Definição 1.3: Uma fórmula Q é uma *subfórmula* de uma fórmula P se e somente se Q é uma subcadeia de P. Em outras palavras, toda subcadeia de P que ainda for uma fórmula é uma subfórmula.

Definição 1.4: A *linguagem proposicional*, denotada por L(α), é o conjunto das fórmulas proposicionais.

2.1.2. Semântica das linguagens proposicionais

As fórmulas de uma linguagem proposicional, incluindo os símbolos proposicionais, terão como significado os valores-verdade *FALSO* ou *VERDADEIRO*, abreviados *F* e *V*, respectivamente.

Definição 1.5: Seja **P** o conjunto de símbolos proposicionais de α. Uma *atribuição de valores-verdade* para α (ou simplesmente uma *atribuição* para α) é uma função $a: \mathbf{P} \rightarrow \{F,V\}$.

Definição 1.6: Seja *a* uma atribuição de valores-verdade. A *função de avaliação* para L(α) *induzida* por *a* é a função v: L(α) $\rightarrow \{F,V\}$ definida da seguinte forma (Obs: note que a *atribuição* é para símbolos proposicionais e a *avaliação* é para fórmulas):

v(A) = a(A), se A é um símbolo proposicional

$v(\neg \text{P}) = V$, se v(P) = F
$= F$, se v(P) = V

v(P $\wedge$ Q) $= V$, se v(P) = v(Q) = V
$= F$, em caso contrário

v(P $\vee$ Q) $= F$, se v(P) = v(Q) = F

= *V*, em caso contrário

$v(P \rightarrow Q)$ = *F*, se $v(P) = V$ e $v(Q) = F$
= *V*, em caso contrário

$v(P \leftrightarrow Q)$ = *V*, se $v(P) = v(Q)$
= *F*, em caso contrário

*Definição 1.7***:** Sejam **P** e **Q** conjuntos de fórmulas em $L(\alpha)$ e R uma fórmula em $L(\alpha)$.

(a) R é *verdadeira* em uma atribuição de valores-verdade *a* se e somente se $v(R) = V$. Em caso contrário, R é *falsa*.

(b) R é uma *tautologia* se e somente se, para toda atribuição de valores-verdade *a*, $v(R) = V$. Isso corresponde a uma coluna inteira de *V* (verdadeiro) na tabela-verdade.

(c) uma atribuição de valores-verdade *a satisfaz* a **P**, ou *a* é um *modelo* para **P**, se e somente se, para toda fórmula S em **P**, $v(S) = V$. Isso corresponde a uma linha inteira de *V* (verdadeiro) na tabela-verdade.

(d) **P** é *satisfatível* se e somente se existe uma atribuição de valores-verdade *a* que satisfaz **P**. Em caso contrário, **P** é *insatisfatível*.

(e) R é uma *conseqüência lógica* de **P**, ou **P** *implica logicamente* R (notação: **P** |= R), se e somente se, para toda atribuição de valores-verdade *a*, se *a* satisfaz **P** então *a* satisfaz R.

(f) **P** é *tautologicamente equivalente* a **Q** (notação: **P** |=| **Q**) se e somente se toda fórmula de **Q** for conseqüência lógica de **P** e vice-versa.

2.1.3. O Método da tabela-verdade

O método da tabela-verdade baseia-se na observação de que, se **P** é um conjunto finito de fórmulas e Q é uma fórmula, há um número finito de símbolos proposicionais ocorrendo em Q e nas fórmulas em **P**. Logo, há um número finito de atribuições de valores-verdade distintas para estes símbolos. Assim, para decidir se Q é uma conseqüência lógica de **P**, basta enumerar todas estas atribuições e, para cada uma delas que satisfizer a todas as fórmulas em **P**, testar se ela também satisfaz a Q. Se esta condição for sempre observada, então pode-se afirmar que Q é uma conseqüência lógica de **P**. Note que, se *a* não satisfizer a alguma fórmula em **P**, o valor que *a* atribui a Q não importará, pela forma como a implicação lógica foi definida. Este método também pode ser usado para decidir se uma fórmula é satisfatível ou se dois conjuntos finitos de fórmulas são tautologicamente equivalentes.

Exemplo: Seja **P** o seguinte conjunto de fórmulas:

1. $A \rightarrow \neg B$

2. $B \wedge A$
3. B,

seja **Q** o seguinte conjunto de fórmulas:

1. $A \vee B$
2. $B \rightarrow A$

e seja R a fórmula $\neg B \rightarrow A$

vamos construir a tabela-verdade para estes conjuntos

		1	2	3	4	5	6
A	B	$A \rightarrow \neg B$	$B \wedge A$	B	$A \vee B$	$B \rightarrow A$	$\neg B \rightarrow A$
F	*F*	*V*	*F*	*F*	*F*	*V*	*F*
F	*V*	*V*	*F*	*V*	*V*	*F*	*V*
V	*F*	*V*	*F*	*F*	*V*	*V*	*V*
V	*V*	*F*	*V*	*V*	*V*	*V*	*V*

Observe, que para o conjunto **P** (colunas 1, 2 e 3), não existe nenhuma atribuição (linha da tabela-verdade) que o satisfaça, ou seja, não existe nenhuma atribuição que resulta sempre em verdadeiro para **P**. Logo, **P** é insatisfatível. Já para o conjunto **Q** (colunas 4 e 5), existem duas atribuições (A=*V*, B=*F* e A=*V*, B=*V*) que satisfazem este conjunto, logo **Q** é satisfatível. Como a fórmula R (coluna 6) também é verdadeira para as duas atribuições que satisfazem **Q**, diz-se que R é conseqüência lógica de **Q**, ou que **Q** implica logicamente R. Observe também que **P** e **Q** não são tautologicamente equivalentes.

2.2. Lógica de Primeira Ordem

A Lógica de primeira ordem, ou Cálculo de Predicados de Primeira Ordem (CPPO) pode ser caracterizada como um sistema formal apropriado a definição de teorias do universo de discurso da Matemática. A motivação para se estudar esta lógica é que a lógica sentencial não dá conta da representação de frases do tipo:

Sócrates é homem.
Platão é homem.
Todos os homens são mortais.

2.2.1. Sintaxe das linguagens de primeira ordem

*Definição 2.1***:** Um *alfabeto de primeira ordem* α consiste de:

símbolos lógicos:
pontuação: (,)

conectivos: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
quantificadores: $\forall$ (quantificador universal)
$\exists$ (quantificador existencial)
variáveis: um conjunto de símbolos distintos dos demais, por convenção, representadas por letras minúsculas do final do alfabeto: *x*, *y*, *z*, etc.
símbolo de igualdade (opcional): =

símbolos não-lógicos: Um conjunto, possivelmente vazio, de *constantes* distintas dos demais símbolos, por convenção, representadas por letras minúsculas do início do alfabeto: a, b, c, etc.

Para cada n > 0, um conjunto, possivelmente vazio, de *símbolos funcionais n-ários* distintos dos demais símbolos (O símbolo funcional representa a função da computação, que efetua um cálculo e retorna um valor), por convenção, representados pelas letras minúsculas a partir da letra f (de funcional): f, g, h, etc.

Para cada n > 0, um conjunto, possivelmente vazio, de *símbolos predicativos n-ários* distintos dos demais símbolos (O símbolo predicativo, ou *predicado*, representa uma relação entre objetos, ou seja sua avaliação será verdadeira ou falsa), por convenção, representados por letras maiúsculas: P, Q, R, etc.

Todas as definições a seguir fazem referência ao alfabeto de primeira ordem α.

*Definição 2.2***:** O conjunto de *termos de primeira ordem* (ou simplesmente *termos*) é o menor conjunto satisfazendo às seguintes condições:

(i) toda variável é um termo;
(ii) toda constante é um termo;
(iii) se $t_1 ,...,t_n$ são termos e f é um símbolo funcional n-ário, então $f(t_1 ,...,t_n)$ também é um termo.

*Definição 2.3***:** O conjunto de *fórmulas* é o menor conjunto satisfazendo às seguintes condições:

(i) se $t_1 ,...,t_n$ são termos e P é um símbolo predicativo n-ário, então $P(t_1 ,...,t_n)$ é uma fórmula, chamada de *fórmula atômica*. (Observe que os argumentos do predicado são termos, ou seja, podem ser constantes, variáveis o símbolos funcionais, mas *nunca* outros predicados).

(ii) se $t_1 ,...,t_n$ são termos e "=" é um símbolo de α, então $(t_1 = t_2)$ é uma fórmula, também chamada de *fórmula atômica*.

(iii) se P e Q são fórmulas, então $(\neg P)$, $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$ e $(P \leftrightarrow Q)$ também são fórmulas.

(iv) se P é uma fórmula e *x* é uma variável, então $\forall x(P)$ e $\exists x(P)$ também são fórmulas.

Definição 2.4: Uma fórmula Q é uma *sub-fórmula* de uma fórmula P se e somente se Q é uma subcadeia de P.

Definição 2.5: A *linguagem de primeira ordem*, denotada por L(α), é o conjunto de termos e fórmulas de primeira ordem.

Definição 2.6:(a) Em uma fórmula da forma $\forall x$(Q) (ou da forma $\exists x$(Q)), Q é o *escopo* de $\forall x$ (ou de $\exists x$).

(b) Uma ocorrência de uma variável *x* em uma fórmula P é *ligada* em P, se a ocorrência se dá em uma subfórmula de P da forma $\forall x$(Q) ou da forma $\exists x$(Q). Caso contrário, a ocorrência de *x* é *livre*. Ou seja, a ocorrência de *x* é ligada se *x* ocorrer dentro do escopo de seu quantificador.

(c) Uma variável *x* é *livre* em P se existe uma ocorrência livre de *x* em P.

(d) Uma fórmula P é uma *sentença* se e somente se nenhuma variável ocorre livre em P. Por exemplo, $\forall x$ (A(x) $\wedge$ B(y)) $\rightarrow$ C(x) não é uma sentença, pois a variável *y* ocorre livre (não existe quantificador para ela, e a segunda ocorrência de *x* também é livre (está fora do escopo de $\forall x$).

Definição 2.7: (a) Dada uma fórmula P, com variáveis livres $x_1,...,x_n$, o *fecho universal* de P é a fórmula $\forall x_1 ... \forall x_n$ (P) e o *fecho existencial* de P é a fórmula $\exists x_1 ... \exists x_n$ (P).

(b) Uma fórmula P está na *forma normal prenex* se e somente se P for da forma Q(M) onde Q, o *prefixo* de P, é uma cadeia de quantificadores e M, a *matriz* de P, é uma fórmula sem ocorrências de quantificadores. Ou seja, forma normal prenex é quando se tem todos os quantificadores à esquerda da fórmula. Por exemplo, a fórmula, que é uma sentença, $\forall x \exists y$ (A(x,y) $\rightarrow$ B(x)) está na forma normal prenex, mas a fórmula, que também é uma sentença, $\forall x$ A(x) $\rightarrow \exists y$ B(y) não está.

(c) Uma fórmula P é uma *conjunção* se e somente se, omitindo-se os parênteses, for da forma $P_1 \wedge ... \wedge P_n$.

(d) Uma fórmula P é uma *disjunção* se e somente se, omitindo-se os parênteses, for da forma $P_1 \vee ... \vee P_n$.

(e) Uma fórmula P está em *forma normal conjuntiva* se e somente se estiver em forma normal prenex e a sua matriz for uma conjunção de disjunções de fórmulas atômicas, negadas ou não. Por exemplo, $\forall x \exists y$ ((A(x,y) $\vee$ B(x)) $\wedge$ (B(y) $\vee$ A(y,y)), está na forma normal conjuntiva. E $\forall x \exists y$ (A(x,y) $\vee$ B(x)) também está. E $\forall x \exists y$ (A(x,y) $\wedge$ B(y)) também, assim como $\forall x \exists y$ (A(x,y)). Por que ?

Regra de Inferência (**Modus Ponens**):
MP. a partir de P e de (P $\rightarrow$ Q), deduza Q

2.3. Notação Clausal

Definição 3.1:

(a) Um *literal positivo* é uma fórmula atômica.
(b) Um *literal negativo* é a negação de uma fórmula atômica.
(c) Um *literal* é ou um literal positivo ou um literal negativo.
(d) Dois literais têm *sinais opostos* se e somente se um deles for positivo e o outro for negativo.
(e) Dois literais são *complementares* se e somente se um deles for a negação do outro.
(f) Uma fórmula atômica *F* é o *átomo* de um literal L, denotado por |L|, se e somente se L for *F* ou $\neg F$.

Definição 3.2: Uma *cláusula* é ou uma seqüência não vazia de literais ou a *cláusula vazia*, denotada por "□".

Definição 3.3: A *linguagem de cláusulas* é o conjunto de todas as cláusulas.

Definição 3.4: Uma interpretação I *satisfaz* uma cláusula não vazia C (denotado por I |= C) se e somente se I satisfaz a sentença *F* definida como

$$\forall x_1 \ldots \forall x_m \, (L_1 \vee \ldots \vee L_n)$$

onde $x_1, \ldots, x_m$ são as variáveis ocorrendo em C e $L_1, \ldots, L_n$ são os literais de C. Diz-se ainda que C e *F* são *equivalentes*. Por convenção, a cláusula vazia é sempre insatisfatível. Com esta definição, está-se querendo dizer que, apesar da cláusula, por definição, ser uma seqüência de literais, estes literais estão ligados através de disjunções, muito embora alguns autores não as tornem explícitas na sua representação. Pode-se dizer então, que uma *cláusula é uma disjunção de literais*.

3.A. Representação clausal de fórmulas

Definição 3.5: Um conjunto de cláusulas **S** é uma *representação clausal* para uma fórmula P se e somente se P é satisfatível se e somente se **S** é satisfatível.

A obtenção da representação clausal de uma fórmula é um processo mecânico, como descrito a seguir:

Algoritmo 3.1: Algoritmo de Representação Clausal

entrada: uma fórmula P
saída: uma representação clausal **S** para P

(a) Tome o fecho existencial de P

Se P contiver uma variável livre x, substitua P por $\exists x(P)$. Repita este passo até que a fórmula não tenha variáveis livres.

(b) Elimine quantificadores redundantes

Elimine todo quantificador "$\forall x$" ou "$\exists x$" que não contenha nenhuma ocorrência livre de x no seu escopo. (Ou seja, elimine todo quantificador desnecessário).

(c) Renomeie variáveis quantificadas mais de uma vez.

Se houver dois quantificadores governando a mesma variável, substitua a variável de um deles e todas as suas ocorrências livres no escopo do quantificador por uma nova variável que não ocorra na fórmula. Repita o processo até que todos os quantificadores governem variáveis diferentes.

(d) Elimine os conectivos "$\rightarrow$" e "$\nleftrightarrow$"

Substitua:

$(Q \rightarrow R)$	por	$(\neg Q \vee R)$
$(Q \leftrightarrow R)$	por	$(\neg Q \vee R) \wedge (Q \vee \neg R)$
$\neg(Q \rightarrow R)$	por	$(Q \wedge \neg R)$
$\neg(Q \leftrightarrow R)$	por	$(Q \wedge \neg R) \vee (\neg Q \wedge R)$

(e) Mova "$\neg$" para o interior da fórmula

Até que cada ocorrência de "$\neg$" preceda imediatamente uma fórmula atômica, substitua:

$\neg\forall x(Q)$	por	$\exists x(\neg Q)$
$\neg\exists x(Q)$	por	$\forall x(\neg Q)$
$\neg(Q \wedge R)$	por	$(\neg Q \vee \neg R)$
$\neg(Q \vee R)$	por	$(\neg Q \wedge \neg R)$
$\neg\neg Q$	por	Q

(f) Mova os quantificadores para o interior da fórmula (opcional)

Substitua, caso x não seja livre em R,

$\forall x\ (Q \vee R)$ por $\forall x\ (Q) \vee R$
$\forall x\ (R \vee Q)$ por $R \vee \forall x(Q)$
$\forall x\ (Q \wedge R)$ por $\forall x\ (Q) \wedge R$
$\forall x\ (R \wedge Q)$ por $R \wedge \forall x(Q)$

$\exists x\ (Q \vee R)$ por $\exists x(Q) \vee R$
$\exists x\ (R \vee Q)$ por $R \vee \exists x(Q)$
$\exists x\ (Q \wedge R)$ por $\exists x(Q) \wedge R$
$\exists x\ (R \wedge Q)$ por $R \wedge \exists x(Q)$

(g) Elimine os quantificadores existenciais

Seja Q a fórmula corrente. Crie a nova fórmula corrente substituindo a subfórmula de Q da forma '$\exists y$(R)", que se situa mais à esquerda, por "R[y/f(x_1,...,x_n)]", onde x_1,...,x_n é uma lista de todas as variáveis livres de '$\exists y$(R)" e "f" é qualquer símbolo funcional n-ário que não ocorre em Q. Quando não houver variáveis livres em '$\exists y$(R)", substitua "$\exists y$(R)" por "R[y/c]", onde "c" é uma constante que não ocorre em Q. Repita o processo até que todos os quantificadores existenciais tenham sido eliminados.

Na verdade, deve-se pegar o escopo do quantificador existencial e verificar se, além da variável quantificada existencialmente, existe alguma variável ocorrendo livre neste escopo? Se existir, substitui-se a variável existencial por uma função de todas as variáveis que ocorrem livre. Se não, substitui-se a variável existencial por uma constante. A explicação para este procedimento pode ser entendida através do seguinte exemplo. Para a sentença ambígua da linguagem natural

Todo homem ama uma mulher.

Esta sentença tem pelo menos duas leituras possíveis. Numa, cada homem ama a sua mulher, portanto para cada homem (são todos) existe uma mulher que ele ama. Já numa outra leitura, existe uma única mulher, que é representada por uma constante a, que todos os homens amam. Logo as leituras gerariam as seguintes fórmulas:

$$\forall x \exists y\ ((H(x) \wedge M(x)) \rightarrow A(x, y))$$
$$\exists y \forall x\ ((H(x) \wedge M(x)) \rightarrow A(x, y))$$

Observe que a diferença das interpretações está apenas nos escopos dos quantificadores. A primeira fórmula gerará uma representação, onde y será substituída por um f(x), ou seja, cada homem tem a sua mulher, portanto y que é mulher é função de x, que é homem. Na segunda fórmula o y será substituído por uma constante, pois é a mesma mulher que todos os homens amam.

Exemplo. Na fórmula $\exists x \forall y\ ((P(x) \wedge Q(y))$, substitui-se a variável x por uma constante a, pois no escopo do quantificador existencial, que é $\forall y\ ((P(x) \wedge Q(y))$, nenhuma variável além do x ocorre livre. Logo, a fórmula fica $\forall y\ ((P(a) \wedge Q(y))$. Já na fórmula $\forall y \exists x\ (P(x) \vee Q(y))$, substitui-se

x por *f(y)*, pois no escopo do quantificador existencial que é $(P(x) \vee Q(y))$, além do *x*, o *y* ocorre livre, logo substitui-se a variável *x* por uma função da variável livre *y*, ou seja, *f(y)*. Assim, a fórmula fica $\forall y\ (P(f(y)) \vee Q(y))$.

Os novos símbolos funcionais assim introduzidos são chamados de *funções de Skolem* e o processo de substituição é chamado de *Skolemização*.

(h) Obtenha a forma normal prenex

Mova os quantificadores universais para a esquerda.

(i) Obtenha a forma normal conjuntiva

Até que a matriz da fórmula seja uma conjunção de disjunções, substitua:

$(Q \wedge R) \vee S$ por $(Q \vee S) \wedge (R \vee S)$
$Q \vee (R \wedge S)$ por $(Q \vee R) \wedge (Q \vee S)$

(j) Simplifique (opcional)

Transforme a fórmula Q resultante do passo (i) em outra mais simples S tal que S ainda esteja em forma normal conjuntiva (sem quantificadores existenciais) e Q seja satisfatível se e somente se S o for.

(k) Obtenha a representação clausal

A representação clausal **S** da fórmula inicial P será o conjunto das cláusulas da forma $L_1 ... L_n$ tais que $L_1 \vee ... \vee L_n$ é uma disjunção da matriz da fórmula resultante do passo anterior. Cada cláusula deve ficar numa linha (disjunção de literais) e cláusulas em linhas diferentes pressupõe conjunção de cláusulas.

EXERCÍCIOS PROPOSTOS

2.1. Considere as seguintes frases:

a) Josualdo gosta de todos os tipos de alimentos.
b) Maçãs são alimentos.
c) Galinha é alimento.
d) Qualquer coisa que qualquer um coma e não morra é alimento.
e) Velasquez come amendoim e ainda está vivo.
f) Solange come tudo o que Velasquez come.

a) Traduza essas frases em fórmulas da lógica de 1ª ordem.

b) Converta as fórmulas da parte (a) em cláusulas.

2.2. O que está errado com o seguinte argumento?

a) Os homens estão amplamente distribuídos sobre a Terra.
b) Sócrates é um homem.
c) Portanto, Sócrates está amplamente distribuído sobre a Terra.

Como os fatos representados por estas frases devem ser colocados na lógica para que este problema não surja?

2.3. Converta as seguintes sentenças em forma clausal:

a) $\forall x \forall y\ (\neg Q(x,y) \rightarrow \neg P(x,y))$
b) $\forall x \forall y\ (P(x,y) \rightarrow (Q(x,y) \wedge R(x,y)))$
c) $\forall x \forall y\ (P(x,y) \rightarrow Q(x,y))$
d) $\neg \forall x \exists y\ (P(x,y) \rightarrow Q(x,y))$
e) $\neg \forall x\ (P(x) \rightarrow (\forall y\ (P(y) \rightarrow P(f(x,y))) \wedge \neg \forall y\ (Q(x,y) \rightarrow P(y))))$
f) $\forall x\ (P(x) \rightarrow P(x))$
g) $\neg \forall x\ P(x) \rightarrow \exists x\ \neg P(x)$

2.4. Considere a interpretação *a* em lógica proposicional dada por:

$$a(P) = F \text{ e } a(Q) = a(R) = V$$

Qual o valor verdade da fórmula abaixo segundo esta interpretação?

$$(\neg P \wedge Q) \vee (P \rightarrow (Q \vee R))$$

2.5. Mostre que cada uma das seguintes fórmulas é uma tautologia:

a) $(P \rightarrow Q) \rightarrow ((R \vee P) \rightarrow (R \vee Q))$
b) $(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$

2.6. Prove que: $\forall z\ (Q(z) \rightarrow P(z)) \rightarrow (\exists x\ (Q(x) \rightarrow P(a)) \wedge (Q(x) \rightarrow P(b)))$

2.7. Mostre que:

a) $P \rightarrow (Q \rightarrow R) \equiv (P \wedge Q) \rightarrow R$
b) $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$
c) $P \vee (\neg Q \vee R) \equiv (\neg P \wedge Q) \rightarrow R$
d) $(P \rightarrow Q) \rightarrow P \equiv P$

BIBLIOGRAFIA

- Casanova, M. A. & Giorno, *F*. A. C. & Furtado, A. L. (1987). Programação em Lógica e a Linguagem Prolog. Editora Edgard Blücher Ltda.
- Rich. E. & Knight, K. (1994). Inteligência Artificial - 2ª. edição. Makron Books.