Inteligência Artificial
Apontamentos para as aulas

Luís Miguel Botelho

Departamento de Ciências e Tecnologias da Informação
Instituto Superior de Ciências do Trabalho e da Empresa

Julho de 2015

Representação de Conhecimento e Raciocínio em Lógica de Predicados de Primeira Ordem

Este componente dos apontamentos de Inteligência Artificial centra-se na representação de conhecimento e no raciocínio (inferência) em lógica de predicados de primeira ordem. Primeiro, apresenta-se a representação e o raciocínio no formato habitual da lógica de predicados. Depois, apresenta-se a representação na forma clausal da lógica de predicados de primeira ordem, a conversão para forma clausal, e a inferência por refutação na forma clausal.

Índice

1 Representação de conhecimento e raciocínio em lógica

A representação de conhecimento em lógica é um assunto relevante do ponto de vista dos sistemas baseados em conhecimento porque, embora a lógica não seja frequentemente usada diretamente para a representação de conhecimento em sistemas, ela constitui a base de outros métodos de representação muito usados na criação de sistemas, especialmente a programação em lógica e a representação através de regras condição-conclusão.

É fácil de ver que as seguintes expressões em lógica de predicados de primeira ordem, em programação em lógica, e representadas através de regras têm todas o mesmo significado e são na verdade bastante semelhantes entre si. Todas as expressões que se seguem representam o conceito *"Se x é um homem e tem um filho, então x é pai"*.

$\forall x \forall y$ Homem(x) $\wedge$ Filho(y, x) $\Rightarrow$ Pai(x)	Lógica de predicados de primeira ordem
pai(X):- homem(X), filho(Y, X).	Programação em lógica
IF homem(X) AND filho(Y,X) THEN pai(X)	Representação por regras condição-conclusão

As semelhanças entre a lógica de predicados, a programação em lógica e a representação através de regras vão para além da aparência das expressões que representam conhecimento. Como veremos mais adiante, os raciocínios que é possível fazer com cada um destes tipos de representação são também bastante semelhantes. No entanto, a expressividade da lógica de primeira ordem é maior do que a dos outros dois métodos de representação referidos (programação em lógica e regras).

Dadas as semelhanças entre a lógica de predicados, a programação em lógica e a representação através de regras condição-conclusão, iniciaremos o assunto dos métodos de representação do conhecimento por uma breve explicação da representação em lógica de predicados de primeira ordem. Este capítulo apresenta a sintaxe da lógica de predicados de primeira ordem acompanhada de uma breve explicação da sua semântica declarativa. Serão igualmente apresentadas regras de inferência do cálculo de predicados de primeira ordem. Finalmente, serão apresentados exercícios de representação e de derivação usando a lógica de predicados de primeira ordem. Resta-nos apenas esclarecer a razão de ser da designação *"lógica de predicados de primeira ordem"*.

A lógica de primeira ordem é uma lógica com variáveis quantificadas. Numa lógica de primeira ordem as variáveis podem apenas fazer o papel de entidades do domínio. Não podemos ter por exemplo uma variável no lugar de um predicado ou substituindo uma proposição. Há lógicas de ordem 0 e de ordem superior à primeira. A lógica proposicional é uma lógica sem quantificadores nem variáveis; diz-se que é uma lógica de ordem 0. O cálculo lambda por exemplo é uma lógica de ordem superior no sentido em que as suas variáveis podem substituir qualquer dos constituintes de uma proposição.

Quanto mais elevada é a ordem de uma lógica, mais expressiva é a linguagem mas mais difícil se torna criar mecanismos computacionais para a usarem em sistemas baseados em conhecimento e menos propriedades desejadas exibe.

A lógica de predicados é uma lógica em que as proposições se baseiam no conceito de relação ou de predicado. Um predicado aplicado ao seu conjunto de argumentos capta uma relação entre esses argumentos.

1.1 Sintaxe e semântica informal do cálculo de predicados de primeira ordem

Esta secção dá conta, de forma bastante informal, da sintaxe e da semântica da lógica de predicados de primeira ordem.

1.1.1 Sintaxe

As proposições da lógica de predicados de primeira ordem expressam afirmações sobre entidades de um dado domínio, as quais são representadas por termos. A sua sintaxe tem de regulamentar a estrutura dos termos e das proposições.

Proposições

A proposição mais simples da lógica de predicados de primeira ordem é a chamada proposição atómica. Chama-se proposição atómica porque não se pode decompor em proposições mais simples. Quando se decompõe uma proposição atómica nos seus constituintes obtêm-se símbolos ou estruturas simbólicas que não representam proposições.

Proposições Atómicas. Há dois tipos de proposição atómica: os símbolos proposicionais e as proposições relacionais.

Um símbolo proposicional é uma sequência de caracteres que exprime uma afirmação. Exemplos de símbolos proposicionais são *Está_a_Chover*, *Faz_Sol*, *Está_Mau_Tempo*, e *Servidor_desligado*.

Uma proposição relacional consiste da aplicação de um predicado aos seus argumentos. Um predicado poderá ter um ou mais argumentos. Exemplos de proposições atómicas deste tipo são *Filho(Ana, João)*, *Filho(João, André)*, *Pai(João)*, *Homem(João)*. Usando a terminologia da lógica de predicados de primeira ordem, os argumentos de um predicado são termos e representam entidades do domínio de que se fala.

Algumas proposições atómicas usando certos operadores relacionais escrevem-se de forma infixa em vez de se usar a forma prefixa mais habitual nos casos comuns. Exemplos são os operadores relacionais aritméticos e de conjuntos, tais como $<$, $=$, $>$, $\leq$, $\geq$, $\neq$, $\supset$, $\supseteq$, $\not\subset$, $\subset$, $\subseteq$, $\in$ e $\notin$. As proposições atómicas que usam estes operadores infixos escrevem-se como $5 < 10$ e $A \in$ Vogais.

Proposições Lógicas. Tendo definido as proposições atómicas podemos definir a sintaxe de todos os outros tipos de proposição, nomeadamente as chamadas proposições lógicas e as proposições quantificadas.

Sendo P e Q duas proposições arbitrárias quaisquer, as seguintes são proposições lógicas:

$\neg$P: Negação de P. $\neg$P é verdade se e só se P for falso.

P$\wedge$Q: Conjunção de P e Q. P$\wedge$Q é verdade se tanto P como Q forem verdade.

P$\vee$Q: Disjunção de P e Q. (P$\vee$Q) é verdade se P for verdade, ou se Q for verdade, ou ambos.

P$\Rightarrow$Q: P implica Q. Para melhor perceber as condições em que (P$\Rightarrow$Q) é verdade, note-se que (P$\Rightarrow$Q) é, por definição, o mesmo que ($\neg$P$\vee$Q). Isto significa que (P$\Rightarrow$Q) é verdade quando ($\neg$P$\vee$Q) for verdade, isto é, quando P é falso ou quando Q é verdade, ou ambos. Dito de forma mais sumária, P$\Rightarrow$Q é verdade se P for falso, ou se Q for verdade.

P$\Leftrightarrow$Q: P e Q são equivalentes. P$\Leftrightarrow$Q é verdade se tanto P como Q forem verdade, ou se tanto P como Q forem falsos. Para melhor compreender esta condição, atente-se na definição da equivalência. Por definição (P$\Leftrightarrow$Q) é o mesmo que ((P$\Rightarrow$Q) $\wedge$ (Q$\Rightarrow$P)). Substituindo as implicações pela definição, obtém-se (($\neg$P$\vee$Q) $\wedge$ ($\neg$Q$\vee$P)). Desenvolvendo, obtém-se ainda (($\neg$P$\vee$Q) $\wedge$ $\neg$Q) $\vee$ (($\neg$P$\vee$Q) $\wedge$ P), ou seja ($\neg$P $\wedge$ $\neg$Q) $\vee$ (Q $\wedge$ $\neg$Q) $\vee$ ($\neg$P $\wedge$ P) $\vee$ (Q $\wedge$ P). Mas (Q $\wedge$ $\neg$Q) é falso, o mesmo acontecendo com ($\neg$P $\wedge$ P). Assim sendo, por definição, (P$\Leftrightarrow$Q) é o mesmo que ($\neg$P $\wedge$ $\neg$Q) $\vee$ *Falso* $\vee$ *Falso* $\vee$ (Q $\wedge$ P). (A $\vee$ *Falso*) é verdade se A for verdade, ou se *Falso* for verdade (ou ambas as coisas), mas *Falso* não é verdade. Assim sendo, (A $\vee$ *Falso*) é verdade se A for verdade, ou seja (A $\vee$ *Falso*) é o mesmo que A. Aplicando este último passo, fica finalmente que (P$\Leftrightarrow$Q) é o mesmo que ($\neg$P $\wedge$ $\neg$Q) $\vee$ (Q $\wedge$ P). Ou seja, (P$\Leftrightarrow$Q) é verdade apenas se ($\neg$P $\wedge$ $\neg$Q) $\vee$ (Q $\wedge$ P) for verdade. Tratando-se de uma disjunção, basta que ($\neg$P $\wedge$ $\neg$Q) seja verdade ou que (Q $\wedge$ P) seja verdade. ($\neg$P $\wedge$ $\neg$Q) é verdade se tanto P como Q forem falsos. (Q $\wedge$ P) é verdade se tanto P como Q forem verdade.

Proposições Quantificadas. As proposições quantificadas obtêm-se de proposições mais simples usando os quantificadores existencial ($\exists$) e universal ($\forall$) e a variável quantificada.

Sendo σ uma variável e P uma proposição arbitrária, as seguintes são proposições quantificadas:

$\exists\sigma$ P: P é verdade para pelo menos um σ

$\forall\sigma$ P: P é verdade para qualquer σ

Deve notar-se que as regras de sintaxe acabadas de apresentar para as proposições quantificadas permitem que P não contenha a variável σ. Embora sintaticamente, se possa escrever algo como

∀x Está_a_Chover (em que a variável quantificada não é usada), este tipo de proposições não tem muito interesse do ponto de vista prático. Em geral, nas proposições quantificadas com interesse, a variável quantificada é usada.

Termos

Em lógica de predicados de primeira ordem existem essencialmente três tipos de termos: variáveis, constantes escalares, e expressões ou termos funcionais.

As variáveis representam-se por letras minúsculas, habitualmente por uma das letra x *x, y, z, w*... mas tal não é forçoso.

As constantes escalares são constantes numéricas ou constantes alfanuméricas, as quais são representadas por sequências de caracteres iniciadas por uma letra maiúscula. Exemplos de constantes alfanuméricas são Ana, André e A8.

Além de variáveis e de constantes escalares as quais são termos simples, existem ainda as expressões funcionais, as quais se formam pela aplicação de uma função aos seus argumentos. Exemplos de termos funcionais são

Capital(Portugal):	Capital de Portugal, a qual representa Lisboa
População(Capital(Portugal)):	(2 Milhões que é a população da capital de Portugal em 2003, altura em que o exemplo foi escrito)
2 × 3:	Número 6 (Produto de 2 por 3; poderia ter sido escrito ×(2 , 3) em vez de se usar a notação infixa habitual)

Tal como acontece com algumas proposições atómicas, é habitual usar alguns operadores funcionais de forma infixa (por exemplo, os operadores aritméticos, por exemplo x +2 e 20 % 3)

1.1.2 Semântica da Lógica de Predicados de Primeira Ordem

Quando criamos um sistema baseado em conhecimento, temos de preencher a sua base de conhecimento com o conhecimento do domínio de aplicação em que o sistema será usado. Se a linguagem de representação de conhecimento usada for a lógica de predicados, a base de conhecimentos será constituída por proposições da lógica de predicados, as quais são afirmações envolvendo as entidades do domínio. Algumas dessas proposições são tão simples que não teremos qualquer dúvida de que representam adequadamente o conhecimento que pretendíamos ter representado. No entanto, há conhecimento que só pode ser representado por proposições mais complexas. Como sabemos se escrevemos a proposição que realmente pretendíamos ter escrito? A semântica declarativa permite-nos analisar uma proposição e verificar se ela representa ou não o conhecimento que pretendemos captar.

O principal objetivo desta secção é transmitir as ideias fundamentais usadas na definição de uma semântica declarativa. Procuraremos dar definições tão rigorosas quanto possível sem nos deixarmos afundar em pormenores formais exagerados de modo a que seja mais fácil fazer compreender as ideias mais importantes.

A semântica declarativa de uma linguagem é um mapeamento entre os símbolos e as estruturas simbólicas dessa linguagem e entidades do domínio de que se procura falar, usando a linguagem. Para especificar a semântica da linguagem da lógica de predicados de primeira ordem é necessário especificar em primeiro lugar o domínio de que se pretende falar, isto é o Universo de Discurso. O universo de discurso é um conjunto finito ou infinito, numerável ou não, constituído por todas as entidades de que se pode falar num determinado domínio.

Por outro lado, na linguagem da lógica de predicados de primeira ordem, temos símbolos que representam constantes, variáveis, predicados ou relações, e funções. A semântica declarativa da lógica de predicados de primeira ordem estabelece um mapeamento entre os símbolos da linguagem (constantes, variáveis, predicados ou relações, e funções) e elementos ou conjuntos de elementos do universo de discurso. Estabelece ainda um mapeamento entre expressões funcionais e elementos do universo de discurso. Finalmente, estabelece as condições em que as proposições da linguagem são verdadeiras. Para simplificar, considerando apenas o essencial, esqueceremos as funções e as expressões funcionais.

Cada símbolo de constante ou de variável da linguagem é mapeado num elemento do universo de discurso. Cada símbolo de predicado é mapeado num conjunto de elementos ou num conjunto de sequências de elementos do universo de discurso, designados *túpulos*. Além do universo de discurso, a

semântica declarativa da linguagem da lógica de predicados de primeira ordem necessita de uma função de interpretação (habitualmente designada por I) que estabelece o mapeamento, isto é, a interpretação de que falamos. Por exemplo, se C for um símbolo de constante da linguagem e se U for o universo de discurso, I[C] é um elemento de U. Ou seja, a interpretação (i.e., a semântica) de uma constante da linguagem é um indivíduo do nosso universo de discurso. É habitual representar a aplicação da função de interpretação com parêntesis retos em vez de parêntesis curvos.

Imagine-se um universo de discurso constituído pelas letras do nosso abecedário, U = {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z}. Imagine-se agora que os símbolos de constante da nossa linguagem são A, B, C, D, E, ……, X, Y, Z. Intuitivamente, sabemos que o símbolo A representa o indivíduo *a* do universo de discurso, e que B representa o *b*, etc. É isso que nos diz a função de interpretação I. I(A) = a, I(B) = b, I(C) = c e assim por diante.

Consideremos agora o símbolo de predicado Vogal que representa as vogais. A interpretação do símbolo de predicado Vogal é o conjunto das vogais que pertencem a U. I(Vogal) = {a, e, i, o, u}. Finalmente, considere-se o símbolo de predicado Seguinte que representa uma relação de ordem entre constantes, por exemplo Seguinte(A, B) lê-se que B se segue imediatamente a A. I(Seguinte) há de ser o conjunto de pares ordenados do universo de discurso, tais que o segundo elemento de cada um desses pares ocorre, no abecedário, imediatamente a seguir ao seu primeiro elemento. I(Seguinte) = {<a, b>, <b, c>, <c, d>, ……, <y, z>}.

A função de interpretação I (que estabelece a interpretação dos símbolos e estruturas simbólicas da linguagem) também faz corresponder cada proposição da linguagem a um dos símbolos *verdade* ou *falso*. A semântica declarativa da linguagem da lógica de predicados de primeira ordem dispõe de um conjunto de regras claras para determinar o valor de verdade de cada tipo de proposição. Por exemplo, a proposição relacional $\rho(\alpha_1, \alpha_2, .. \alpha_n)$ é verdade, sse $<I(\alpha_1), I(\alpha_2), .. I(\alpha_n)>$ pertencer a $I(\rho)$. Concretizando para o caso do predicado Vogal, temos que Vogal(A) é verdade, ou seja, I(Vogal(A)) = *verdade* porque $I(O) \in I(Vogal)$. I(A) = a, I(Vogal) = {a, e, i, o, u}, portanto de facto $I(A) \in I(Vogal)$. Mas podemos ver que Vogal(B) é falso porque $I(B) \notin I(Vogal)$. Sendo I(B) = b, $b \notin I(Vogal)$.

Será por exemplo que Seguinte(A, C) é verdade? Temos de ver se $<I(A), I(C)> \in I(Seguinte)$. Sendo I(A) = a e I(C) = c, como $<a, c> \notin I(Seguinte)$, então I(Seguinte(A, C)) = *falso*.

Depois destes exemplos, apresentam-se de seguida as regras que estabelecem a semântica da linguagem da lógica de predicados de primeira ordem, correndo embora o risco de algumas repetições. Começamos por indicar como se especifica o mapeamento dos termos da linguagem. Depois, passaremos para o mapeamento dos símbolos de predicados. Finalmente, passaremos para a interpretação das proposições.

Interpretação dos Termos

Para cada símbolo de constante, tem que se indicar o objeto do universo de discurso representado por esse símbolo. Por exemplo

I[Ana] = Ana Maria

I[Jose] = José António

I[5] = 5

As variáveis também representam objetos do universo de discurso. A função de interpretação I também mapeia variáveis nos elementos do universo de discurso que elas representam.

A interpretação de termos mais complexos tais como conjuntos é criada à custa da composição das interpretações dos seus elementos. Por simplicidade, ignoraremos os termos compostos, até porque não é essencial falar neles para transmitir as principais ideias subjacentes a uma semântica declarativa.

Interpretação de Predicados

Predicados ou relações representam conjuntos. A interpretação de um predicado de aridade N é um conjunto dos N-Túpulos formados pelos elementos do universo de discurso que estão na relação representada por esse predicado. A interpretação do predicado *Filho/2* é o conjunto de pares ordenados $<\alpha, \beta>$ tal que α é filho de β. Por exemplo, I[Filho] = {<Ana Maria, José António>, <José António, Joaquim José>, <Joaquim José, Maria Catarina>, ...}. A interpretação do predicado *Homem/1* é o

conjunto de todas as pessoas de sexo masculino que pertencem ao nosso universo de discurso. Por exemplo, I[Homem]={José António, Joaquim José, ... }

Interpretação de Proposições

A função de interpretação mapeia cada proposição num dos elementos do conjunto {*Verdade, Falso*} dependendo do valor de verdade da proposição considerada. A interpretação de uma proposição é o símbolo *Verdade* se a proposição reflete uma afirmação verdadeira de acordo com o universo de discurso e a função de interpretação escolhidos. É natural que uma proposição seja satisfeita para uma certa função de interpretação e seja falsa para outra função de interpretação. No que se segue vamos explicar as regras que determinam quando uma proposição é verdadeira. Em todos os outros casos, a proposição será falsa.

A interpretação das proposições deve seguir a sua estrutura sintática a qual foi especificada na secção 1.1.1. Começamos pelas proposições atómicas, as mais simples de todas as proposições.

Proposições Atómicas. A interpretação dos símbolos proposicionais é um dos símbolos do conjunto {*Verdade, Falso*}. Se pretendermos dizer que a proposição representada pelo símbolo P é verdadeira, basta dizer que a sua interpretação é *Verdade*, por exemplo I[Faz_Sol]=*Verdade* significa que é verdade que faz sol.

A interpretação das proposições relacionais depende da interpretação dos predicados envolvidos e das interpretações dos seus argumentos. A regra que permite determinar a interpretação de proposições atómicas formadas por um predicado ρ aplicadas aos seus argumentos $<\tau_1, ..., \tau_n>$ é a seguinte: $I[\rho(\tau_1, ..., \tau_n)]$=*Verdade* sse $<I[\tau_1], ..., I[\tau_n]> \in I[\rho]$. Caso contrário, $I[\rho(\tau_1, ..., \tau_n)]$ =*Falso*. Concretizando com os exemplos dos predicados Filho e Homem:

I[Filho(Ana, Ze)] é verdade se <I[Ana], I[Ze]> ∈ I[Filho] = {<Ana Maria, José António>, <José António, Joaquim José>, <Joaquim José, Maria Catarina>, ...}. Admitindo que I[Ana] = Ana Maria e I[Ze] = José António, temos que I[Filho(Ana, Ze)]=*Verdade* porque <Ana Maria, José António> pertence a I[Filho].

I[Homem(Ana)] = *Falso* porque I(Ana) não pertence a I[Homem], dado que se admite que Ana Maia não pertence a {José António, Joaquim José, ... }.

Tendo as condições para determinar as interpretações de proposições atómicas, podemos agora especificar as condições para determinar as interpretações das proposições lógicas e das proposições quantificadas.

Proposições Lógicas. As regras para determinar a interpretação de proposições lógicas seguem a estrutura sintática dessas proposições. Sendo P e Q proposições arbitrárias são as seguintes as regras de para a sua interpretação semântica

$I[\neg P]$=*Verdade* sse I[P]=*Falso*.

$I[P \wedge Q]$ =*Verdade* sse I[P]=*Verdade* e I[Q]=*Verdade*.

$I[P \vee Q]$ =*Verdade* sse I[P]=*Verdade* ou I[Q]=*Verdade* ou ambos.

$I[P \Rightarrow Q]$ =*Verdade* sse I[P]=*Falso* ou I[Q]=*Verdade*.

$I[P \Leftrightarrow Q]$ =*Verdade* sse I[P]=*Verdade* e I[Q]=*Verdade*, ou se I[P]=*Falso* e I[Q]=*Falso* (alternativamente, poderia dizer-se, se I[P] = I[Q]).

Proposições Quantificadas. Bastam duas regras para determinar a interpretação das proposições quantificadas, uma relativa ao quantificador universal e outra relativa ao quantificador existencial.

A ideia subjacente à interpretação de $\forall x$ P é que, para todo e qualquer valor de x, a interpretação de P substituindo o x por esse valor é verdade. Mais formalmente, pode dizer-se que $I[\forall x P]$=*Verdade* sse para todo e qualquer elemento α do universo de discurso, I'[P]=*Verdade*, em que I' é uma interpretação igual a I para todos os elementos da linguagem exceto para a variável x -- I'[x] = α e I'[μ]=I[μ] para $\mu \neq x$.

No caso da quantificação existencial, $\exists x$ P é verdade se existir pelo menos um valor de x para o qual a interpretação de P substituindo x por esse valor é verdade. De uma forma mais rigorosa temos que $I[\exists x P]$=*Verdade* sse existir pelo menos um elemento α do universo de discurso, para o qual I'[P]=*Verdade*, em que I' é uma interpretação igual a I para todos os elementos da linguagem exceto para a variável x -- I'[x] = α e I'[μ]=I[μ] para $\mu \neq x$..

Comentário Final

Tendo especificado a interpretação das proposições para todas as construções sintáticas possíveis, temos a certeza de não ter deixado nada para trás. Desta forma garante-se a existência de regras para a determinação da interpretação de todas as proposições existentes na linguagem da lógica de predicados de primeira ordem.

1.2 Regras de inferência da lógica de predicados de primeira ordem

O raciocínio consiste na aplicação sucessiva de regras de inferência. No caso da lógica de primeira ordem, essas regras chamam-se regras de dedução. À geração de uma proposição pela aplicação de uma ou mais regras de inferência chama-se derivação.

Uma regra de inferência é uma regra geral (independente do domínio da aplicação) que especifica as conclusões que podem ser produzidas quando determinadas condições (ou premissas) são satisfeitas. Tanto as premissas como as conclusões de uma regra de inferência são especificadas através de padrões que representam o formato de proposições lógicas. Quando a base de conhecimentos contém um conjunto de proposições que emparelha com as premissas de uma regra de inferência, são concluídas proposições que emparelham com as conclusões dessa regra.

Uma regra de inferência não deve ser confundida com uma implicação ($\forall_x P(x) \Rightarrow Q(x)$). Enquanto que as regras de inferência são regras gerais, as implicações (regras do domínio) são específicas desse domínio.

O raciocínio dedutivo garante que as conclusões produzidas com base em premissas corretas serão também corretas.

Existem muitas regras de dedução. Em geral, existem regras de eliminação e de introdução para cada uma das conectivas lógicas (negação, conjunção, disjunção e implicação); e regras de generalização e de instanciação para cada quantificador (existencial e universal).

Uma regra de eliminação de uma conectiva permite passar de proposições com essa conectiva para proposições sem essa conectiva. Por exemplo, na eliminação da conjunção, parte-se de uma conjunção (e.g., de *forte*$\wedge$*rapido*) e derivam-se proposições sem a conjunção (e.g., *forte* e *rapido*).

Uma regra de introdução funciona ao contrário: parte-se de proposições sem uma dada conectiva e chega-se a proposições com essa conectiva (e.g., de *forte* e de *rapido* deriva-se *forte*$\wedge$*rapido*).

Uma regra de instanciação permite passar de uma proposição com uma variável quantificada para uma proposição com a variável substituída por uma constante[1]. Por exemplo, de $\forall_x P(x)$ deriva-se P(A).

As regras de generalização permitem introduzir quantificadores. Por exemplo, de P(A) deriva-se $\exists_x P(x)$.

As regras de instanciação podem ser vistas como regras de eliminação de quantificadores, e as regras de generalização podem ser vistas como regras de introdução de quantificadores.

Regras da eliminação da implicação

Existem duas regras que eliminam a implicação: Modus Ponens (MP) e Modus Tollens (MT).

Modus Ponens	Modus Tollens
$A \Rightarrow B$ A _____ B	$A \Rightarrow B$ $\neg B$ _____ $\neg A$

As regras MP e MT podem ler-se da seguinte forma: "de uma proposição com a forma $A \Rightarrow B$ e uma proposição com a forma A, deriva-se (conclui-se, infere-se, ou deduz-se) uma proposição com a forma B", e "de uma proposição com a forma $A \Rightarrow B$ e uma proposição com a forma $\neg B$, deriva-se uma

[1] Não apenas constantes. Como veremos mais adiante, em certas condições, também podem ser variáveis e outros termos mais complexos.

proposição com a forma ¬A ", respetivamente. Note-se que A e B podem ser quaisquer. Por exemplo de *(p⇒q)⇒r* e *(p⇒q)* deriva-se *r*.

Eliminação da conjunção (AE, "And Elimination")

Eliminação da conjunção	
A∧B ——— A B	B∧A ——— A B

Se A∧B é verdade, então A é verdade e B também é verdade.

Introdução da conjunção (AI, "And Introduction")

Introdução da conjunção	
A B ——— A∧B	A B ——— B∧A

Se A é verdade e B é verdade, então A∧B é verdade.

Eliminação da disjunção (OE, "Or elimination")

Eliminação da disjunção	
A∨B ¬A ——— B	A∨B ¬B ——— A

Se ou A ou B são verdade e A é falso, então B é verdade.

Introdução da disjunção (OI, "Or introduction")

Introdução da disjunção	
A ——— A∨B	B ——— A∨B

A regra da introdução da disjunção (OI) deve ser usada com cuidado e apenas com o propósito de produzir informação relevante (e.g., uma conclusão desejada ou uma conclusão necessária num passo intermédio de uma derivação). Se não for usada com critério, a IO pode conduzir a informação totalmente irrelevante. Por exemplo, da proposição "o objeto vídeo V001 tem comprimento médio" deriva-se "ou o objeto vídeo V001 tem comprimento médio ou a Bolsa de Lisboa fechou em alta", o que não tem muito sentido (apesar de ser uma conclusão válida).

Eliminação e introdução da dupla negação

Introdução da dupla negação	Eliminação da dupla negação
A ——— ¬¬A	¬¬A ——— A

Estas duas regras são triviais.

Instanciação universal (UI, "Universal instantiation")

A ideia da instanciação universal consiste na substituição de uma variável quantificada universalmente por uma constante. Se todos os objetos x têm a propriedade P, então o objeto particular A também tem a propriedade P.

A forma mais simples da instanciação universal é

Instanciação Universal
$\forall_x P(x)$ ——— P(*cte*)

em que *cte* representa qualquer constante do domínio. Esta especificação da UI não é geral por duas razões. Primeiro, a expressão quantificada pode ter qualquer número de ocorrências da variável quantificada (incluindo 0 ocorrências). Segundo, a variável quantificada pode ser substituída também por outros termos (para além de constantes). Por exemplo, de $\forall_x P(x)$ deriva-se $P(y)$ em que *y* é uma variável livre; e de $\forall_x P(x)$ deriva-se $P(F(y, A, z))$ em que *F* é uma função, *A* é uma constante e *z* e *y* são variáveis livres. No entanto, há limites para as possibilidades de instanciação. Por exemplo, da proposição

$\forall_x \exists_y [Homem(x) \Rightarrow (Mulher(y) \wedge Mae(x, y))]$ *todo o homem tem uma mãe que é mulher*

não se pode concluir

$\exists y [Homem(y) \Rightarrow (Mulher(y) \wedge Mae(y, y))]$ *existe um homem que é mãe de si mesmo*

A forma geral da UI tem que prever todos estes casos:

Instanciação Universal
$\forall_x P$ ——— $P_{\mid x=\tau}$

em que τ é um termo livre para x em P. Isto significa que x não ocorre dentro do âmbito de quantificação de variáveis contidas em τ.

Instanciação existencial (EI, "Existential instantiation")

Instanciação existencial
$\exists_x P$ ——— $P_{\mid x=\tau(v1,\ldots,vn)}$

em que τ é um símbolo de função novo e ν_1 a ν_n são as variáveis livres em P. A ideia subjacente a esta regra consiste em arranjar uma identificação para o objeto que verifica uma dada propriedade. Os detalhes da EI estão explicados a propósito da conversão para forma clausal (secção **Error! Reference source not found.**).

Generalização existencial (EG, "Existential generalization")

Na sua forma mais simples, a generalização existencial diz "se o objeto *A* tem a propriedade *P*, então existe pelo menos um objeto com a propriedade *P*".

Generalização existencial
P(A) ——— $\exists_x P(x)$

Esta especificação da GE não é geral, porque se restringe a predicados com um único argumento.

Conclusão e Comentários

Apesar de não terem sido apresentadas todas as regras de inferência da lógica de predicados de primeira ordem, foram mesmo assim descritas várias regras (mais de 10). Além disso, as regras discutidas, quando são aplicadas, dão origem a passos de inferência muito pequenos. Para se fazer uma dada derivação, serão necessários muitos passos até que o resultado desejado seja obtido. Se tivermos de criar um mecanismo de inferência que use todas essas regras, o processamento envolvido será exagerado porque, a cada passo do processo de inferência, há múltiplas hipóteses de regras a considerar, e ainda porque o processo será constituído por muitos passos. No entanto, a construção de sistemas inteligentes poderá beneficiar muito da criação desse tipo de mecanismos computacionais. Para que isso se faça com maior eficiência seria necessário reduzir o número de regras de inferência e usar regras que pudessem dar passos maiores em direção ao resultado final.

Uma tentativa no sentido de aumentar o tamanho de certos passos de inferência foi a criação de uma regra de inferência que, de uma só vez, efetua mais operações. Trata-se da regra do Modus Ponens Generalizado. Com essa regra seria pois possível reduzir o número de passos de inferência de uma derivação, melhorando a eficiência do processo.

A regra do Modus Ponens pode ser combinada com a regra da Instanciação Universal e com sucessivas aplicações da regra da Introdução da Conjunção para obter uma regra muito mais poderosa capaz de efetuar passos de inferência maiores. À regra obtida chama-se Modus Ponens generalizado. A ideia subjacente a esta nova regra consiste mais ou menos no seguinte

Modus Ponens Generalizado (ideia básica)
$\forall_x P_1(x) \wedge \ldots \wedge P_n(x) \Rightarrow Q(x)$ $P_1(A)$ … $P_n(A)$ _____ $Q(A)$

Esta especificação do Modus Ponens Generalizado não é geral, porque se restringe à utilização de predicados com um único argumento. Se as proposições da base de conhecimento estiverem representadas na forma normal implicativa, é possível escrever uma definição do Modus Ponens Generalizado:

Para proposições atómicas r_i, p_i e q em que existe uma substituição de variáveis θ tal que $Subst(\theta,r_i)=Subst(\theta,p_i)$ para todo o i:

Modus Ponens Generalizado
$r_1,\ldots, r_n, (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$ _____ $Subst(\theta,q)$

Esta regra de inferência tem três vantagens em relação às outras mais elementares:

1. Dá passos de inferência maiores, combinando várias inferências numa só;
2. Dá passos sensatos na medida em que as instanciações que usa são garantidamente úteis – instanciações que fazem com que pares de expressões fiquem iguais (se necessário e possível);
3. A transformação de todas as proposições da base de conhecimentos em forma canónica (forma normal implicativa) é feita apenas uma vez, no início da computação (em vez de tentar transformações dinamicamente à medida que forem necessárias).

No entanto, nem tudo são vantagens. Esta regra, apesar de mais poderosa, permitindo passos maiores na direção do resultado pretendido não substitui todas as regras que combinou numa só porque estas continuam a ser necessárias para casos em que não seja possível aplicar o Modus Ponens Generalizado. Não tendo substituído as regras que combina numa única, o resultado é que aumentou o número de regras disponíveis. Consequentemente, cada passo de inferência tem mais uma regra que deve considerar. Tudo junto é difícil dizer se as suas vantagens ultrapassam as desvantagens. Veremos mais adiante uma outra ideia na tentativa de reduzir a quantidade de processamento envolvido em mecanismos de inferência

baseados na lógica de predicados de primeira ordem. Trata-se de uma abordagem com uma única regra, chamada a regra da Resolução.

Entretanto concluímos esta secção realçando o facto de que as regras descritas não formam um sistema de dedução completo no sentido em que não são suficientes para gerar todas as consequências lógicas de uma base de conhecimentos. Faltam, por exemplo, regras de introdução da implicação e regras da introdução da negação.

1.3 Exemplo de representação e de raciocínio

Esta secção apresenta um exemplo de representação de conhecimento e de dedução usando a lógica de predicados de primeira ordem. Primeiro, explora-se o problema da representação de conhecimento. Seguidamente, as regras de dedução descritas são usadas sobre as proposições do exemplo de representação de conhecimento.

Uma câmara de vídeo regista a passagem de veículos num túnel. Um programa de processamento de imagem extrai diversos parâmetros de cada objeto filmado e envia esses parâmetros a um sistema de classificação baseado em conhecimento.

Os objetos do universo de discurso são VO_1, VO_2, VO_3 … (*video objets*)

BC:

"Se o comprimento de um objeto vídeo for médio e a sua largura for média, o objeto é um carro"
(1) $\forall_x$ [(Comprimento(x, Medio)$\wedge$Largura(x, Medio))$\Rightarrow$Carro(x)]

"Se um objeto vídeo tem um comprimento pequeno e uma largura muito pequena, então é uma mota ou uma bicicleta"
(2) $\forall_x$ [(Comprimento(x, Pequeno)$\wedge$Largura(x, MtPequeno))$\Rightarrow$(Mota(x)$\vee$Bicicleta(x))]

"Nenhum objeto vídeo é uma bicicleta (trânsito proibido a bicicletas)"
(3) $\forall_x \neg$Bicicleta(x)

"O comprimento do objeto vídeo V01 é grande"
(4) Comprimento(V01, Grande)

(5) Largura(V01, Medio)

(6) Comprimento(V02, Pequeno)

(7) Largura(V02, MtPequeno)

Na Figura 1 apresentam-se os passos da derivação. Cada proposição pertencente à base de conhecimentos é assinalada com a letra Δ, cada proposição derivada é anotada com os números das proposições de que derivou e com o nome da regra de dedução usada.

1. $\forall_x$ Comprimento(x, Medio)$\wedge$Largura(x, Medio)$\Rightarrow$Carro(x) Δ
2. $\forall_x$ Comprimento(x, Pequeno)$\wedge$Largura(x, MtPequeno)$\Rightarrow$Mota(x)$\vee$Bicicleta(x) Δ
3. $\forall_x \neg$Bicicleta(x) Δ
4. Comprimento(V01, Grande) Δ
5. Largura(V01, Medio) Δ
6. Comprimento(V02, Pequeno) Δ
7. Largura(V02, MtPequeno) Δ
8. Comprimento(V02, Pequeno)$\wedge$Largura(V02, MtPequeno)
   $\Rightarrow$Mota(V02)$\vee$Bicicleta(V02) 2 UI
9. Comprimento(V02, Pequeno)$\wedge$Largura(V02, MtPequeno) 6,7 AI
10. Mota(V02)$\vee$Bicicleta(V02) 8,9 MP
11. $\neg$Bicicleta(V02)3 UI
12. Mota(V02) 10,11 OE

Figura 1 - Derivação de Mota(V02)

A criação de um sistema computacional de dedução com as regras de inferência descritas na secção 1.2 envolve a definição de uma estratégia apropriada de seleção das regras e das proposições a usar em cada passo. A derivação da Figura 1 foi feita em poucos passos porque já se sabia (de antemão) onde se pretendia chegar e que proposições usar. Se tivesse sido usado um programa de computador para efetuar a mesma derivação, certamente teriam sido dados muito mais passos e teriam sido tentadas diversas vias alternativas até se chegar à conclusão desejada.

O Prolog ("Programming in Logic") é capaz de fazer derivações a partir de bases de conhecimento escolhendo, em cada passo, uma cláusula capaz de chegar à conclusão desejada. Além disso, o Prolog não usa as regras de inferência descritas. Em vez disso, as proposições do domínio são representadas em forma clausal e é usada a resolução como única regra de dedução (ver capítulo 2).

1.4 Raciocínio abdutivo e raciocínio indutivo

Usando uma linguagem de representação semelhante à da lógica de primeira ordem, podemos efetuar outros tipos de raciocínio, por exemplo o raciocínio abdutivo e o raciocínio indutivo.

1.4.1 Raciocínio abdutivo

O raciocínio abdutivo consiste em encontrar explicações possíveis para certas observações ou para certos factos. Este tipo de raciocínio é usado geralmente em tarefas de diagnóstico, em que se pretende encontrar uma explicação para o comportamento anómalo observado num dado sistema. Para muitos autores, o raciocínio abdutivo recorre à seguinte regra de inferência:

Inferência abdutiva baseada na implicação
A$\Rightarrow$B B _____ A

Por exemplo, sabendo que a varicela origina o aparecimento de pequenas babas avermelhadas na pele, e observando um paciente com pequenas babas avermelhadas na pele, pode concluir-se que uma boa explicação para a observação é que o paciente tem varicela. Isto é, de *varicela*$\Rightarrow$*babas* e, *babas* infere-se (abduz-se) *varicela*.

A abdução não garante que as conclusões sejam corretas. A verdadeira explicação para as observações pode ser diferente da explicação inferida. No caso do paciente com pequenas babas na pele, a explicação pode ser a mordedura de um inseto.

Esta forma de encarar o raciocínio abdutivo tem outros problemas. Por exemplo, considere-se o exemplo do hospital em que todos os doentes da enfermaria 1 sofrem de cancro, isto é $enfermaria_1 \Rightarrow cancro$. Neste caso, não é lícito concluir que a explicação para o facto de um paciente ter cancro seja o facto de estar na enfermaria 1. O problema com este tipo de raciocínio é que não se aplica a implicações mas apenas a causas. Uma implicação não é uma causa. Pode definir-se uma regra da abdução com um operador de causalidade:

Inferência abdutiva baseada na causalidade
A CAUSES B B _____ A

Se A causa B e se observamos B, então uma boa explicação para B é A.

1.4.2 Raciocínio indutivo

O raciocínio indutivo é usado para efetuar generalizações a partir de observações particulares.

A Figura 2 apresenta um exemplo de indução de uma implicação que descreve as condições em que determinada proposição é verdade (as condições em que um objeto é preto).

$Corvo(Objeto_1)$ *o objeto 1 é um corvo*	$Preto(Objeto_1)$ *o objeto 1 é preto*
$Corvo(Objeto_2)$	$Preto(Objeto_2)$
$Sapato(Objeto_3)$	$Preto(Objeto_3)$
$Sapato(Objeto_4)$	$Castanho(Objeto_4)$
…	…
$Corvo(Objeto_n)$	$Preto(Objeto_n)$
$\forall_x(Corvo(x) \Rightarrow Preto(x))$ *todos os corvos são pretos*	

Figura 2 - Indução de uma implicação

O problema descrito na Figura 2 é um exemplo de raciocínio indutivo em que se procura descobrir uma condição geral em que determinada relação é verdadeira. O raciocínio indutivo também é falível. Por exemplo, nada nos garante que não haja um corvo castanho.

Outro exemplo, seria determinar as condições em que um dado sistema executa uma determinada ação.

2 Forma clausal

A Programação em Lógica é um método de representação de conhecimento que pode ser usado no desenvolvimento de sistemas inteligentes, em especial de Sistemas Baseados em Conhecimento (SBC). O Prolog é a linguagem de programação em lógica mais conhecida e divulgada. Em vez de usar a lógica de predicados de primeira ordem na sua forma mais vulgar, o Prolog usa a forma clausal (ou forma normal conjuntiva). Usando a forma clausal, um programa não tem que tentar a aplicação de muitas regras de inferência. Existe uma regra de inferência única chamada Regra da Resolução, que pode ser usada para efetuar deduções sobre uma base de conhecimentos escrita na forma clausal.

A secção 2.1 descreve a forma clausal e o processo de conversão de fórmulas da lógica de predicados de primeira ordem para forma clausal. A secção 2.2 descreve a regra de inferência da resolução, explica a utilização da Resolução na inferência por refutação, e exemplifica a sua aplicação.

2.1 Representação em forma clausal. Conversão para forma clausal

Qualquer fórmula bem formada da lógica de primeira ordem pode ser escrita em forma clausal. Uma base de conhecimentos em forma clausal é um conjunto de cláusulas $\{C_1, C_2, C_3, \ldots\}$ o qual representa a sua conjunção $C_1 \wedge C_2 \wedge C_3 \wedge \ldots$ Cada cláusula, C_i, é um conjunto de literais $\{L_{i.1}, L_{i.2}, \ldots, L_{i.m}\}$ que representa a sua disjunção $L_{i.1} \vee L_{i.2} \vee \ldots \vee L_{i.m}$. Um literal é um átomo (isto é, uma fórmula atómica) ou um átomo negado. Finalmente, um átomo é um símbolo proposicional ou um predicado aplicado a uma lista de argumentos, por exemplo *assassino(lee)*, *presidente(kennedy)*, *matou(X, Y)*, *fuma(luke)*.

A conversão para forma clausal é um processo sistemático composto pela sequência de passos descrita de seguida. Os passos da conversão serão ilustrados pelo seguinte exemplo:

$\forall_x$ [Homem(x) $\Rightarrow \exists_y$ (Mulher(y) $\wedge$ Mae(x, y))]

Para cada homem, existe uma mulher que é sua mãe

1º Remover as implicações

Neste passo, as implicações (a $\Rightarrow$ b) são substituídas por disjunções ($\neg$a $\vee$ b). Por exemplo,

$\forall_x$ [Homem(x) $\Rightarrow \exists_y$ (Mulher(y) $\wedge$ Mae(x, y))]

é substituída por

$\forall_x$ [$\neg$Homem(x) $\vee \exists_y$ (Mulher(y) $\wedge$ Mae(x, y))]

2º Mover as negações para dentro

Este passo consiste em aplicar transformações à expressão obtida no passo anterior, de tal modo que todas as negações existentes sejam aplicadas apenas a átomos (i.e., fórmulas atómicas). Para isso, usam-se as seguintes equivalências:

$\neg(a \wedge b) \equiv \neg a \vee \neg b$ $\qquad$ $\neg\forall_x P \equiv \exists_x \neg P$

$\neg(a \vee b) \equiv \neg a \wedge \neg b$ $\qquad$ $\neg\exists_x P \equiv \forall_x \neg P$

$\neg\neg a \equiv a$

A fórmula

$\forall_x$ [$\neg$Homem(x) $\vee \exists_y$ (Mulher(y) $\wedge$ Mae(x, y))]

não sofre qualquer modificação porque a única negação existente já está aplicada a uma fórmula atómica: $\neg$Homem(x).

3º "Skolemização"

A "Skolemização" serve para remover os quantificadas existenciais. Para isso, substitui-se cada variável quantificada existencialmente por uma constante de Skolem ou por uma expressão funcional com uma função de Skolem. A fórmula $\exists_x$ P(x) significa que existe pelo menos um objeto x (não identificado) para

o qual a propriedade P é verdade. Suponhamos que a constante Sk1 denota um desses objetos para o qual P é verdade. Então podemos substituir a fórmula $\exists_x$ P(x) pela fórmula P(Sk1) mantendo o mesmo valor de verdade. Sk1 é uma constante nova, à qual se chama constante de Skolem. É importante que a constante de Skolem não seja ainda conhecida (seja nova). Se for usado uma constante que já tenha aparecido, corre-se o risco de dizer que P é verdade para uma constante errada. Por exemplo, de $\exists$xAlto(x) não queremos inferir Alto(João), pois o João pode não ser alto. Nos casos de variáveis quantificadas existencialmente no âmbito de quantificadores universais, a substituição não é tão simples. Enquanto que a fórmula $\exists_y\forall_x$P(x, y) significa que existe pelo menos um y para o qual P(x, y) é verdade (para todos os x), a fórmula$\forall_x\exists_y$P(x, y) significa que para cada x, existe um y correspondente a esse x para o qual P(x, y) é verdade. Portanto, as variáveis quantificadas existencialmente dentro do âmbito de quantificadores universais (y, no segundo caso) dependem das variáveis quantificadas por esses quantificadores universais (x, no segundo caso).

A Skolemização consiste em substituir cada expressão com o formato geral $\exists_x$P, por outra expressão P'. Se x não ocorrer no âmbito de nenhum quantificador universal, P' obtém-se de P pela substituição de todas as ocorrências de x por uma constante de Skolem. Se x ocorrer no âmbito de quantificadores universais, P' obtém-se de P pela substituição de todas as ocorrências de x por um termo funcional resultante da aplicação de uma nova função a todas as variáveis quantificadas por esses quantificadores universais. A esta função chama-se função de Skolem.

Por exemplo, a fórmula

$\forall_x$ [¬Homem(x) $\vee$ $\exists_y$ (Mulher(y) $\wedge$ Mae(x, y))]

deve ser substituída por

$\forall_x$ [¬Homem(x) $\vee$ (Mulher(Sk2(x)) $\wedge$ Mae(x, Sk2(x)))]

em que o termo Sk2(x) denota o indivíduo que é a mãe de x. Seria um erro substituir y por uma constante. Isso significaria que todos os homens têm a mesma mãe. A função de Skolem também tem que ser nova, para que não se caia no risco de obter uma fórmula como a seguinte:

$\forall_x$ [¬Homem(x) $\vee$ (Mulher(Coseno(x)) $\wedge$ Mae(x, Coseno(x)))]

o que seria absurdo, pois Coseno(x) não é mulher nem muito menos mãe de x.

A Skolemização é o passo que introduz maiores danos em todo o processo da conversão porque a proposição obtida neste passo não é equivalente à proposição original, no entanto temos algumas garantias: se a fórmula original tiver um dado valor de verdade, então o resultado da conversão terá o mesmo valor, e se a fórmula resultante da conversão for verdadeira, pode garantir-se que a fórmula original também é verdadeira.

4º Retirar os quantificadores universais

Nesta altura, todas as variáveis da fórmula estão quantificadas universalmente, pelo que podemos retirar os quantificadores. Por exemplo, a fórmula

$\forall_x$ [¬Homem(x) $\vee$ (Mulher(Sk2(x)) $\wedge$ Mae(x, Sk2(x)))]

pode ser substituída por

¬Homem(x) $\vee$ (Mulher(Sk2(x)) $\wedge$ Mae(x, Sk2(x)))

porque já sabemos que a variável x está quantificada universalmente.

5º Distribuir os $\wedge$'s sobre $\vee$'s

Neste passo pretende-se passar a uma forma em que não apareçam conjunções dentro de disjunções. Para isso, aplicam-se as seguintes regras:

a $\wedge$ (b $\vee$ c) $\equiv$ (a $\wedge$ b) $\vee$ (a $\wedge$ c)

a $\vee$ (b $\wedge$ c) $\equiv$ (a $\vee$ b) $\wedge$ (a $\vee$ c)

até que a fórmula seja uma conjunção de literais, ou uma conjunção de disjunções de literais.

A fórmula

¬Homem(x) $\vee$ (Mulher(Sk2(x)) $\wedge$ Mae(x, Sk2(x)))

passa a

$(\neg Homem(x) \vee Mulher(Sk2(x)) \wedge (\neg Homem(x) \vee Mae(x, Sk2(x)))$

a qual é uma conjunção de disjunções, como pretendido.

6º Escrever em forma de cláusulas

Nesta altura, a fórmula reduz-se a uma conjunção $C_1 \wedge \ldots \wedge C_n$ em que cada C_i é um literal ou uma disjunção de literais. Cada um dos C_i dá origem a uma cláusula, a qual é representada por um conjunto de um ou mais literais: $\{L_{i.1}, \ldots, L_{i.m}\}$.

Por exemplo, a fórmula

$(\neg Homem(x) \vee Mulher(Sk2(x)) \wedge (\neg Homem(x) \vee Mae(x, Sk2(x)))$

dá origem às duas cláusulas

$\{\neg Homem(x), Mulher(Sk2(x))\}$

$\{\neg Homem(x), Mae(x, Sk2(x))\}$

Usando a notação do Prolog, cada cláusula pode escrever-se com o formato

Cabeça :- Corpo

em que a cabeça da cláusula é a lista dos literais positivos separados por ponto e vírgula, e o corpo é a lista dos literais negativos separados por vírgulas. Note-se que o Prolog apenas permite a utilização de cláusulas em que o número de literais positivos é, no máximo, um (cláusulas de Horn); as variáveis têm que começar por maiúscula e as constantes têm que começar por minúscula.

Em termos do exemplo, temos o seguinte

mulher(sk2(X)) :- homem(X).

Se X é um homem, então sk2(X) (i.e., a mãe de X) é uma mulher.

mae(X, sk2(X)) :- homem(X).

Se X é um homem, então sk2(X) é mãe de X.

2.2 Resolução

Nesta secção, apresenta-se a regra de inferência da resolução e descreve-se o mecanismo de inferência de por refutação, usando a Regra de Resolução.

2.2.1 A regra da resolução

Na sua forma mais simples, a resolução diz que se tivermos uma cláusula A com um literal A_h e uma clausula B com um literal B_k, e $A_h \equiv \neg B_k$, então deriva-se uma cláusula com todos os literais de A exceto A_h e todos os literais de B exceto B_k.

$$\frac{\{A_1, \ldots, A_{h-1}, A_h, A_{h+1}, \ldots, A_n\} \quad \{B_1, \ldots, B_{k-1}, B_k, B_{k+1}, \ldots, B_m\}}{\{A_1, \ldots, A_{h-1}, A_{h+1}, \ldots, A_n, B_1, \ldots, B_{k-1}, B_{k+1}, \ldots, B_m\}}$$

Esta regra de inferência, aparentemente menos intuitiva do que o Modus Ponens e as outras regras do cálculo de predicados na sua forma habitual, pode tornar-se bastante intuitiva em situações particulares.

Vejamos um exemplo de aplicação de resolução com duas cláusulas muito simples. Se tivermos as cláusulas $\{\neg A, B\}$ e $\{A\}$ podemos derivar $\{B\}$ por resolução. Se examinarmos este exemplo mais de perto facilmente percebemos que a resolução fez o papel do Modus Ponens. De facto, a clausula $\{\neg A, B\}$ representa exatamente o mesmo que a implicação $A \Rightarrow B$, enquanto que a clausula $\{A\}$ representa o literal A. A aplicação do Modus Ponens a estas duas fórmulas resulta na fórmula B, cuja forma clausal é exatamente $\{B\}$.

Além do Modus Ponens, a resolução também desempenha o papel do Modus Tolens. Suponhamos para isso que temos as clausulas $\{\neg A, B\}$ e $\{\neg B\}$. Usando resolução nestas duas clausulas, obtemos a clausula $\{\neg A\}$. Para vermos que esta aplicação da resolução é equivalente à aplicação do Modus Ponens basta verificar que $\{\neg A, B\}$ e $\{\neg B\}$ são as representações em forma clausal de $A \Rightarrow B$ e de $\neg B$, e que aplicando o Modus Tolens a estas duas fórmulas se obtém $\neg A$ cuja representação clausal é exatamente $\{\neg A\}$.

Mas a resolução é mais geral do que o Modus Ponens e o Modus Tolens. Por exemplo, tendo as clausulas $\{\neg A, B, C\}$ e $\{A, D\}$, podemos obter $\{B, C, D\}$ o que é bem mais do que o Modus Ponens ou o Modus Tolens.

Mais ainda. Sabendo que uma clausula representa a disjunção dos seus literais e que P é equivalente a $P \vee P$, então aplicando a resolução a $\{\neg A, B\}$ e $\{A, B\}$ obtém-se $\{B\}$.

Até agora a resolução foi definida de modo que só se pode aplicar a literais constantes, mas na sua forma mais geral, a resolução pode também aplicar-se a literais com variáveis, usando substituições.

Uma substituição é um conjunto de pares formados por uma variável e um termo, ν/τ, por exemplo $\{x/A, y/w, z/F(w)\}$. A aplicação de uma substituição a um literal resulta noutro literal em que cada ocorrência das variáveis comuns ao literal e à substituição são substituídas pelos termos correspondentes especificados na substituição. Por exemplo, a aplicação da substituição $\{x/A, y/w, z/F(w)\}$ ao literal P(x, z, B, s) resulta no literal P(A, F(w), B, s). A substituição de y por w não se aplica porque P(x, z, B, s) não contém y; A não é substituído porque se trata de uma constante; e s não é substituído porque a substituição $\{x/A, y/w, z/F(w)\}$ não especifica qualquer substituto para s.

Recorrendo a substituições pode enunciar-se uma versão mais geral da regra da resolução. Tendo duas clausulas A contendo A_h e B contendo B_k então, se existir uma substituição ϕ tal que $\phi A_h \equiv \neg \phi B_k$, pode derivar-se a cláusula constituída pelos literais que resultam da aplicação de ϕ a todos os literais de A exceto A_h e da aplicação de ϕ a todos os literais de B exceto B_k.

$$\frac{\{A_1, \ldots, A_{h-1}, A_h, A_{h+1}, \ldots, A_n\} \quad \{B_1, \ldots, B_{k-1}, B_k, B_{k+1}, \ldots, B_m\}}{\{\phi A_1, \ldots, \phi A_{h-1}, \phi A_{h+1}, \ldots, \phi A_n, \phi B_1, \ldots, \phi B_{k-1}, \phi B_{k+1}, \ldots, \phi B_m\}}$$

em que ϕP é a aplicação da substituição ϕ a P.

Esta nova versão da resolução é mais poderosa do que as regras do Modus Ponens e do Modus Tolens generalizados (ver secção **Error! Reference source not found.**).

Vejamos agora alguns exemplos de aplicação.

Tendo as clausulas $\{\neg P(x, B, z), Q(x, y, z)\}$ e $\{P(A, y, z)\}$, podemos obter $\{Q(A, B, z)\}$. Note-se que uma das substituições ϕ que permite fazer com que $\phi \neg P(x, B, z) \equiv \neg \phi P(A, y, z)$ é $\phi = \{x/A, y/B\}$, z não necessita ser substituído. No entanto, a substituição $\{x/A, y/B, z/C\}$ também faria o mesmo serviço. Porque razão escolhemos a primeira e não a segunda substituição? Porque a segunda impõe uma restrição adicional inútil que poderá comprometer futuras aplicações da resolução. O enunciado da regra da resolução deve ser tornado mais preciso dizendo que se usa a substituição mais geral possível. Uma substituição é mais geral do que outra se especificar substitutos para menos variáveis.

Neste segundo exemplo, temos as clausulas $\{\neg P(x, B, z), Q(x)\}$ e $\{P(A, y, z), Q(B)\}$. Aplicando a resolução com a substituição $\{x/A, y/B\}$, obtém-se a cláusula $\{Q(A), Q(B)\}$. Note-se que a aplicação da substituição $\{x/A, y/B\}$ a Q(x) dá origem a Q(A), a qual é diferente de Q(B). Consequentemente, os dois literais em Q pertencem ambos à cláusula final.

Finalmente, analisemos um último exemplo com as cláusulas $\{\neg P(x, A), Q(x)\}$ e $\{P(x, F(A))\}$. Neste caso não há nenhuma substituição capaz unificar $\neg P(x, A)$ com $\neg P(x, F(A))$ porque não há maneira de tornar A=F(A). É bom de enfatizar que não se trata de escolher uma função F que, aplicada a A, tivesse o valor A. Aqui, a expressão F(A) deve ser encarada como uma estrutura simbólica. Consequentemente, não se pode aplicar a resolução às duas cláusulas deste exemplo.

2.2.2 Prova por refutação usando resolução

A aplicação da resolução a uma base de conhecimentos em fórmula clausal não garante que se consigam derivar todas as consequências lógicas dessa base de conhecimentos. No entanto, essas consequências podem ser provadas de forma indireta, por refutação. Em vez de tentar produzir uma conclusão P a partir de uma base de conhecimentos Δ, prova-se que ¬P é inconsistente com Δ.

O processo geral de provar P a partir de Δ por refutação usando a regra da resolução consiste em negar P, converter ¬P em forma clausal, juntar as cláusulas da negação de P a Δ e derivar, usando resolução, a clausula vazia {}, a qual representa a contradição.

Pode demonstrar-se que se P for uma consequência lógica de Δ, então pode provar-se P a partir de Δ por refutação, usando a resolução. Isto significa que este método de prova é um método completo (porque pode provar todas as consequências lógicas de uma base de conhecimentos).

Pode também demonstrar-se que se for provado por refutação usando resolução que P resulta de Δ, P é de facto uma consequência lógica de Δ. Diz-se que a refutação usando resolução é um método de prova correto (porque não prova que P deriva de Δ se P não for uma consequência lógica de Δ).

2.2.3 Exemplo de derivação usando refutação por resolução

Vamos recorrer de novo ao exemplo da secção 1.3, mas agora usando refutação por resolução para provar Mota(V02). Para isso, começa por se converter a base de conhecimentos em forma clausal.

Depois, há primeiro que negar a fórmula que pretendemos provar, converter a sua negação em forma clausal e juntá-la à base de conhecimentos. Seguidamente, aplicam-se passos de resolução até que se obtém a cláusula vazia, i.e., a contradição. A Figura 3 mostra todo o processo.

1.	{¬Comprimento(x, Medio), ¬Largura(x, Medio), Carro(x)}	Δ
2.	{¬Comprimento(x, Pequeno), ¬Largura(x, MtPequeno), Mota(x), Bicicleta(x)}	Δ
3.	{¬Bicicleta(x)}	Δ
4.	{Comprimento(V01, Grande)}	Δ
5.	{Largura(V01, Medio)}	Δ
6.	{Comprimento(V02, Pequeno)}	Δ
7.	{Largura(V02, MtPequeno)}	Δ
8.	{¬Mota(V02)}	¬G
9.	{¬Largura(V02, MtPequeno), Mota(V02), Bicicleta(V02)}	2,6
10.	{Mota(V02), Bicicleta(V02)}	7,9
11.	{Mota(V02)}	3,10
12.	{}	8,11 cqd

Figura 3 – Derivação usando refutação por resolução

Este exemplo simples poderia ter sido resolvido sem usar a refutação. Poderia ter-se partido da base de conhecimentos em forma clausal e aplicado a resolução sucessivamente até derivar Mota(V02). Em geral, essa estratégia não permite produzir os resultados desejados. Se usarmos refutação, poderemos sempre provar qualquer consequência da base de conhecimentos.