Lógica dos Predicados -

Regras de Inferência

Todas as regras de inferência definidas na Lógica Proposicional são válidas para a Lógica de Predicados, apenas referenciando-as para os quantificadores.

Exemplo: ~F(a) v ∃xF(x), ∃xF(x) → P |- F(a) → P
Prova:

1 ~F(a) v ∃xF(x)	Premissa
2 ∃xF(x) → P	Premissa
3 F(a)	Hipótese
4 ~~F(a)	3 DN
5 ∃xF(x)	1,4 SD
6 P	2,5 MP
7 F(a) → P	3,6 PC

1. **Eliminação Universal (EU)**: De uma fbf quantificada universalmente **∀xϕ(x)**, infere-se uma fbf da forma **ϕ(a)**, a qual resulta de se substituir cada ocorrência da variável **x** em **ϕ** por uma letra nominal **a**. Esta regra é, as vezes, chamada de Instanciação Universal.
   Exemplo: ∀x(H(x) → M(x)), H(a) |- M(a)
   Prova:

1 ∀x(H(x) → M(x))	Premissa
2 H(a)	Premissa
3 H(a) → M(a)	1 EU
4 M(a)	2,3 MP

2. **Introdução Universal (IU)**: De uma fbf contendo uma letra nominal **a**, que não ocorre em qualquer premissa ou em qualquer hipótese vigente na linha em que **ϕ** ocorre, infere-se uma fbf da forma **∀xϕ(x)**, onde **ϕ(x)** é o resultado de se substituir todas as ocorrências de **a** em **ϕ** por uma variável **x** que não ocorra em **ϕ**.
   Exemplo: ∀x(P(x) → C(x)), ∀x(C(x) → V(x)) |- ∀x(P(x) → V(x))
   Prova:

1 ∀x(P(x) → C(x))	Premissa
2 ∀x(C(x) → V(x))	Premissa
3 P(a) → C(a)	1 EU
4 C(a) → V(a)	2 EU
5 P(a) → V(a)	3,4 SH
6 ∀x(P(x) → V(x))	5 IU

3. **Introdução Existencial (IE)**: Dada uma fbf **ϕ** contendo uma letra nominal **a**, infere-se uma fbf da forma **∃xϕ(x)**, onde **ϕ(x)** é o resultado de se substituir uma ou mais ocorrências de **a** em **ϕ** por uma variável **x** que não ocorra em **ϕ**. Entre as restrições apresentadas para a utilização da IE ressalta-se:
   - **a** pode ocorrer em uma hipótese, não utilizada ainda, ou em uma premissa;
   - a variável **x** não precisa substituir todas as ocorrências de **a** em **ϕ**, é preciso substituir somente uma ou mais;
   - IE permite introduzir somente um quantificador existencial por vez e somente do lado esquerdo da fórmula.

   Exemplo: ∀x(F(x) v G(x)) |- ∃x(F(x) v G(x))
   Prova:

1 ∀x(F(x) v G(x))	Premissa
2 F(a) v G(a)	1 EU
3 ∃x(F(x) v G(x))	2 IE

4. **Eliminação Existencial (EE)**: Dada uma fbf quantificada existencialmente **∃xϕ(x)** podemos inferir **ϕ(a)**, contanto que a letra nominal não ocorra em **ϕ(x)**, nem em qualquer premissa, nem em qualquer hipótese e nem em qualquer passo anterior da derivação. Estas restrições podem ser facilmente satisfeitas escolhendo uma nova letra nominal cada vez que a Eliminação Existencial for aplicada.
   Exemplo: ∃x(F(x) ^ G(x)) |- ∃x(F(x))
   Prova:

1 ∃x(F(x) ^ G(x))	Premissa
2 F(a) ^ G(a)	1 EE
3 F(a)	2 ^T
4 ∃x(F(x))	3 IE

Algoritmo: Converter em Forma Clausal

Exemplo a ser seguido durante os passos: Suponha que saibamos que todos os romanos que conhecem Marcos odeiam César ou acham que todos os que odeiam alguém são loucos.

∀x: [Romano(x) ^ conhece(x, Marcos)] → [odeia(x, César) v (∀y: ∃z: odeia(y, z) → acha-louco(x, y))]

1. Elimine os conectivos lógicos → e ↔, usando o fato de que a → b eqüivale a ~a v b, e a ↔ b eqüivale a (a → b) ^ (b → a). Esta transformação na fbf anterior produz:

∀x: ~[Romano(x) ^ conhece(x, Marcos)] v [odeia(x, César) v (∀y: ~(∃z: odeia(y, z) v acha-louco(x, y))]

2. Reduza o escopo de cada ~ a um único termo, trazendo os sinais de negação para antes dos átomos, através da repetição das seguintes leis:
   - ~~F = F
   - ~(F v G) = ~F ^ ~G
   - ~(F ^ G) = ~F v ~G
   - ~(∀xF(x)) = ∃x(~F(x))
   - ~(∃xF(x)) = ∀x(~F(x))

   Estas transformações na fbf do passo 1 produz:

∀x: [~Romano(x) v ~conhece(x, Marcos)] v [odeia(x, César) v ∀y: ∀z: ~odeia(y, z) v acha-louco(x, y))]

3. Padronize as variáveis para que cada quantificador fique vinculado a uma única variável. Uma vez que as variáveis são apenas nomes fictícios, este processo não pode afetar o valor-verdade da fbf. Por exemplo, a fórmula

   ∀x: P(x) v ∀x: Q(x)

   poderia ser convertida em

   ∀x: P(x) v ∀y: Q(y).

   Este passo prepara para o passo seguinte.

4. Mova todos os quantificadores à esquerda da fórmula sem mudar sua ordem relativa. Isto é possível porque não há conflitos entre os nomes das variáveis. Esta operação na fórmula da etapa 2 resulta em:

∀x: ∀y: ∀z: [~Romano(x) v ~conhece(x, Marcos)] v [odeia(x, César) v (~odeia(y, z) v acha-louco(x, y))]

Neste ponto, a fórmula está no formato que chamamos de **forma normal prenex**. Ele consiste em um prefixo de quantificadores seguido de uma matriz, que é livre de quantificadores.

5. Elimine os quantificadores existenciais. Uma fórmula que contém uma variável quantificada em termos existenciais afirma que existe um valor que pode ser substituído pela variável e que faz com que a fórmula seja verdadeira. Podemos eliminar o quantificador substituindo a variável por uma referência a uma função que produza o valor desejado. Uma vez que não sabemos necessariamente como produzir o valor, precisamos criar um novo nome de função para cada substituição. Não fazemos nenhuma definição sobre estas funções, exceto a de que elas precisam existir. Portanto, por exemplo, a fórmula

   ∃y: Presidente(y)

   pode ser transformada na fórmula

   Presidente(S1)

   onde S1 é uma função sem argumentos, que de algum modo produz um valor que satisfaça Presidente.

   Se ocorrerem quantificadores existenciais dentro do escopo dos quantificadores universais, então o valor que satisfaz o predicado pode depender dos valores das variáveis quantificadas universalmente. Por exemplo, na fórmula

   ∀x: ∃y: pai-de(y, x)

   o valor que satisfaz pai-de depende do valor particular de x. Assim, precisamos gerar funções onde o número de argumentos é igual ao número de quantificadores universais em cujo escopo a expressão ocorre. Portanto, este exemplo seria transformado em

   ∀x: pai-de(S2(x), x)

   Estas funções geradas são chamadas de **funções de Skolem**. Às vezes, funções deste tipo sem nenhum argumento são chamadas de constantes de Skolem.

6. Elimine o prefixo. Neste ponto, todas as variáveis restantes foram quantificadas universalmente, portanto, o prefixo pode ser eliminado e qualquer procedimento de prova usado pode simplesmente assumir que qualquer variável encontrada é universalmente quantificada. Agora a fórmula produzida no passo 4 aparece como

   [~Romano(x) v ~conhece(x, Marcos)] v [odeia(x, César) v (~odeia(y, z) v acha-louco(x, y))]

7. Converta a matriz em uma conjunção de disjunções. No caso do nosso exemplo, como não há nenhum E, é necessário apenas explorar a propriedade associativa do OU [ou seja, a v (b v c) = (a v b) v c] e simplesmente remover os parênteses, produzindo

   ~Romano(x) v ~conhece(x, Marcos) v odeia(x, César) v ~odeia(y, z) v acha-louco(x, y)

   Entretanto, é também freqüentemente necessário explorar a propriedade distributiva [isto é, (a ^ b) v c = (a v c) ^ (b v c)]. Por exemplo, a fórmula

   (inverno ^ usar-botas) v (verão ^ usar-sandálias)

   torna-se, depois de uma aplicação da regras,

   [inverno v (verão ^ usar-sandálias)] ^ [usar-botas v (verão ^ usar-sandálias)]

   e depois de uma segunda aplicação, necessária, já que ainda há conjunções unidas por OU's

   (inverno v verão) ^ (inverno v usar-sandálias) ^ (usar-botas v verão) ^ (usar-botas v usar-sandálias)

8. Crie uma cláusula separada que corresponda a cada conjunção. Para que uma fbf seja verdadeira, todas as cláusulas geradas a partir dela precisam ser verdadeiras. Se vamos trabalhar com várias fbf's, todas as cláusulas geradas por cada uma delas podem agora ser agrupadas para representar o mesmo conjunto de fatos representados pelas fbf's originais.

   ~Romano(x)
   ~conhece(x, Marcos)
   odeia(x, César)
   ~odeia(y, z)
   acha-louco(x, y)

9. Padronize distintamente as variáveis do conjunto de cláusulas geradas na etapa 8. Isto significa renomear as variáveis para que duas cláusulas não façam referência à mesma variável. Ao fazermos esta transformação, baseamo-nos no fato de que

   $(\forall x : P(x) \wedge Q(x)) = \forall x: P(x) \wedge \forall(x): Q(x)$

   Assim, já que cada cláusula é um conjunto separado e como todas as variáveis são universalmente quantificadas, não precisa haver nenhum relacionamento entre as variáveis de duas cláusulas, mesmo que elas tenham sido geradas a partir da mesma fbf.

   ~Romano(x1)
   ~conhece(x2, Marcos)
   odeia(x3, César)
   ~odeia(y1, z1)
   acha-louco(x4, y2)

RESOLUÇÃO EM LÓGICA PROPOSICIONAL

Algoritmo: Resolução Proposicional

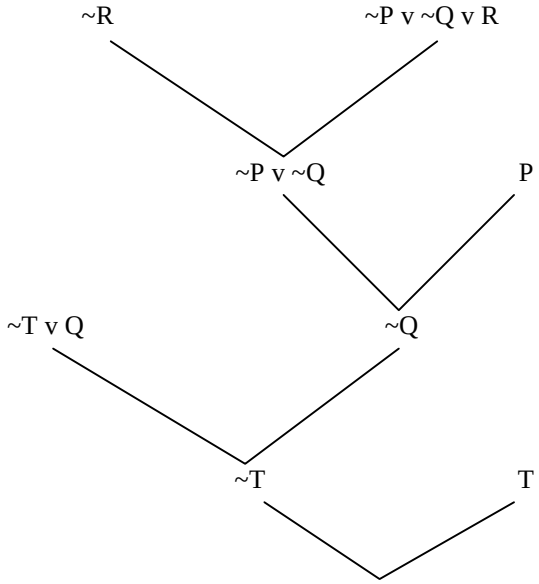
1. Converta todas as proposições de F para a forma clausal.
2. Negue P e converta o resultado para a forma clausal. Acrescente-o ao conjunto de cláusulas obtido na etapa 1.
3. Repita até que uma contradição seja encontrada ou até que nenhum progresso a mais possa ser feito:

   (a) Selecione duas cláusulas. Chame-as de cláusulas-pai.
   (b) Resolva as duas juntas. A cláusula resultante, chamada de resolvente, será a disjunção de todos os literais de ambas as cláusulas-pai com a seguinte exceção: se houver pares de literais L e ~L tais que uma das cláusulas-pai contenha L e a outra ~L, selecione um desses pares e elimine tanto L quanto ~L do resolvente.
   (c) Se o resolvente for a cláusula vazia, foi encontrada uma contradição. Se não for, acrescente-o ao conjunto de cláusulas disponíveis ao procedimento.

Exemplo:

Axiomas Dados	Conversão para a Forma Clausal	
P	P	(1)
$(P \wedge Q) \rightarrow R$	~P v ~Q v R	(2)
$(S \vee T) \rightarrow Q$	~S v Q	(3)
	~T v Q	(4)
T	T	(5)

Provar R.



ALGORITMO DE UNIFICAÇÃO

Algoritmo: Unificar (L1, L2)

1. Se L1 e L2 forem ambos constantes ou variáveis, então:

   (a) Se L1 e L2 forem idênticos, então retorne NIL.
   (b) Caso contrário, se L1 for uma variável, então, se L1 ocorrer em L2, retorne {FRACASSO} ou retorne (L2/L1).
   (c) Caso contrário, se L2 for uma variável, então, se L2 ocorrer em L1, retorne {FRACASSO}, ou retorne (L1/L2).
   (d) Caso contrário, retorne {FRACASSO}.

2. Se os símbolos do predicado original em L1 e L2 não forem idênticos, então retorne {FRACASSO}.
3. Se L1 e L2 tiverem um número diferente de argumentos, então retorne {FRACASSO}.
4. Atribua NIL a SUBST. (Ao final deste procedimento, SUBST conterá todas as substituições usadas para unificar L1 e L2).
5. De i ← 1 para número de argumentos em L1:

   (a) Chame UNIFICAR com o i-ésimo argumento de L1 e com o i-ésimo argumento de L2, colocando o resultado em S.
   (b) Se S contém FRACASSO, então retorne {FRACASSO}.
   (c) Se S não for igual a NIL, então:
      I. Aplique S ao que resta em L1 e L2.
      II. SUBST := JUNTAR(S, SUBST).
6. Retorne SUBST.

Exemplos: Os literais odeia(x, y) e odeia (Marcos, z) poderiam ser unificados com muitas das seguintes substituições:
(Marcos/x, z/y)
(Marcos/x, y/z)
(Marcos/x, César/y, César/z)
(Marcos/x, Polônio/y, Polônio/z)

RESOLUÇÃO EM LÓGICA DE PREDICADOS

Algoritmo: Resolução

1. Converta todas as declarações de F para a forma clausal.
2. Negue P e converta o resultado para a forma clausal. Acrescente-o ao conjunto de cláusulas obtido em 1.
3. Repita até ser encontrada uma contradição, até não ser mais possível nenhum progresso, ou até que uma quantidade predeterminada de esforço tenha sido despendida:

   (a) Selecione duas cláusulas. Chame-as de cláusulas-pai.
   (b) Resolva as duas juntas. O resolvente será a disjunção de todos os literais de ambas as cláusulas-pai com a realização das substituições apropriadas e com a seguinte exceção: se houver um par de literais T1 e ~T2 tal que uma das cláusulas-pai contenha T1 e a outra contenha T2 e se T1 e T2 forem unificáveis, então nem T1 nem T2 devem aparecer no resolvente. T1 e T2 são chamados de literais complementares. Use a substituição produzida pela unificação para criar o resolvente. Se houver mais de um par de literais complementares, apenas um par deve ser omitido do resolvente.
   (c) Se o resolvente for a cláusula vazia, então uma contradição foi encontrada. Se não, acrescente-o ao conjunto de cláusulas disponíveis ao procedimento.

Exemplo: Axiomas em forma clausal:

1. homem(Marcos)
2. Pompeano(Marcos)
3. ~Pompeano(x1) v Romano(x1)
4. Soberano(César)
5. ~Romano(x2) v leal-a(x2, César) v odeia(x2, César)
6. leal-a (x3, f1(x3))
7. ~homem(x4) v ~soberano(y1) v tenta-assassinar(x4, y1) v leal-a(x4, y1)
8. tenta-assassinar(Marcos, César)

Prova: odeia(Marcos, César)