Paradigma (Lógico) da Programação III

2^o Ano de LMCC

Paulo J Azevedo
Departamento de Informática
`pja@di.uminho.pt`

Predicate Logic is a useful language for stating problems and for representing the pragmatic information for effective problem solving.

R. Kowalski

Introdução à Programação em Lógica

Um outro paradigma da programação:

- Uso de lógica para expressar conhecimento do mundo real a ser retratado.
- Uso de inferência para processar conhecimento i.e. computação = dedução
- Lógica + Controle = Programas
- Programação em Lógica = Raciocínio Mecanizado
- Importância da noção de *consequência lógica*

Programa em Lógica

- Programa = um conjunto de "hipóteses" A
- Output = uma esperada consequência C
- Computação = dedução de C a partir de A
- Semântica de um programa ("meaning") = todas as consequências de A

No nosso caso específico:

- Linguagem é a "Lógica de Cláusulas"
- Método de Inferência é Resolução (Resolution)
- Temos uma estratégia para controlar a inferência

$$\text{Hipóteses} \stackrel{resolution}{\longrightarrow} \text{Conclusões}$$

Caso específico do **Prolog** (**Pro**graming in **Lo**gic):

- linguagem = lógica de Horn
- sistema de inferência = resolução SLD
- estratégia = selecção sequêncial

Um exemplo de um programa em lógica:

```
satisfeito(Prof,Aluna) se boa(Aluna)
boa(Aluna) se atenta(Aluna)
atenta(joana)
```

Pergunta:

"Eu estou satisfeito com alguém ?"

$?\mathsf{satisfeito}(\mathsf{eu}, \mathsf{Alguem})$

Computação para responder à questão:

$?\mathsf{satisfeito}(\mathsf{eu}, \mathsf{Alguem})$

$?\mathsf{boa}(\mathsf{Alguem})$

$?\mathsf{atenta}(\mathsf{Alguem})$

Solução para $\mathsf{Alguem} = \mathsf{joana}$.

Conclusão derivada por esta computação:

$\mathsf{satisfeito}(\mathsf{eu}, \mathsf{joana})$.

Revisões de lógica de primeira ordem (FOL)

Alguma notação (elementos da linguagem):

constantes: $\mathsf{a}, \mathsf{b}, \mathsf{c}, ..., 1, 2, 3$

funções: $\mathsf{f}, \mathsf{g}, \mathsf{h}, ...$

variáveis: $\mathsf{X}, \mathsf{Y}, \mathsf{Z}, ...$

com estes símbolos construímos *termos*: e.g. $\mathsf{f}(\mathsf{b}, \mathsf{g}(\mathsf{X}), \mathsf{a})$

predicados: $\mathsf{satisfeito}, \mathsf{atento}, \mathsf{bom}$

Quantificadores: $\exists, \forall$

fórmulas quantificadas e.g. $(\forall \mathsf{X})(\mathsf{criou}(\mathsf{deus}, \mathsf{X}))$

"Deus criou tudo"

símbolos conectivos: $\&$ $\vee$ $\neg$ $\leftarrow$ (if) $\leftrightarrow$ (iff)

Com estes conectivos construímos fórmulas compostas

e.g. $(\forall \mathsf{X})(\exists \mathsf{Y}\ \mathsf{pai}(\mathsf{Y}, \mathsf{X}) \leftarrow \mathsf{filho}(\mathsf{X}))$

"Todo o filho tem um pai"

Estrutura das fórmulas:

Uma fórmula ou é atómica ou é a composição de fórmulas da seguinte forma (onde A e B são fórmulas):

- A (a própria fórmula)
- $\neg$A (a sua negação)
- (QX)A (onde Q é um quantificador)
- A & B (conjunção)
- A $\vee$ B (disjunção)
- A $\leftarrow$ B (A se B)
- A $\leftrightarrow$ B (A se e só se B)

O âmbito ou contexto (scope) de aplicação do quantificador QX na fórmula (QX)W é definido como sendo a subfórmula W.

Mais definições de linguagem:

Uma variável X é presa (ou ligada) (*bound*) numa fórmula W se:

está no âmbito de um quantificador em W

ou está presente num quantificador em W.

Uma variável X é livre (*free*) numa fórmula W se não é presa (bound).

Uma frase lógica (ou uma fórmula fechada) é uma fórmula em que todas as ocorrências de uma variável são limitadas (bound).

e.g. $(\forall X)((\exists Y)pai(Y, X) \leftarrow filho(X))$ é uma frase

Para a fórmula

$$\mathrm{W} = \mathsf{gosta}(\mathsf{ana}, \mathsf{X}) \leftarrow \mathsf{gosta}(\mathsf{X}, \mathsf{Z})$$

dizemos que:

$\mathsf{gosta}(\mathsf{ana}, \mathsf{X})$	é o consequente de W, e
$\mathsf{gosta}(\mathsf{X}, \mathsf{Z})$	o antecedente de W,

Uma instância de uma fórmula W é o resultado de substituir qualquer termo $t1, t2, ...$ por qualquer variável $X1, X2, ...$.

A instância é representada por $W[X1/t1, X2/t2, ...]$

Exemplo:

$$\mathsf{gosta}(\mathsf{ana}, \mathsf{luis}) \leftarrow \mathsf{gosta}(\mathsf{luis}, \mathsf{pai}(\mathsf{ana}))$$

é uma instância de W, para a substituição

$$\mathrm{W}[\mathsf{X}/\mathsf{luis}, \mathsf{Z}/\mathsf{pai}(\mathsf{ana})].$$

Teoria dos Modelos

Lógica fornece duas noções de *consequência*:

- (Teoria das Provas) Consequência lógica é tudo o que é possível provar usando as premissas (hipóteses).
- (Teoria dos Modelos) Consequência lógica denomina tudo que tem de ser verdade quando as premissas (hipóteses) são verdade.

Interpretação: atribuir significado ("meaning") a uma frase lógica, associando valores verdade aos elementos que a constituem. Esta atribuição tem em conta um domínio específico escolhido.

e.g. a frase $\mathsf{gosta}(\mathsf{ana}, \mathsf{luis})$ é verdade na interpretação:

$$< \mathsf{ana}, \mathsf{ana} >, < \mathsf{ana}, \mathsf{luis} >, < \mathsf{luis}, \mathsf{luis} >$$

para o predicado gosta.

Uma interpretação que faz uma frase lógica ser verdadeira designa-se um *modelo* da frase (ou que satisfaz a frase).

e.g. a interpretação anterior é um modelo de $\mathsf{gosta}(\mathsf{ana}, \mathsf{luis})$.

No entanto as interpretações $\emptyset$ e $< \mathsf{luis}, \mathsf{ana} >$ não são um modelo de $\mathsf{gosta}(\mathsf{ana}, \mathsf{luis})$.

(notar que é tradição as interpretações só mencionarem os átomos que tem o valor *verdadeiro* atribuído. Todos os outros tem valor $falso$).

Uma interpretação que não satisfaz uma frase diz-se um *contra-modelo*.

Exemplo de uma frase que não tem modelos:

$$(\exists X \exists Y)(gosta(X, Y) \ \& \ \neg gosta(X, Y))$$

Mais sobre teoria dos modelos:

Uma frase lógica em que todas as suas interpretações são modelos diz-se ser uma frase *válida*.

a frase $S1$ implica lógicamente a frase $S2$

se e só se

cada modelo de $S1$ é um modelo de $S2$.

formalmente: $S1 \models S2$

Dizemos que:

$S2$ é uma consequência lógica (ou semântica) de $S1$.

Exemplo:

$$(\forall X)(gosta(ana, X)) \models gosta(ana, luis)$$

Teoria dos modelos (cont)

Assim podemos ter a noção de que um conjunto de frases pode implicar uma frase e.g. $\{\neg A, A \leftarrow B\} \models \neg B$

Outro exemplo,

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

Duas frases são lógicamente equivalentes (i.e. $S1 \equiv S2$) se e só se tiverem os mesmos modelos.

Em termos computacionais raramente é necessário recorrer a frases válidas mas há importantes ligações entre *validade* e *implicação lógica.*

Por exemplo $S1 \models S2$ se e só se $(S2 \leftarrow S1)$ é válido.

Conversão para a Forma Clausal

Cláusulas: fórmulas normalizadas em disjunções de literais.

Todas as frases de lógica de primeira ordem podem ser convertidas num conjunto de cláusulas. O "algoritmo" para a conversão é dado nos seguintes passos (note o uso implícito da comutatividade e associatividade dos operadores & e $\vee$).

1. Reescrever cada subfórmula $\mathsf{W1} \leftrightarrow \mathsf{W2}$ em:

$$(\mathsf{W1} \leftarrow \mathsf{W2})\ \&\ (\mathsf{W2} \leftarrow \mathsf{W1})$$

2. Reescrever cada subfórmula $\mathsf{W1} \leftarrow \mathsf{W2}$ em:

$$\mathsf{W1} \vee \neg \mathsf{W2}$$

3. Distribuir todas as ocorrências de $\neg$ reescrevendo:

- $\neg(\exists \mathsf{X})\mathsf{W}$ em $(\forall \mathsf{X})\neg \mathsf{W}$
- $\neg(\forall \mathsf{X})\mathsf{W}$ em $(\exists \mathsf{X})\neg \mathsf{W}$
- $\neg(\mathsf{W1} \vee \mathsf{W2})$ em $\neg \mathsf{W1}\ \&\ \neg \mathsf{W2}$
- $\neg(\mathsf{W1}\ \&\ \mathsf{W2})$ em $\neg \mathsf{W1} \vee \neg \mathsf{W2}$
- $\neg\neg \mathsf{W}$ em W

4. Distribuir todas as ocorrências de $\vee$ reescrevendo:
   - $\mathsf{W} \vee (\mathsf{W1} \,\&\, \mathsf{W2})$ em $(\mathsf{W} \vee \mathsf{W1}) \,\&\, (\mathsf{W} \vee \mathsf{W2})$
   - $\mathsf{W1} \vee (\forall \mathsf{X})\mathsf{W2}$ em $(\forall \mathsf{X})(\mathsf{W1} \vee \mathsf{W2})$
   - $\mathsf{W1} \vee (\exists \mathsf{X})\mathsf{W2}$ em $(\exists \mathsf{X})(\mathsf{W1} \vee \mathsf{W2})$

5. Distribuir todas as ocorrências de $\forall$ reescrevendo:

$$(\forall \mathsf{X})(\mathsf{W1} \,\&\, \mathsf{W2}) \text{ em } (\forall \mathsf{X})\mathsf{W1} \,\&\, (\forall \mathsf{X})\mathsf{W2}$$

6. *Skolemization*: eliminar todas as ocorrências de $\exists$ reescrevendo a fórmula:

$$(\forall \mathsf{X}_1)...(\forall \mathsf{X}_\mathsf{n})(\exists \mathsf{Y})\mathsf{W}(\mathsf{Y})$$

$$\text{em } (\forall \mathsf{X}_1)...(\forall \mathsf{X}_\mathsf{n})\mathsf{W}(\mathsf{f}(\mathsf{X}_1, ..., \mathsf{X}_\mathsf{n}))$$

e todas as fórmulas:

$$(\exists \mathsf{Y})\mathsf{W}(\mathsf{Y}) \text{ em } \mathsf{W}(\mathsf{c}).$$

O processo de Skolemização introduz novos símbolos de funções e constantes para fazer referência ao *nome* do Y que existe.

7. repetir todos os passos de reescrita até não ser possível mais aplicações.

Exemplo:

$(\forall X)(P(X) \leftarrow (\forall U)(Q(U,X) \leftarrow R(U)))$

Do passo 2. da anterior conversão resulta:

$(\forall X)(P(X) \ \vee \ \neg(\forall U)(Q(U,X) \ \vee \ \neg R(U)))$

Do passo 3. temos:

$(\forall X)(P(X) \ \vee \ (\exists U)\neg(Q(U,X) \ \vee \ \neg R(U)))$

$(\forall X)(P(X) \ \vee \ (\exists U)(\neg Q(U,X) \ \& \ \neg\neg R(U)))$

$(\forall X)(P(X) \ \vee \ (\exists U)(\neg Q(U,X) \ \& \ R(U)))$

Passo 4.

$(\forall X)(\exists U)(P(X) \ \vee \ (\neg Q(U,X) \ \& \ R(U)))$

$(\forall X)(\exists U)((P(X) \ \vee \ \neg Q(U,X)) \ \& \ (P(X) \ \vee \ R(U)))$

Passo 6. (Passo 5 não pode ser usado):

$(\forall X)((P(X) \ \vee \ \neg Q(f(X),X)) \ \& \ (P(X) \ \vee \ R(f(X))))$

Passo 7.

$(\forall X)(P(X) \ \vee \ \neg Q(f(X),X)) \ \& \ (\forall X)(P(X) \ \vee \ R(f(X)))$

A forma clausal final é:

$$P(X) \ \vee \ \neg Q(f(X),X)$$

$$P(X) \ \vee \ R(f(X))$$

Forma Clausal

Sendo S uma frase em lógica de primeira ordem e C o conjunto de cláusulas que é a tradução de S:

- S tem um modelo se e só se C tem um modelo.
- $C \models S$
- Se não houver Skolemização temos também $S \models C$.

Uma *forma normal* da lógica de primeira ordem é:

(quantificadores)(disjunção de literais)

Por exemplo:

$$(\forall X)(gosta(ana, X) \vee \neg gosta(X, logica))$$

$$(\forall Y) gosta(luis, Y)$$

Programas em lógica (inclusivé programas Prolog) são simplesmente *conjuntos de cláusulas*. É normal apresentar estas cláusulas sem os quantificadores universais e substituindo as sequências $\vee\neg$ por $\leftarrow$. Assim sendo, o exemplo anterior é:

$$gosta(ana, X) \leftarrow gosta(X, logica)$$

$$gosta(luis, Y)$$

Em Prolog seria:

```
gosta(ana,X) :- gosta(X,logica).
gosta(luis,Y).
```

Reescrever frases lógicas em forma clausal preserva modelos e consequências lógicas. O facto de usarmos a forma clausal vai trazer grandes vantagens nomeadamente na simplificação da *Interpretação Procedimental da Lógica*.