



SCC-630 - Capítulo 5

Representação de Conhecimento através do Prolog

João Luís Garcia Rosa¹

¹Departamento de Ciências de Computação
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos

2011

Sumário

- 1 **Representação do Conhecimento**
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - Cláusulas Definidas
- 2 **Relações entre Objetos**
 - Exemplo
 - Incluindo Regras
 - Regra recursiva
- 3 **Encadeamento lógico do Prolog**
 - Como Prolog responde questões
 - Significados declarativo e procedimental
 - Dados estruturados do Prolog: Listas

Sumário

- 1 **Representação do Conhecimento**
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - Cláusulas Definidas
- 2 **Relações entre Objetos**
 - Exemplo
 - Incluindo Regras
 - Regra recursiva
- 3 **Encadeamento lógico do Prolog**
 - Como Prolog responde questões
 - Significados declarativo e procedimental
 - Dados estruturados do Prolog: Listas

Prolog

- A ideia de usar lógica como um formalismo executável em computador recebeu um grande ímpeto com o advento da linguagem Prolog (“PROgrammation en LOGic”).
- Desenvolvida na década de 1970 em Edinburg, a linguagem Prolog tem a sua aplicação dirigida à computação simbólica (não-numérica).
- Trata-se de uma evolução das linguagens de computador, pois Prolog tem características de linguagens procedimentais (o como), e de linguagens declarativas (o que).

Inferência Lógica do Prolog

- Prolog é uma linguagem de programação lógica.
- Um programa Prolog é uma base de conhecimento, onde cada asserção é uma tradução de uma implicação da lógica.
- A máquina de inferência do Prolog é capaz de realizar deduções lógicas a partir desta base de conhecimento e produzir conhecimento novo. Considere, por exemplo, as seguintes proposições [3]:
 - 1 Todos os homens são mortais.
 - 2 Platão é um homem.
 - 3 Platão é mortal.

Inferência Lógica do Prolog

- Para a lógica, as duas primeiras proposições são premissas e a terceira é a conclusão lógica destas premissas.
- Na lógica de predicados de primeira ordem tem-se:

$$\forall X(homem(X) \rightarrow mortal(X)) \quad (1)$$

$$homem(platao) \quad (2)$$

$$mortal(platao) \quad (3)$$

onde a terceira fórmula seria facilmente obtida a partir das duas primeiras usando qualquer ferramenta lógica de dedução, como por exemplo a regra da resolução.

Inferência Lógica do Prolog

- Toda implicação da lógica da forma $a \rightarrow b$, equivalente a $\neg a \vee b$, gera uma regra Prolog da forma `b :- a.`
- Logo, um programa Prolog para esta base de conhecimento seria:

```
mortal(X) :- homem(X) .  
homem(platão) .
```
- A partir deste programa, ao se perguntar se `mortal(platão)` é verdadeiro, a máquina de inferência do Prolog responde que sim (*yes*).

Sumário

- 1 **Representação do Conhecimento**
 - Inferência Lógica do Prolog
 - **Encadeamento Regressivo do Prolog**
 - Cláusulas Definidas
- 2 **Relações entre Objetos**
 - Exemplo
 - Incluindo Regras
 - Regra recursiva
- 3 **Encadeamento lógico do Prolog**
 - Como Prolog responde questões
 - Significados declarativo e procedimental
 - Dados estruturados do Prolog: Listas

Encadeamento Regressivo do Prolog

- A linguagem Prolog trabalha com encadeamento regressivo. Por exemplo, seja o seguinte programa:

```
p(X, Y) :- q(X, Y).  
q(a, b).
```

- E se coloque a seguinte questão:

```
?- p(a, b).
```

- Prolog faz as instanciações $X = a$ e $Y = b$, e dispara a primeira regra, ou seja, troca $p(a, b)$ por $q(a, b)$.
- (Na verdade, como $p(X, Y) :- q(X, Y)$ corresponde à implicação da lógica $q(X, Y) \rightarrow p(X, Y)$, então o Prolog está trocando o conseqüente da implicação $p(a, b)$ pelo seu antecedente $q(a, b)$, ou seja, encadeamento regressivo).

Sumário

- 1 **Representação do Conhecimento**
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - **Cláusulas Definidas**
- 2 **Relações entre Objetos**
 - Exemplo
 - Incluindo Regras
 - Regra recursiva
- 3 **Encadeamento lógico do Prolog**
 - Como Prolog responde questões
 - Significados declarativo e procedimental
 - Dados estruturados do Prolog: Listas

Cláusulas Definidas

- Uma cláusula (disjunção de literais) consiste de literais positivos e negativos:

$$p_0 \vee p_1 \vee \dots \vee \neg n_0 \vee \neg n_1 \vee \dots \quad (4)$$

- Usando a lei de De Morgan:

$$\neg p \vee \neg q \equiv \neg(p \wedge q) \quad (5)$$

- pode-se expressar as cláusulas como uma única implicação:

$$(n_0 \wedge n_1 \wedge \dots) \rightarrow (p_0 \vee p_1 \vee \dots) \quad (6)$$

Cláusulas Definidas

- Prolog não é baseado na forma clausal completa, mas sim num subconjunto bem menos expressivo, as *cláusulas de Horn*, que são cláusulas com no máximo um literal positivo.
- Existem então apenas três tipos de cláusulas de Horn:
 - *cláusulas unitárias*: com um literal positivo da forma p_0 (ou, equivalentemente, $\rightarrow p_0$).
 - *cláusulas não unitárias*: com um literal positivo e um ou mais literais negativos, isto é, da forma $p_0 \vee \neg n_0 \vee \neg n_1 \vee \dots$ (ou, equivalentemente, $(n_0 \wedge n_1 \wedge \dots) \rightarrow p_0$).
 - *cláusulas negativas*: sem literais positivos, com um ou mais literais negativos, isto é, da forma $\neg n_0 \vee \neg n_1 \vee \dots$ (ou, equivalentemente, $(n_0 \wedge n_1 \wedge \dots) \rightarrow$).

Cláusulas Definidas

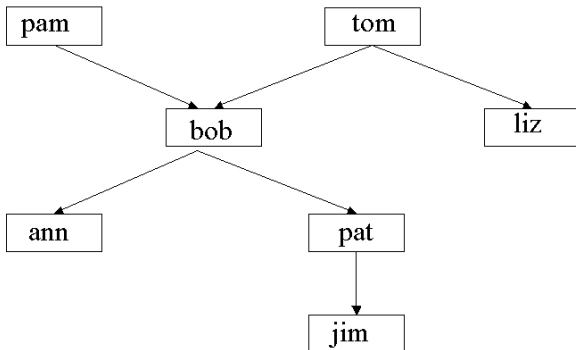
- Os dois primeiros tipos de cláusulas de Horn são conhecidas como *cláusulas definidas* porque têm exatamente um literal positivo - uma única conclusão definida para a implicação - ao contrário de cláusulas gerais com seu conseqüente potencialmente disjuntivo e indefinido.

Sumário

- 1 Representação do Conhecimento
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - Cláusulas Definidas
- 2 Relações entre Objetos
 - Exemplo
 - Incluindo Regras
 - Regra recursiva
- 3 Encadeamento lógico do Prolog
 - Como Prolog responde questões
 - Significados declarativo e procedimental
 - Dados estruturados do Prolog: Listas

Exemplo: Relações familiares

- Prolog é adequada para resolver problemas que envolvem objetos e relações entre objetos.
- Veja o exemplo da árvore familiar da figura abaixo [1]:



Exemplo: Relações familiares

- O fato de que Tom é “pais” (pai/mãe) de Bob se escreve:
`pais (tom, bob) .`
`pais: relação`
`tom, bob: argumentos`
- Os fatos na árvore familiar do exemplo são:
`pais (pam, bob) .`
`pais (tom, bob) .`
`pais (bob, ann) .`
`pais (bob, pat) .`
`pais (tom, liz) .`
`pais (pat, jim) .`

Exemplo: Relações familiares

- São 6 cláusulas declarando fatos sobre a relação “pais.”
Uma vez comunicado ao sistema Prolog os fatos sobre a relação “pais,” pode-se colocar questões:

```
? - pais(bob,pat) .
```

```
yes
```

```
? - pais(liz,pat) .
```

```
no
```

```
? - pais(X,liz) .
```

```
X = tom
```

- Outra relação, avós:

```
? - pais(Y,jim),pais(X,Y) .
```

```
X = bob
```

```
Y = pat
```

Resumo

- Uma relação é definida pelo estabelecimento das n-uplas de objetos que satisfazem a relação.
- Um programa Prolog é constituído de cláusulas. Cada cláusula termina com um ponto.
- Os argumentos das relações podem ser (entre outras coisas) constantes ou variáveis.
- Perguntas ao sistema Prolog são constituídas de um ou mais objetivos. Uma seqüência de objetivos separados por vírgulas significa a conjunção desses objetivos.

Sumário

- 1 Representação do Conhecimento
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - Cláusulas Definidas
- 2 **Relações entre Objetos**
 - Exemplo
 - **Incluindo Regras**
 - Regra recursiva
- 3 Encadeamento lógico do Prolog
 - Como Prolog responde questões
 - Significados declarativo e procedimental
 - Dados estruturados do Prolog: Listas

Estendendo o exemplo através de regras

- Vai-se incluir uma nova informação para a base de dados: o sexo das pessoas. Pode-se usar uma relação unária:
feminino(pam) .
masculino(tom) .
- ou usar uma relação binária:
sexo(pam, feminino) .
sexo(tom, masculino) .

Estendendo o exemplo através de regras

- Para a também nova relação “filhos” (filho/filha), que é o inverso da relação pais.

cabeça *corpo*
filhos(*Y*, *X*) :- pais(*X*, *Y*) .
(conclusão) **(condição)**

- em Linguagem de Predicados de Primeira Ordem:

$$\forall X \forall Y (\text{pais}(X, Y) \rightarrow \text{filhos}(Y, X)) \quad (7)$$

Estendendo o exemplo através de regras

- A diferença principal entre fatos e regras, é que o fato é uma cláusula incondicionalmente verdadeira, e a regra é uma cláusula que pode ser verdade dependendo se alguma condição for verdade.
- Como as regras são usadas em Prolog? Para a verificação da veracidade do seguinte fato:

```
? - filhos(liz,tom). % liz é filha de tom?
```

o procedimento do Prolog, para executar a prova de tal fato é o seguinte:

Estendendo o exemplo através de regras

- 1 Procura tal fato na “base de conhecimento.” Não existe tal fato.
- 2 Procura se existe uma regra sobre a relação filhos. Como existe, aplica a particular instanciação $X = tom$ e $Y = liz$, ou seja, `filhos(liz,tom) :- pais(tom,liz).`
- 3 O objetivo original `filhos(liz,tom)` é substituído por um novo objetivo: `pais(tom,liz).`
- 4 O novo objetivo é encontrado como um fato na “base de conhecimento” e o Prolog responde `yes.`

Resumo

- Cláusulas em Prolog são constituídas de cabeça e corpo.
- O corpo é uma lista de objetivos separados por vírgulas, esta entendida como conjunção de objetivos.
- Cláusulas em Prolog: fatos, regras e questões:
 - *Fatos* declaram coisas verdadeiras e possuem só cabeça.
 - *Regras* declaram coisas que são verdade dependendo de uma condição e possuem cabeça e corpo.
 - *Questões*: através delas o usuário pode perguntar ao programa sobre a verdade de certas coisas e possuem somente corpo.
- Durante a execução, uma variável pode ser instanciada por algum objeto.
 - Variáveis são assumidas ser universalmente quantificadas e são lidas como *para todo*.
 - Leituras alternativas são possíveis para variáveis que aparecem somente no corpo.

Sumário

- 1 Representação do Conhecimento
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - Cláusulas Definidas
- 2 **Relações entre Objetos**
 - Exemplo
 - Incluindo Regras
 - **Regra recursiva**
- 3 Encadeamento lógico do Prolog
 - Como Prolog responde questões
 - Significados declarativo e procedimental
 - Dados estruturados do Prolog: Listas

Definição recursiva de regra

- Para definir uma nova relação, predecessor, vai-se fazê-lo da seguinte forma:

```
predecessor(X, Z) :- pais(X, Z) .
```

```
predecessor(X, Z) :- pais(X, Y) ,
```

```
predecessor(Y, Z) .
```

- Note que a definição do predicado predecessor usa recursividade.
- A programação recursiva é um dos princípios fundamentais da programação em Prolog.
- A relação predecessor é definida por duas cláusulas, o que consiste em um **procedimento**.

Sumário

- 1 **Representação do Conhecimento**
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - Cláusulas Definidas
- 2 **Relações entre Objetos**
 - Exemplo
 - Incluindo Regras
 - Regra recursiva
- 3 **Encadeamento lógico do Prolog**
 - **Como Prolog responde questões**
 - Significados declarativo e procedimental
 - Dados estruturados do Prolog: Listas

Como Prolog responde questões

- Uma questão em Prolog é uma seqüência de objetivos (um ou mais).
- Prolog tenta satisfazer os objetivos, isto é, demonstrar que os objetivos seguem logicamente a partir dos fatos e regras do programa.
- Se as questões contêm variáveis, Prolog também tenta achar a particular instanciação que satisfaz os objetivos.
- Os fatos e regras são aceitos como um conjunto de axiomas (hipóteses).

Como Prolog responde questões

- A questão do usuário é aceita como uma possível tese de um teorema.
- Prolog tenta provar esse teorema (demonstrar que ele segue logicamente dos axiomas).
- Prolog realiza encadeamento lógico regressivo, ou seja, a partir da meta, aplica as regras (no sentido regressivo) e termina nos fatos.
- Pode-se dizer também que Prolog parte da negação (implícita) da meta até chegar a cláusula vazia, utilizando a estratégia por preferência unitária (demonstração por absurdo): **refutação**.

Como Prolog responde questões

- Exemplo: seqüência de prova da conjectura (tese):

?- predecessor(tom, pat) .

- *Observação*: note o sinal “-” após a interrogação no prompt do interpretador Prolog.
- Esse sinal representa que a cláusula meta deve ser negada antes do início da prova.
- Essa negação é feita internamente pelo mecanismo de inferência do Prolog.
- Mesmo porque, a questão Prolog deve ser um (ou mais) literal negativo!

Como Prolog responde questões

- 1 Prolog procura um fato que combine com o objetivo:
 - se sim: yes
 - se não: vai para o passo 2.

Quando Prolog encontra o fato que unifica com a meta, aplica-se a regra da resolução (literal negativo da meta e literal positivo do fato se cancelam, resultando na cláusula vazia). A razão do Prolog procurar primeiro o fato é que a estratégia usada é a preferência unitária e o fato é cláusula unitária. Apesar da estratégia por preferência unitária não ser completa, é a mais eficiente de todas. Caso a busca não seja bem sucedida, o Prolog realiza o *backtracking* e uma nova busca é efetuada.

Como Prolog responde questões

2 Prolog procura uma regra cuja cabeça combine com o objetivo:

- se não: no
- se sim: vai para o passo 3.

Se não há fato e nem regra cuja cabeça (literal positivo), então não há como aplicar a regra da resolução. Portanto não há solução.

Como Prolog responde questões

3 Prolog “dispara” a regra para a particular instanciação

```
predecessor(X, Z) :- pais(X, Z).
```

```
X = tom
```

```
Z = pat
```

e substitui o objetivo corrente por

```
? - pais(tom, pat).
```

A cabeça da regra corresponde ao conseqüente da implicação lógica (conclusão) e o corpo ao antecedente (condição). Quando Prolog dispara a regra, na verdade “troca” a cabeça pelo corpo, ou seja, “retorna” do conseqüente para o antecedente na implicação lógica (encadeamento regressivo). Em termos de regra da resolução: trocar a cabeça pelo corpo significa “cancelar” o literal negativo da meta com o literal positivo da cabeça, resultando no literal negativo do corpo da cláusula Prolog.

Como Prolog responde questões

4 Prolog procura uma cláusula cuja cabeça combina com o objetivo (novo)

- se sim e cláusula = fato: yes
- se sim e cláusula = regra: “dispara” a regra
- se não: *backtrack*

Repete-se o processo. Caso encontre um fato, termina, pois no encadeamento regressivo o encadeamento termina no fato. E na refutação, encontrar o fato significa chegar à cláusula vazia (lembre-se: havia um literal negativo e o fato é seu literal complementar).

Como Prolog responde questões

5 Prolog “dispara” a segunda regra

```
predecessor(X,Z) :- pais(X,Y),  
predecessor(Y,Z).
```

X = tom

Z = pat

e substitui o objetivo corrente

```
? - pais (tom,Y), predecessor (Y,pat).
```

Como Prolog responde questões

- 6 Prolog procura o primeiro objetivo, que combina com o fato `pais(tom,bob)`, instanciando `Y = bob`.
- 7 Resta analisar
? - `predecessor(bob,pat)` .
- 8 Prolog “dispara” a primeira regra novamente e substitui o objetivo
? - `pais(bob,pat)` .
que finalmente é encontrado como fato no banco de conhecimento.

Sumário

- 1 Representação do Conhecimento
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - Cláusulas Definidas
- 2 Relações entre Objetos
 - Exemplo
 - Incluindo Regras
 - Regra recursiva
- 3 Encadeamento lógico do Prolog
 - Como Prolog responde questões
 - **Significados declarativo e procedimental**
 - Dados estruturados do Prolog: Listas

Significados declarativo e procedimental

- É possível distinguir entre os significados *declarativo* (o que) do *procedimental* (o como) de um programa Prolog.
- A habilidade do Prolog realizar muitos detalhes procedimentais por si mesmo é considerado uma de suas vantagens.

Resumo

- A programação em Prolog consiste em definir relações e fazer questões sobre essas relações.
- Prolog consiste de cláusulas de três tipos: fatos, regras e questões.
- Uma relação pode ser especificada por fatos simplesmente estabelecendo as n-uplas de objetos que satisfazem a relação, ou estabelecendo regras a respeito da relação.
- Um procedimento é um conjunto de cláusulas a respeito de uma mesma relação.

Resumo

- Questionar sobre relações, através de perguntas, lembra questionar um banco de dados. A resposta consiste de um conjunto de objetos que satisfazem a questão.
- A resposta é obtida através de um complexo processo que envolve inferência lógica, exploração entre alternativas, *backtracking*.
- Existem dois significados nos programas Prolog: declarativo e procedimental. O declarativo é vantajoso do ponto de vista da programação. Apesar de que os detalhes procedimentais muitas vezes devem ser considerados pelo programador.

Sumário

- 1 Representação do Conhecimento
 - Inferência Lógica do Prolog
 - Encadeamento Regressivo do Prolog
 - Cláusulas Definidas
- 2 Relações entre Objetos
 - Exemplo
 - Incluindo Regras
 - Regra recursiva
- 3 Encadeamento lógico do Prolog
 - Como Prolog responde questões
 - Significados declarativo e procedimental
 - **Dados estruturados do Prolog: Listas**

Listas

- A lista é uma estrutura de dados simples muito usada em programação não-numérica.
- Uma lista é uma seqüência de qualquer número de itens, tais como *ana*, *tenis*, *tom*, *esqui*. Tal lista pode ser escrita em Prolog como:

```
[ana, tenis, tom, esqui]
```
- Esta é entretanto, apenas a aparência externa das listas, pois todos os objetos estruturados em Prolog são árvores.

Listas

- Como se pode representar uma lista como um objeto Prolog padrão?
- Deve-se considerar dois casos: ou a lista está vazia ou não vazia.
- No primeiro caso, a lista é simplesmente escrita como o átomo do Prolog, `[]`.
- No segundo caso, a lista pode ser vista como consistindo de duas coisas:
 - 1 o primeiro item, chamado a *cabeça* da lista;
 - 2 a parte remanescente da lista, chamada de *cauda*.

Resumo

- Uma lista é uma estrutura de dados que ou está vazia ou consiste de duas partes: uma cabeça e uma cauda. A cauda por sua vez também é uma lista.
- As listas são manipuladas pelo Prolog com um caso especial de árvores binárias. Para melhorar a legibilidade, Prolog provê uma notação especial para listas:

```
[Item1, Item2, ...]
```

OU

```
[Cabeça | Cauda]
```




OU

```
[Item1, Item2, ... | Outros]
```



Conclusões

- Prolog é a linguagem para programação lógica e pode fazer muitas coisas. Mas tem quatro fraquezas lógicas fundamentais [5]:
 - 1 Prolog não permite fatos ou conclusões disjuntivos, ou seja, sentenças onde uma ou mais coisas são verdadeiras, mas não se sabe quais.
 - 2 Prolog não permite fatos ou conclusões negativos, isto é, sentenças diretas onde alguma coisa é falsa.
 - 3 Prolog não permite que muitos fatos, conclusões ou regras tenham quantificação existencial, isto é, sentenças onde exista algum valor de variável, ainda que não se saiba qual, tal que o predicado que a contém seja verdadeiro.
 - 4 Prolog não permite diretamente lógica de segunda ordem, nomes de predicados como variáveis, isto é, sentenças sobre P onde P significa um nome de predicado.

Referências I

-  [1] Bratko, I.
Prolog Programming for Artificial Intelligence.
3rd edition. Addison-Wesley Pub Co., 2000.
-  [2] Casanova, M. A., Giorno, F. A. C., Furtado, A. L.
Programação em Lógica e a Linguagem Prolog.
Ed. Edgard Blücher Ltda., 1987
-  [3] Pereira, F. C. N., Shieber, S. M.
Prolog and Natural Language Analysis.
CSLI - Center for the Study of Language and Information,
1987.

Referências II

-  [4] Rosa, J. L. G.
Fundamentos da Inteligência Artificial.
Editora LTC. Rio de Janeiro, 2011. *No prelo.*
-  [5] Rowe, N. C.
Artificial Intelligence Through Prolog.
Prentice Hall, 1988.