



Programação Lógica e Prolog

Prof. Carlos Lopes




Programação Lógica

- Um programa lógico é um conjunto de axiomas definindo relacionamento entre objetos.
- Uma computação de um programa lógico é uma dedução de conseqüências do programa.
- A linguagem Prolog é a linguagem mais representativa do paradigma de programação lógica.




Relação entre Prolog e Lógica

- A sintaxe de Prolog é herdada da lógica de primeira ordem escrita na forma clausal (forma na qual os quantificadores não aparecem explicitamente escritos) e restritas a cláusulas de Horn (cláusulas com no máximo um literal positivo).
- Prolog usa uma técnica de resolução para provar teoremas chamada SLD.




Aplicações de Programação Lógica

- Sistemas Baseados em Conhecimento
 - sistemas que aplicam mecanismos automatizados de raciocínio para a representação e inferência de conhecimento
- Bancos de Dados 'Inteligentes'
 - sistemas que empregam 'agentes' de busca de dados com base em critérios
- Sistemas Especialistas
 - sistemas que emulam a especialização humana em algum domínio específico.
- Processamento da Linguagem Natural
 - usada para desenvolvimento de ferramentas para a comunicação homem-máquina em geral e para a construção de interfaces



A Linguagem Prolog

- Histórico:
 - 1972: Kowalski - Lógica como formalismo executável
 - 1973: Colmerauer / GIA / Universidade de Marseille - primeiro interpretador PROLOG
 - 1979: Pereira e Warren / Universidade de Edinburgh: primeiro compilador PROLOG (PROLOG)
- Implementações: diferem na sintaxe e nas bibliotecas
 - Exemplos: Arity Prolog, Turbo Prolog, Strawberry, LPA Prolog, SWI-Prolog, Visual Prolog etc.



Um Programa Prolog

- Prolog é uma linguagem adequada para processamento simbólico. É indicada para resolver problemas que envolvam objetos e seus relacionamentos.
- Exemplo: o relacionamento familiar. Em Prolog para especificar que Carlos é pai de Ana escreve-se:


```
pai(carlos,ana).
```

Exemplo de Programa Prolog (cont.)

- Usando a terminologia de Prolog, o nome do relacionamento (pai) é chamado de predicado. Os objetos relacionados (carlos e ana) são constantes. Predicados e constantes são escritos com letras minúsculas.
- Continuando com o exemplo da família deve-se especificar que Carlos tem mais uma filha: Juliana. Assim temos que:
`pai(carlos,ana).`
`pai(carlos,juliana).`

Estendendo o Programa

- O programa neste momento constitui-se de duas cláusulas. Cada uma destas cláusulas declara um fato sobre a relação *pai*. Observe que os fatos são finalizados por um ponto.
- Exercício: estender o programa Prolog por incluir os relacionamentos *mae*, *irma*, *homem* e *mulher*.

Estendendo o Programa: consultas

- Após comunicar este programa ao sistema Prolog podemos fazer com que ele nos responda a perguntas. Isto se faz por colocar o sistema num modo em que isto é possível. Em geral o símbolo `?-` (que é um prompt) representa este estado.
- Por exemplo, podemos perguntar: Carlos é pai de Ana? Em Prolog isto se faz por escrever:
`?- pai(josé,ana).`

Estendendo o Programa: consultas

- Prolog consegue deduzir que a resposta é verdadeira pois o fato `pai(carlos,ana)` está presente na base de fatos. Assim ele responde:
`yes`
- Para a pergunta:
`?- pai(josé,ana).`
a resposta seria:
`no`

Estendendo o programa: consultas

- Prolog aceita perguntas como metas a serem satisfeitas.
- A resposta para uma pergunta pode ser positiva (*yes*) ou negativa (*no*).
- No caso de uma resposta positiva dizemos que a meta foi satisfeita e que a meta obteve sucesso.
- No caso de uma resposta negativa dizemos que a meta não foi satisfeita e que a meta não teve sucesso.

Estendendo o Programa: consultas

- Perguntas mais interessantes podem ser formuladas. Exemplo: Quem é filho de Carlos?
`?-pai(carlos,X)`
- Desta vez a resposta não será *yes* ou *no*. Prolog nos dirá qual é o valor de X que tornará a cláusula verdadeira.
`X=ana`

Estendendo o Programa: consultas

- X é chamada de variável e representa um objeto desconhecido. Variáveis são escritas com letra inicial maiúscula.
- Conforme especificado anteriormente, temos que Carlos tem mais de uma filha. Para dizer a Prolog que queremos outras soluções digitamos ; na frente da resposta dada:
?- pai(carlos,X)
X=ana;
X=juliana;
no

Estendendo o Programa: consultas

- Exercício: escrever em Prolog as seguintes consultas:
 - Quem é pai de Ana?
 - Quem é pai de Quem?
- A consulta
?- pai(carlos,X).
pode ser lida como:
Existe um individuo X tal que carlos é pai de X?
Dizemos que X é quantificada existencialmente.

Estendendo o Programa: consultas

- Uma pergunta mais complicada: Quem é o avô de Ana?
- Obs:
 - Não foi dito ao Prolog nenhum relacionamento avô
 - A consulta tem que ser decomposta em dois passos:
 - Quem é o pai de Ana? Assuma que é algum Y.
 - Quem é o pai de Y? Assuma que é algum X.

Estendendo o Programa: consultas

- Tal consulta composta é escrita em prolog da seguinte forma:
?- pai(Y,ana),pai(X,Y).
- A consulta pode ser lida da seguinte forma:
Encontre X e Y de tal forma que
pai(Y,ana) e pai(X,Y) sejam satisfeitos.

Estendendo o Programa: consultas

- **Exercícios:** escrever em Prolog as seguintes consultas:
 - Quem são os netos de João?
 - Juliana e Ana têm o mesmo pai?
- Consultas ao sistema constituem-se de uma ou mais metas.
- A sequência de metas
pai(X,ana),pai(X,juliana)
Significa a conjunção de metas:
X é pai de Ana e X é pai de Juliana.

Estendendo o programa: regras

- Poderíamos estender o programa por acrescentar a relação filho (relação inversa de pai).
filho(ana,carlos). filho(juliana,carlos).
- Entretanto, a relação filho pode ser definida de uma forma mais elegante sabendo-se que filho é a relação inversa de pai e a relação pai já foi definida. Esta alternativa pode ser baseada na seguinte sentença lógica:
Para todo X e Y se X é pai de Y então Y é filho de X.

Estendendo o programa: regras

- A cláusula Prolog que tem o mesmo significado é
`filho(X,Y) :- pai(Y,X).`
- Tais tipos de cláusulas Prolog são denominadas regras.
- As regras especificam relações que podem ser verdade dado que alguma condição seja satisfeita.

Estendendo o programa: regras

- Portanto dizemos que as regras possuem duas partes:
 - Condição (ou antecedente ou corpo da regra): parte direita da regra (que vem após `:-`).
 - Conclusão (ou consequente ou cabeça da regra): parte esquerda da regra (que vem antes de `:-`).
- Se a condição `pai(X,Y)` é verdade então uma consequência lógica disto é `filho(Y,X)`

Estendendo o programa: regras

- Como as regras são usadas por Prolog? Seja a consulta `?- filha(ana,carlos).`
 - Verifica se existem fatos sobre filha no programa (não existe!)
 - Verifica se existem regras sobre filha (foi definida anteriormente). A regra é geral pois é aplicada a quaisquer objetos X e Y. Portanto ela pode ser aplicada a objetos como ana e carlos. Para aplicá-la a ana e carlos X deve ser substituído por ana e Y deve ser substituído por carlos. Dizemos que as variáveis X e Y ficam instanciadas para:
`X=ana` e `Y=carlos`

Estendendo o programa: regras

- Como as regras são usadas por Prolog? (cont.)
 - Após a instânciação tem-se um caso especial da regra geral:
`filha(ana,carlos) :- pai(carlos,ana).`
 - Agora Prolog tenta descobrir se a condição é verdadeira. Desta forma a meta inicial `filha(ana,carlos)` é substituída pela submeta `pai(carlos,ana)`.
 - Esta nova meta é trivial pois corresponde a um fato do nosso programa. Portanto a condição é verdadeira e Prolog deduz que a conclusão é verdadeira e responde com `yes`.

Estendendo o programa: regras

- Definindo uma regra para a relação avo:
`avo(X,Y) :- pai(X,Z),pai(Z,Y).`
lê: para todo X,Y e Z X é avô de Y se X é pai de Z e Z é pai de Y.
- Uma vírgula entre duas condições indica a conjunção das condições significando que ambas condições devem ser verdadeiras.
- **Exercício:** definir a regra com cabeça `irma(X,Y)`.

Estendendo o programa: definindo regras recursivas

- Definindo a relação ancestral. Esta relação será definida em termos da relação pai. A definição completa pode ser expressa por meio de duas regras:
 - 1) `ancestral(X,Y) :- pai(X,Y).`
 - 2) `ancestral(X,Y) :- ...`
- A segunda regra é mais complicada!

Estendendo o programa: definindo regras recursivas

- 1) `ancestral(X,Y) :-
 pai(X,Z),
 pai(Z,Y).`
 - 2) `ancestral(X,Y) :-
 pai(X,Z),
 pai(Z,Z1),
 pai(Z1,Y).`
 - 3) `ancestral(X,Y) :-
 ...`
- Este programa torna-se longo e funciona parcialmente.

Estendendo o programa: definindo regras recursivas

- Definição correta e mais elegante:
`ancestral(X,Y) :-
 pai(X,Z),
 ancestral(Z,Y).`
- Portanto, as regras a seguir definem completamente a relação ancestral:
`ancestral(X,Y) :-
 pai(X,Y).`
`ancestral(X,Y) :-
 pai(X,Z),
 ancestral(Z,Y).`

Como Prolog responde a consultas?

- Uma consulta para Prolog é sempre uma sequência de uma ou mais metas.
- Para responder uma pergunta Prolog tenta satisfazer todas as metas.
- Satisfazer uma meta significa demonstrar que a meta é verdadeira assumindo que todas as relações são verdadeiras.
- Em outras palavras significa demonstrar que a meta é consequência lógica dos fatos e regras do programa.

Como Prolog responde a consultas?

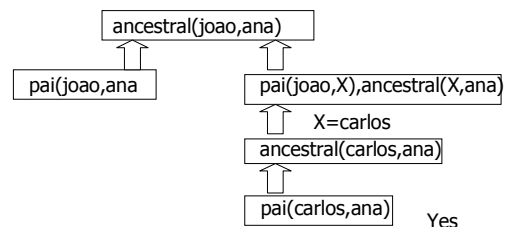
- Se uma consulta tem variáveis, Prolog tenta encontrar quais são os objetos particulares (no lugar das variáveis) para os quais as metas são satisfeitas. A instânciação destas variáveis é mostrada ao usuário.
- Se Prolog não consegue mostrar para alguma instânciação que as metas são consequências lógicas do programa então Prolog responde *no*.

Como Prolog responde a consultas?

- Uma visão apropriada da interpretação de um programa Prolog em termos matemáticos:
 - Um programa Prolog constitui-se de fatos e regras, o que corresponde a um conjunto de axiomas.
 - A pergunta do usuário deve ser interpretada como um teorema a ser provado.

Como Prolog responde a consultas?

- Exemplo: mostrando como Prolog prova `ancestral(joao,ana)`.





Significado Declarativo e Procedimental de Programas

- O significado declarativo de um programa diz respeito somente as relações definidas pelo programa.
- O significado procedimental determina como a saída é obtida.
- Os aspectos declarativos dos programas são mais fáceis de entender que os detalhes procedimentais.
- Entretanto, os detalhes procedimentais não podem ser completamente ignorados por razões de eficiência.