Prolog

MATA56 - Paradigmas de Linguagens de Programação,
UFBA, 2016.1. Prof. Rodrigo Rocha
(rodrigo@dcc.ufba.br)

Iniciando

Conceitos

- Baseada na lógica de primeira ordem (lógica de predicados)
- Concebida na década de 1970
- Usada nos seguintes domínios
  - inteligência artificial
  - linguística computacional
  - sistemas de recomendação

Ferramenta

SWI Prolog

- Site: `http://www.swi-prolog.org/`
- Instalação (Linux): `sudo apt-get install swi-prolog`
- Interpretador online: `http://swish.swi-prolog.org/`
  - Exemplos: `http://swish.swi-prolog.org/example/examples.swinb`

Mão na massa

Base de conhecimento

- Crie um arquivo `base.pl`, com o seguinte conteúdo:

```
humano(joao).
```

- Esse arquivo é nossa *base de conhecimento*.
- Através do arquivo, estamos informando que `joao` é humano.
- `humano(joao)`, nesse contexto, é um *fato*.

Consultas

- Agora abra o interpretador Prolog: `prolog`
- Vai aparecer o *prompt* `?-`
- Digite `consult('base.pl')` para carregar a base de conhecimento.
- Então digite:

```
humano(joao).
```

- Você está fazendo uma *consulta*: `joao` é humano?
- O Prolog responde `true.`

Consultas

- Dica:
  - use `prolog -s base.pl` para abrir o Prolog já carregando o arquivo `base.pl`.

Consultas

- Outras consultas para testar:
  - `humano(shrek).`
  - `humano(caetano).`
  - `humano(X).`
- `false` significa "não é possível provar que é verdadeiro".
  - *hipótese do mundo fechado*

Variáveis

- Em `humano(X)`, `X` é uma *variável*.
- Em Prolog, variáveis começam com uma letra *maiúscula*.
- O Prolog retorna todos os valores de `X` para os quais `X` é humano.

Exemplo mais completo

- Base (arquivo `base.pl`):

```
humano(joao).
humano(maria).
humano(pedro).
mulher(maria).
homem(joao).
homem(pedro).
```

- Consulta: `humano(X).`
- Tecle `;` para continuar a busca.

Regras

- Se sabemos que `joao` é homem, não deveríamos precisar dizer que ele é humano, pois *todo homem é humano*!
- A mesma lógica vale para as mulheres!
- Podemos inserir *regras* na nossa base de conhecimento

Regras

- Regras
  - Se X é homem, então X é humano.
  - Se X é mulher, então X é humano.
- Atualize o arquivo `base.pl`:

```
humano(X) :- homem(X).
humano(X) :- mulher(X).
mulher(maria).
homem(joao).
homem(pedro).
```

- Note o uso da variável X.
- A condicional é invertida.

Busca

- Agora realize a consulta `humano(joao).` Qual é o retorno?
- Realize também a consulta `humano(X).` Qual é o retorno?

Comentários

- Comentários em Prolog podem ser escritos assim:

```
% este é um comentário de uma linha
/*
 Este é um
 comentário
 de muitas
 linhas.
*/
```

Comandos úteis

- `halt.` - fecha o interpretador.
- `listing.` - mostra fatos e regras carregados.
- `help.` - mostra ajuda.
- `write('oi').` - escreve “oi”.
  - `nl.` - escreve uma quebra de linha (**n**ew **l**ine)
- `assert( fato ).` - adiciona um fato à base
  - `retract( fato )` - remove um fato da base

Conceitos

Definições

- Um programa Prolog é composto de *cláusulas* terminadas por ponto.

```
humano(X) :- homem(X).
humano(X) :- mulher(X).
mulher(maria).
homem(joao).
```

- Uma cláusula pode ser um *fato* ou uma *regra*.

Fatos e regras

- Um programa Prolog é uma *base de conhecimento*, que contém
  - **fatos**, afirmações consideradas verdadeiras
  - **regras**, a partir das quais se deduzem fatos não-declarados
- No interpretador do Prolog podem ser realizadas **consultas**
  - o retorno pode ser true/false;
  - as consultas podem conter **variáveis**;
  - nesse caso, o retorno são os valores das variáveis que tornam a expressão verdadeira

Fatos

- Exemplo: `mulher(maria)` é um fato
- `mulher` é um *predicado*.
- Predicados contêm *argumentos* separados por vírgulas.
- Exemplo: `pai(joao, maria)` (`joao` é pai de `maria`)
  - `pai` é um predicado binário (dois argumentos)
  - referimos a ele como `pai/2` (predicado com 2 argumentos)
- `maria` é um *átomo*, i.e., uma constante não-numérica, um símbolo

Termos

- Fatos, números, átomos, predicados e listas são considerados *termos*.
- (Listas serão vistas mais adiante)
- Termos complexos são da forma `functor(argumento1, argumento2, ...)`
  - o `functor` é um átomo
  - argumentos podem ser qualquer termo

Regras

- Exemplo: `humano(X) :- homem(X)` é uma **regra**
- Toda regra possui uma **cabeça** e um **corpo**.
  - Formato: `cabeça :- corpo`
- `X` é uma variável
- O corpo é composto de um ou mais termos separados por vírgulas.
- O corpo representa as condições que precisam ser satisfeitas para se chegar à conclusão representada pela cabeça.

Regras

- Exemplo:

```
pai(X, Y) :- progenitor(X, Y), homem(X).
```

- `X` é pai de `Y` se `X` é progenitor de `Y` **e** `X` é homem (para quaisquer `X` e `Y`).
- Vírgula = **e** (conjunção lógica).
- Ponto-e-vírgula = **ou** (disjunção lógica).
  - Ex.: `avo(X, Z) :- pai(X, Y), pai(Y, Z) ; pai(X, Y), mae(Y, Z).`
  - Equivalente a escrever duas cláusulas:
    - `avo(X, Z) :- pai(X, Y), pai(Y, Z).`
    - `avo(X, Z) :- pai(X, Y), mae(Y, Z).`

Consultas (ou metas)

- No prompt interativo do Prolog, escrevem-se *consultas*
  - também conhecidas como *metas*
  - a meta do Prolog é deduzir se a expressão é verdadeira
- Exemplos:
  - `pai(X, X)` - retorna pessoas quem sai pais de si próprias
  - `pai(X, Y)` - retorna pais e filhos; note que `X` pode ser igual a `Y`.

Variáveis

- Variáveis começam com letra maiúscula.
- Exemplos: `X`, `Y`, `Pessoa`.
- O escopo de uma variável é a cláusula em que se encontra.
  - Se `X` aparecer em duas cláusulas diferentes, são duas variáveis diferentes.
- Em uma consulta, podemos usar a variável anônima `_` para denotar uma variável que não nos interessa saber o valor.
  - `pai(X, Y)` - retorne todos os pares (`X`, `Y`) nos quais `X` é pai de `Y`.
  - `pai(X, _)` - retorne todos os `X` tais que `X` é pai de alguém.

Consultas compostas

- Uma consulta também pode conter vários termos separados por vírgulas.
- Nesse caso são retornadas os valores que satisfazem a todos os termos simultaneamente.
- Exemplo:
  - `progenitor(X, Y), progenitor(Y, Z).`
  - Retorna todos as pessoas que possuem neto.

Exercícios básicos

- Escreva um programa em Prolog para resolver o problema das 3 casas.
- `http://swish.swi-prolog.org/example/movies.pl`
- `http://cs.union.edu/~striegnk/courses/esslli04prolog/practical.day1.php`
- `http://cs.union.edu/~striegnk/courses/esslli04prolog/practical.day1.php?s=practical.day1.node6`