Paradigmas de Programação

Prof. Maicon R. Zatelli

Prolog - Programação Lógica
Predicados Dinâmicos

Universidade Federal de Santa Catarina
Florianópolis - Brasil
2018/1

Prolog

Em Prolog, é possível adicionar e remover fatos e regras em tempo de execução.

Inicialmente, precisamos definir quais predicados são dinâmicos. Para isso, informamos o predicado e a sua aridade.

```
:- dynamic genitor/2, homem/1, pai/2.
```

Prolog

Adicionando novos fatos e regras

- **asserta(Termo)** permite adicionar um novo termo (que pode ser uma regra ou um fato) antes de todos os demais do mesmo tipo.
- **assertz(Termo)** permite adicionar um novo termo (que pode ser uma regra ou um fato) depois de todos os demais do mesmo tipo.

```
?- assertz(homem(tom)).
true.
?- asserta(homem(bob)).
true.
?- asserta(genitor(tom,bob)).
true.
?- assertz((pai(X,Y) :- genitor(X,Y), homem(X))).
true.
```

Removendo fatos e regras

- **retract(Termo)** remove cada fato ou regra que unifica com o termo passado como parâmetro.
- **retractall(Termo)** remove todos os fatos ou regras que unificam com o termo passado como parâmetro.

```
?- retract(homem(X)).
X = bob ;
X = tom.

?- retractall(homem(X)).
true.
```

Programação Dinâmica

Prolog

Computar o n-ésimo número da sequência de Fibonacci.

```
fib(0,0) :- !.
fib(1,1) :- !.
fib(N,K) :-
    N1 is N - 1,
    N2 is N - 2,
    fib(N1,K1),
    fib(N2,K2),
    K is K1 + K2.
```

- Note que ao executar algo como **fib(35,X)** irão ser efetuados muitos cálculos repetidos, uma vez que não há nenhum fato que diga quais são os $n-1$ primeiros números da sequência de Fibonacci, exceto para F_0 e F_1.
- Tente executar **fib(35,X)** em seu computador e veja quando tempo irá demorar para calcular.

Prolog

Agora vamos armazenar os números da sequência de Fibonacci já encontrados.

```
:- dynamic fib2/2.
:- retractall( fib2(_,_) ).

fib2(0,0) :- !.
fib2(1,1) :- !.
fib2(N,K) :-
    N1 is N - 1, N2 is N - 2,
    fib2(N1,K1), fib2(N2,K2),
    K is K1 + K2,
    asserta(fib2(N,K) :- !).
```

- Inicialmente digo que **fib2/2** é um predicado de aridade 2 e é dinâmico.
- Depois removo todos os **fib2/2** já computados anteriormente (em outra execução).
- A cada novo número da sequência de Fibonacci encontrado, adiciono uma regra para ele: **asserta(fib2(N,K) :- !)**

O problema do troco

Dados valores de moedas V_i e um valor de troco T, o problema consiste em encontrar o menor número de moedas para dar o troco.

Por exemplo, assuma os valores de moedas 20, 30, e 60 e deve-se dar um troco de 110. Qual o menor número de moedas que pode ser usado para dar esse troco?

O problema do troco

Dados valores de moedas V_i e um valor de troco T, o problema consiste em encontrar o menor número de moedas para dar o troco.

Por exemplo, assuma os valores de moedas 20, 30, e 60 e deve-se dar um troco de 110. Qual o menor número de moedas que pode ser usado para dar esse troco?

- 3 moedas, uma de cada tipo.

Estratégia de solução

Estratégia de solução

- Descubro qual é a última moeda que deve ser utilizada para dar o troco T.

Estratégia de solução

- Descubro qual é a última moeda que deve ser utilizada para dar o troco T.
- Somo 1 ao total de moedas.

Estratégia de solução

- Descubro qual é a última moeda que deve ser utilizada para dar o troco T.
- Somo 1 ao total de moedas.
- Ao utilizar a última moeda, irá sobrar um valor restante de troco T', que vem do uso da última moeda, então repito esse processo até que o valor de troco restante T' seja 0 ou impossível de se usar qualquer moeda para completar o troco.

Estratégia de solução

- Descubro qual é a última moeda que deve ser utilizada para dar o troco T.
- Somo 1 ao total de moedas.
- Ao utilizar a última moeda, irá sobrar um valor restante de troco T', que vem do uso da última moeda, então repito esse processo até que o valor de troco restante T' seja 0 ou impossível de se usar qualquer moeda para completar o troco.
- Por exemplo, ao ter o troco de 110 para dar, posso querer utilizar como última moeda, a moeda de valor 20, então o troco restante T' será 90. Se usar como última moeda a moeda de valor 30, o troco restante T' será 80. Se usar como última moeda a moeda de valor 60, o troco restante T' será 50.

Estratégia de solução

- Descubro qual é a última moeda que deve ser utilizada para dar o troco T.
- Somo 1 ao total de moedas.
- Ao utilizar a última moeda, irá sobrar um valor restante de troco T', que vem do uso da última moeda, então repito esse processo até que o valor de troco restante T' seja 0 ou impossível de se usar qualquer moeda para completar o troco.
- Por exemplo, ao ter o troco de 110 para dar, posso querer utilizar como última moeda, a moeda de valor 20, então o troco restante T' será 90. Se usar como última moeda a moeda de valor 30, o troco restante T' será 80. Se usar como última moeda a moeda de valor 60, o troco restante T' será 50.
- Continua ...

Estratégia de solução

- Depois, tento descobrir o menor número de moedas necessárias para dar o troco restante, de 90, 80 e 50, e assim sucessivamente.

Estratégia de solução

- Depois, tento descobrir o menor número de moedas necessárias para dar o troco restante, de 90, 80 e 50, e assim sucessivamente.
- Cuidados: o troco restante de 80 pode ser gerado com 4 moedas de 20, ou 2 moedas (uma de 60 e outra de 20), ou 3 moedas (duas de 30 e outra de 20). Assim, note que por diversas vezes iremos ter de recalcular os mesmos valores.

Estratégia de solução

- Depois, tento descobrir o menor número de moedas necessárias para dar o troco restante, de 90, 80 e 50, e assim sucessivamente.
- Cuidados: o troco restante de 80 pode ser gerado com 4 moedas de 20, ou 2 moedas (uma de 60 e outra de 20), ou 3 moedas (duas de 30 e outra de 20). Assim, note que por diversas vezes iremos ter de recalcular os mesmos valores.
- Solução: salve sempre o menor número de moedas para computar algum valor!

Prolog

```
:- dynamic minimoMoedas/2.
:- retractall( minimoMoedas(_,_) ).

moeda(20). %Alguns valores de moedas
moeda(30).
moeda(60).

minimoMoedas(N,1) :- moeda(N), !.
minimoMoedas(N,K) :-
    N > 0,
    findall(KResto,
    (moeda(X), X < N, Resto is N - X, minimoMoedas(Resto, KResto)),
    ListaKResto),
    min_list(ListaKResto, KMinResto),
    K is KMinResto + 1,
    asserta(minimoMoedas(N,K) :- !).
```

- Tento adivinhar a última moeda a ser utilizada.
- Escolho a que utilizar o menor número de moedas e somo 1.
- Salvo o resultado.

Prolog

```
[debug]  ?- minimoMoedas(110,X).
```

Prolog

```
[debug]  ?- minimoMoedas(110,X).
X = 3.
```

Prolog - Alguns Links Úteis

- `http://www.swi-prolog.org/pldoc/man?section=dynpreds`
- `http://www.swi-prolog.org/pldoc/man?section=recdb`

Prolog

Ver atividade no Moodle