Paradigma de Programação Lógico

4 - Retrocesso e corte - Exercícios

4.1) [pip 7.9] Três números naturais A, B, C são chamados de terno Pitagórico se $A^2 + B^2 = C^2$. Escreva um predicado `terno_pitagorico(A, B, C)` que gere ternos Pitagóricos. Dica: use o gerador `int` visto em sala. Exemplo

```
?- terno_pitagorico(A, B, C).
A = 3,
B = 4,
C = 5 ;
A = 6,
B = 8,
C = 10 ;
A = 5,
B = 12,
C = 15 ;
...
```

4.2) [pp99 1.26] Defina um predicado `combinacao(K, L, C)` que é verdadeiro se `C` é uma combinação de `K` elementos de `L`. Este predicado deve ser capaz de gerar todas as combinações de `K` elementos via retrocesso. Exemplo

```
?- combinacao(3, [a, b, c d], C).
C = [a, b, c] ;
C = [a, b, d] ;
C = [a, c, d] ;
C = [b, c, d] ;
false.
```

4.3) Defina um predicado `subconjunto(L, S)` que é verdadeiro se `S` é um subconjunto de `L`. Este predicado deve ser capaz de gerar todos os subconjuntos de `L` via retrocesso. Exemplo

```
?- subconjunto([a, b, c], S).
S = [] ;
S = [a] ;
S = [b] ;
S = [c] ;
S = [a, b] ;
S = [a, c] ;
S = [b, c] ;
S = [a, b, c] ;
false.
```

4.4) Dado um conjunto de números inteiros e um inteiro S, o problema da soma dos subconjuntos consiste em verificar se existe um subconjunto não vazio cuja soma é S. Defina um predicado `soma_subconjunto(A, S, P)` que é verdadeiro se `P` é um subconjunto de `A` e a soma dos elementos de `P` é `S`. Uma estratégia simples (e ingênua) para implementar este predicado é testar os subconjunto até encontrar um que tenha a soma esperada.

```
?- soma_subconjunto([-7, -3, -2, 5, 8], 0, P).
P = [-3, -2, 5].
```

4.5) Defina um predicado `ordenacao(L, S)` que é verdadeiro se `S` é a lista `L` com os elementos ordenados. Este predicado deve implementar um algoritmo de ordenação bastante ingênuo, que testa as permutações de `S` até encontrar uma permutação ordenada. Seu predicado deve ser determinístico. Exemplo

```
?- ordenacao([7, 2, 4, 3], S).
S = [2, 3, 4, 7].
```

4.6) Defina um predicado `primo(N)` que é verdadeiro se `N` é um número primo. Seu predicado deve funcionar se `N` estiver instanciado ou não. Se `N` não estiver instanciado o seu predicado deve gerar os número primos via retrocesso. Veja os predicados pré-definidos `var` e `nonvar`. Exemplos

```
?- primo(7).
true.
?- primo(N).
N = 2 ;
N = 3 ;
N = 5 ;
N = 7 ;
...
```

4.7) [pip 11-8] Implemente um predicado para encontrar todas as formas de posicionar 4 rainhas em um tabuleiro de xadrez 4x4 de maneira que nenhuma rainha ataque outra. Uma forma de fazer este predicado é criar um gerador de permutações e testar se as rainhas foram posicionadas de maneira correta.

4.8) [lpn 6.6] Existe uma rua com três casas vizinhas com cores diferentes (vermelho, azul e verde). Em cada casa vive uma pessoa de uma nacionalidade diferente e que têm uma animal de estimação diferente. Mais alguns fatos sobre as casas:

- O inglês vive na casa vermelha.
- O espanhol tem como animal de estimação um jaguar.
- O japonês vive ao lado de quem tem uma cobra.
- Quem tem um cobra vive a esquerda da casa azul.

Defina um predicado `zebra(N)` que é verdadeiro se a pessoa com nacionalidade N tem como animal de estimação uma zebra.

4.9) Defina um predicado que resolva o problema de lógica descrito em rachacuca.com.br.

Referências

- [lpn]. Lear Prolog Now
- [pip]. Programming in Prolog.
- [pp99]. 99 problemas para resolver em (Prolog)

Licença