Inteligência Artificial 04/05 — Aula 11

1. Crie um predicado `parenteses(+L)` usando uma Gramática de Cláusulas Definidas (DCG - Definite Clause Grammar) que sucede se `L` for uma lista não vazia de parênteses emparelhados, em que os átomos `apc`, `fpc`, `apr` e `fpr` representam `'('`, `')'`, `'['` e `']'` respectivamente. Exemplos:

```
?- parenteses([apc,apr,apc,fpc,fpr,fpc,apc,apc,fpc,fpc,apr,fpr]).
yes

?- parenteses([apc,apr,apc,fpr,fpc,fpc,apc,apc,fpc,fpc,apr,fpr]).
no

?- parenteses([apc,apr,apc,fpc,fpr,fpc,apc,apc,fpc,fpc,apr]).
no
```

2. Crie um predicado `parenteses(+L,?N)`, análogo ao predicado do exercício anterior, onde o inteiro `N` corresponde ao nível de profundidade máximo de `L`. Exemplos:

```
?- parenteses([apc,apr,apc,fpc,fpr,fpc,apc,apc,fpc,fpc,apr,fpr],N).
N = 3 ? ;
no

?- parenteses([apc,apr,apc,fpr,fpc,fpc,apc,apc,fpc,fpc,apr,fpr],N).
no

?- parenteses([apc,apr,apc,fpc,fpr,fpc,apc,apc,fpc,fpc,apr],N).
no
```

3. Crie um predicado `functor_args(+L,?FAs)`, usando uma DCG, onde `L` é uma lista de caracteres e `FAs` é um termo da forma `fas(F,LA)`, onde `F` corresponde ao functor e `LA` corresponde à lista de argumentos. Se a lista de argumentos for vazia `FAs` corresponde apenas ao functor. Pode assumir que a lista de caracteres só contém letras minúsculas, vírgulas e parênteses curvos. Exemplos:

```
?- functor_args("x(a,b,yy(cd))",FAs).
FAs = fas(x,[a,b,fas(yy,[cd])]) ? ;
no
```

```
?- functor_args("f",FAs).
FAs = f ? ;
no

?- functor_args("a(b,c,d(g(h(i),j)),k,l)",FAs).
FAs = fas(a,[b,c,fas(d,[fas(g,[fas(h,[i]),j])]),k,l]) ? ;
no
```

4. Crie um predicado `expr_bool(+Expressao,?Resultado)` usando DCG's que reconhece expressões booleanas do seguinte tipo:

   ```
   EXPR: EXPR and EXPR | EXPR or EXPR | true | false
   ```

   `Expressao` é uma lista de átomos que são `true`, `false`, `and` ou `or` e `Resultado` deve ser o resultado da avaliação da expressão (o significado das expressões é o óbvio). Note que deve ser respeitada a prioridade da operações, ou seja, a operação `and` tem prioridade sobre a operação `or`.

   Exemplos:

   ```
   ?- expr_bool([true],R).
   R = true ? ;
   no

   ?- expr_bool([true,and,false],R).
   R = false ? ;
   no

   ?- expr_bool([true,or,false],R).
   R = true ? ;
   no

   ?- expr_bool([false,and,false,or,true],R).
   R = true ? ;
   no

   ?- expr_bool([false,or,true,and,false,or,true,and,false],R).
   R = false ? ;
   no
   ```

5. Extenda o programa criado anteriormente para criar um predicado `expr_bool_if(+Expressao,+Variaveis,?Resultado)` usando DCG's que consegue produzir resultados para expressões do tipo:

```
EXPR: EXPR and EXPR | EXPR or EXPR | true | false
EXPR: ( EXPR ) | var(ATOMO)
EXPR: if ( EXPR ) then { EXPR } else { EXPR }
EXPR: if ( EXPR ) then { EXPR }
```

`Variaveis` é uma lista de tuplos no formato `(Variavel,Valor)` indicando que a variável de nome `Variavel` deve tomar o valor descrito em `Valor`. O significado das expressões é p óbvio. Quando a expressão estiver sintacticamente correcta mas não for possível saber o seu resultado o predicado deve devolver `undefined`. Isto pode acontecer por exemplo quando for preciso o valor de uma variável que não está definida. Veja os exemplos para esclarecer as suas dúvidas.

Exemplos:

```
?- expr_bool_if([false,or,true,and,false,or,true,and,false],[],R).
R = false ? ;
no

?- expr_bool_if([var(a),or,var(b)],[(b,true),a(false)],R).
R = true ?
yes

?- expr_bool_if([var(a),and,var(b)],[(b,true)],R).
R = undefined ?
yes

?- expr_bool_if([var(a),and,var(b)],[(b,false)],R).
R = false ? ;
no

?- expr_bool_if([false,and,'(',false,or,true,')'],[],R).
R = false ? ;
no

?- expr_bool_if([if,'(',true,')',then,'{',var(x),'}'],[(x,true)],R).
R = true ? ;
no
```

```
?- expr_bool_if([if,'(',false,')',then,'{',var(x),'}'],[(x,true)],R).
R = undefined ? ;
no

?- expr_bool_if([if,'(',var(a),or,var(b),')',
                 then,'{',var(c),and,var(d),'}',
                 else,'{',
                    if,'(',var(e),')',
                    then,'{',var(f), or, var(g), and, var(h),'}',
                    else,'{',var(i),'}',
                 '}'],
                [], R).
R = undefined ? ;
no

:- expr_bool_if([if,'(',var(a),or,var(b),')',
                 then,'{',var(c),and,var(d),'}',
                 else,'{',
                    if,'(',var(e),')',
                    then,'{',var(f), or, var(g), and, var(h),'}',
                    else,'{',var(i),'}',
                 '}'],
                [(a,true),(c,false)], R).
R = false ? ;
no

:- expr_bool_if([if,'(',var(a),or,var(b),')',
                 then,'{',var(c),and,var(d),'}',
                 else,'{',
                    if,'(',var(e),')',
                    then,'{',var(f), or, var(g), and, var(h),'}',
                    else,'{',var(i),'}',
                 '}'],
                [(a,false),(b,false),(e,true),
                 (f,false),(g,true),(h,true)], R).
R = true ? ;
no
```