Predicados Dinâmicos

Predicados dinâmicos são predicados que podem ser alterados durante o processamento, por meio dos predicados assert e retract, e seus derivados.

Um predicado dinâmico precisa ser declarado como dinâmico antes de ser usado, mas fora isso, não é necessário ter sequer uma instância do predicado antes do processamento, isto é, ele pode ser declarado **apenas** por assert.

Chamada dynamic

Para usar os predicados dinâmicos, deve-se declará-lo usando o predicado dynamic. A chamada do predicado dynamic é um pouco diferente da de outros predicados, como demonstrado a seguir:

```
:- dynamic([
predicado1/1,
predicado2/0,
predicado3/5,
]).
```

O predicado dynamic leva como argumento uma única lista, que deve ter como membros as identificações dos predicados, na forma de `predicado/aridade`, onde aridade é o número de argumentos do predicado.

Predicado assert

Para adicionar novas instâncias de predicado durante o processamento, se utiliza o predicado assert. Podem ser adicionados tanto fatos como regras, desta forma:

```
pred_dos_assert(bla) :-
    assert(predicado1), % fato com aridade 0
    assert(predicado2(arg1,arg2)), % fato com aridade 2
    assert(predicado3(arg1) :- cond1,cond2). % regra de aridade 1 e duas condições.
```

É importante lembrar que o assert deve ser chamado, ou diretamente pelo usuário(não recomendado), ou dentro de outros predicados de nível mais elevado.

Predicado retract

Para remover instâncias de um predicado durante o processamento, utiliza-se o predicado retract, ou seu derivado mais geral, retractall.

O predicado retract remove a primeira instância de um predicado que aceite os argumentos dados. Se dadas variáveis como argumentos, o retract remove a primeira instância do predicado cujos argumentos dados tenham o mesmo "formato" das variáveis dadas no retract, demonstrado a seguir:

```
pred_dos_retract(bla) :-
    retract(predicado1), % predicado com 0 aridade, remove a primeira instância
    retract(predicado2(A)), % se A já for definido(usando = ou is), remove a instância que a
                            % senão, salva A como primeiro argumento válido para o predicado
    retract(predicado3([A|Lista])). % mesma coisa do anterior, mas com listas
```

Um exemplo de uso inteligente do retract é na contagem e na manipulação de pilha:

```
contador :- % simplesmente aumenta em 1 o valor guardado no fato conta/1
    retract(conta(A)),
    A1 is A + 1,
    assert(conta(A1)).

tiradapilha :- % remove o primeiro membro de uma lista salva no fato pilha/1
    retract(pilha([Lixo|Lista])),
    assert(pilha(Lista)).

addpilha(Novo) :- % adiciona Novo à lista salva no fato pilha/1
    retract(pilha(Lista)),
    assert(pilha([Novo|Lista])).
```

Derivado: retractall

O predicado retractall remove **todos** os predicados que aceitem os argumentos dados, diferentemente do retract, que remove apenas o primeiro. Para limpar completamente todas as instâncias de um predicado, utiliza-se o retractall com argumentos vazios:

```
pred_rall :-
    retractall(predicado(_,_,_)).
```

Isso é uma forma fácil de limpar tudo que foi adicionado durante o processamento para começar um novo ciclo.

Obs: É importante notar que enquanto o swi-prolog não for fechado, todos os asserts e retracts continuam em efeito, portanto, ao começar um novo ciclo de processamento, limpe corretamente seu programa.

That's all, folks