EP 1– Programação em Prolog: Listas, Conjuntos, Gramáticas

Entrega: 06/05/2025

Parte A: Listas e Conjuntos

O objetivo desta primeira parte é acelerar a familiarização dos alunos com conceitos de Programação Lógica e manipulação recursiva de listas, para que todos estejam preparados para a segunda parte do exercício.

Exercício 1

Implementar o predicado `lista_para_conjunto(Xs, Cs)` em que `Cs` é uma lista que contém os mesmos elementos de `Xs`, na mesma ordem de sua primeira ocorrência, mas cujo número de ocorrências é apenas 1. Por exemplo, a consulta

```
?- lista_para_conjunto([1,a,3,3,a,1,4], Cs).
```

sucede apenas para `Cs=[1,a,3,4]`. A consulta

```
?- lista_para_conjunto([1,a,3,3,a,1,4], [a,1,3,4])
```

deve falhar.

A lista `Cs` do enunciado anterior será chamada de *lista-conjunto*, ou seja, uma lista com apenas uma ocorrência de cada elemento.

Exercício 2

Implementar o predicado `mesmo_conjunto(Cs,Ds)` que sucede quando `Cs` e `Ds` são listas conjunto que representam os mesmo conjunto, ou seja, possuem os mesmos elementos ocorrendo uma única vez, em qualquer ordem. A consulta

```
?- lista_para_conjunto([1,a,3,4], Cs)
```

deve suceder quando `Cs` é uma lista que contem qualquer permutação dos elementos `a,1,3,4`.

Exercício 3

Implementar o predicado `uniao_conjunto(Cs,Ds,Es)` em que `Cs, Ds, Es` são listas-conjuntos em que o terceiro representa a união dos dois primeiros.

Exercício 4

Implementar o predicado `inter_conjunto(Cs,Ds,Es)` em que `Cs, Ds, Es` são listas-conjuntos em que o terceiro representa a intersecção dos dois primeiros.

Exercício 5

Implementar o predicado `diferenca_conjunto(Cs,Ds,Es)` em que `Cs, Ds, Es` são listas-conjuntos em que o terceiro representa o conjunto diferença do primeiro, excluindo-se os membros do segundo.

Parte B: Gramáticas do Português Contemporâneo

O objetivo dessa segunda parte do trabalho é desenvolver uma gramática para uma variante do português falado em São Paulo. Em particular vamos desenvolver uma gramática com as seguintes características:

- Nomes são invariantes para número. Por exemplo, a palavra *cara* pode denotar uma única pessoa ou um conjunto de pessoas, de qualquer sexo. No entanto, nomes possuem gênero, e no caso anterior, o gênero da palavra *cara* é masculino.
- Os determinantes (artigos) possuem número e gênero. Por exemplo, *a* é um determinante feminino singular, *os* é um determinante masculino plural.
- Um sintagma nominal pode possuir um determinante, e o número do determinante será o número do sintagma nominal. Além disso, os determinantes devem concordar como o gênero do nome. Por exemplo, *a mina* é um sintagma nominal feminino singular, e *os cara* é um sintagma nominal masculino plural.
- Um sintagma nominal pode ser formado apenas por um nome, em cujo caso o sintagma pode ser considerado tanto singular como plural. Por exemplo, *mina* pode ser um sintagma nominal singular ou plural.
- Verbos na terceira pessoa são sempre invariantes; tanto no caso de verbos intransitivos ou transitivos.

Traços

A forma de lidar com questões de concordância, tanto de gênero quanto de número, em gramáticas de estrutura frasal, é extender o conceito de Categoria (sintática ou morfossintática), com informações chamadas de *traços*.

No caso, extenderemos os elementos de nossa gramática com traços de gênero e/ou número. Assim, os nomes, que na gramática do exercício são invariantes de número, podem ser acrescidos com um traço de gênero:

```
nome( n( N, Genero ) ) --> [N],{nome(N, Genero)}.

nome( cara, m ).
nome( mina, f ).
nome( gente, _ ). % gênero neutro
```

Similarmente, os determinantes são extendidos com traços de gênero e número:

```
determinante( d( X, Gen, Num ) ) --> [X], {det(X,Gen,Num)}.

det( a, f, s ).
det( o, m, s ).
det( as, f, p ).
det( os, m, p ).
```

Na hora de formar os sintagmas nominais, essas informações devem ser unificadas, para garantir a relação adequada de concordância. Inclusive, pode ser necessário atribuir traços para os próprios sintagmas, que seriam derivados dos traços de seus componentes, se eles forem usados em níveis de concordância entre sintagmas; por exemplo, quando há concordância de número entre sujeito (sintagma nominal) e predicado (sintagma verbal).

Implementação

Pede-se implementaro o predicado `gramatical(Sentenca)` usando DCGs, tal que a `Sentença` é uma lista de palavras; aconselha-se também usar um predicado auxiliar, em formato DCG, `sentenca(Arvore)`, onde o argumento é um termo-Prolog representando o parseamento (análise sintática) da Sentença, levando em consideração os traços de gênero e número.

Seu predicado deve reconhecer as seguintes sentenças:

1. *As mina encarô os cara*
2. *O cara encarô a mina*
3. *O cara encarô as mina*

Seu predicado deve rejeitar as seguintes sentenças:

1. *Os caras encararam as minas*
2. *Os mina encarô os cara*
3. *A mina encarô as cara*

Instruções para entrega

Você deve submeter via eDisciplinas um zip[1] contendo dois arquivos, chamados chamado de `ep1A.pro` e `ep1B.pro`. Uma versão preliminar de cada um destes arquivos é fornecida com este enunciado. Você deve entregar os arquivos editados, contendo a sua solução. Sua solução pode conter os predicados auxiliares que julgar necessário. Para evitar que seu EP seja zerado, certifique-se que cada arquivo foi submetido sem problemas (baixando e executando o arquivo do site) e que eles consistem de um programa executável escrito em SWI Prolog com o comando:

```
?-run_tests.
```

Você está recebendo junto com este enunciado dois arquivos chamados de `ep1A.pro` e `ep1B.pro`, obtendo uma série de testes nos quais seu programa deve passar. Você deverá escrever nestes mesmos arquivos as suas respostas aos itens deste enunciado.

[1] Não entregue arquivos compactados no formato RAR!!! Aceitamos zip ou tgz, apenas.