Universidade Federal de Uberlândia
Faculdade de Computação
Programação Lógica (GSI010) — Prof. Dr. Paulo H. R. Gabriel

Correção da Segunda Avaliação

Instruções:

- A seguir, são apresentadas as duas questões da segunda avaliação que exigiam implementação.
- É apresentada, também, uma questão extra.
- Resolva essas questões utilizando o SWISH.
- Em seguida, crie um arquivo do tipo texto (no Bloco de Notas, por exemplo), e copie suas respostas nele. Salvo-o com a extensão TXT e encaminhe pelo Moodle.

Prazo final para entrega: segunda-feira, dia 10/06.

1. (a) Escreva uma regra para o predicado `entre(X, A, B)`, que indica que o número `X` está entre os números `A` e `B`. Assuma que `A` é menor ou igual a `B`.

   (b) Reescreva a regra do item anterior de forma que funcione para quaisquer dois valores inteiros de `A` e `B`, seja `A` maior ou menor do que `B`.

2. Considere o modelo lógico de Pokémons descrito a seguir. Cada Pokémon é de um tipo (fogo, gelo, água, dentre outros), como exemplificado pelo predicado `tipo(charmander, fogo)`. Alguns Pokémons podem evoluir para outros Pokemóns, como indicado pelo predicado

   `evoluiPara(charmander, charmeleon)`.

   As evoluções são sempre do mesmo tipo do Pokémon original. Considerando uma base de conhecimento que só informe o tipo dos Pokémons que não são evolução de outros Pokémons, defina o predicado `tipoTotal(P, T)`, que indica que o Pokémon `P` é do tipo `T`, considerando os tipos e as evoluções da base de conhecimento. Crie predicados auxiliares se necessário.

   Por exemplo, considere a base de conhecimento a seguir:

   ```
   tipo(charmander, fogo).
   tipo(jynx, gelo).
   evoluiPara(charmander, charmeleon).
   evoluiPara(charmeleon, charizard).
   ```

   Nesse caso, a consulta `tipoTotal(charizard, T)` retorna `T = fogo`, pois o `charizard` é uma evolução do `charmander`, que é do tipo `fogo`.

   Dica: Modele o problema como se fosse uma árvore genealógica, em que os pais são os Pokémons originais e os filhos são suas evoluções.

3. Considere a seguinte base de conhecimento:

```
deCarro(auckland, hamilton).
deCarro(hamilton, raglan).
deCarro(valmont, saarbruecken).
deCarro(valmont, metz).

deTrem(metz, frankfurt).
deTrem(saarbruecken, frankfurt).
deTrem(metz, paris).
deTrem(saarbruecken, paris).

deAviao(frankfurt, bangkok).
deAviao(frankfurt, singapore).
deAviao(paris, losAngeles).
deAviao(bangkok, auckland).
deAviao(losAngeles, auckland).
```

Escreva um predicado `viagem/2` que determina se é possível viajar de um lugar para outro combinando trajetos de carro, trem e avião. Por exemplo, seu programa deve responder `true` à consulta `viagem(valmont, raglan)`.