Aula de laboratório: lógica proposicional

Prof. Anderson Nakano

Lembremos qual é o nosso projeto: desenvolver, em Python, um *prompt* de comando em que se pode inserir frases, formuladas numa sintaxe próxima da linguagem natural (que definiremos em momento oportuno), que irão compor uma "base de crenças" de uma pessoa. Para cada nova inserção p, deve-se verificar

1. se p é uma tautologia ou contradição; se esse for o caso, não adicionar p na base de dados e imprimir "p não possui conteúdo informacional".
2. se p é uma consequência dedutiva de outras sentenças já inseridas; se esse for o caso, não adicionar p na base de dados e imprimir "informação redundante";
3. se a adição de p faz a base de dados ficar inconsistente; nesse caso, não adicionar p na base de dados e imprimir "informação conflitante com a base de crenças" (opcional: deixar o usuário escolher entre desistir de adicionar p ou remover outras frases já inseridas até que a inserção de p não torne mais a base de crenças inconsistente).

Especificação da sintaxe de entrada de dados

A sintaxe da entrada de dados que escolheremos, em um primeiro momento, será muito próxima à da linguagem da lógica proposicional; exceto que, ao invés de P, Q, R, etc., (i.e., dos símbolos usados para frases atômicas) utilizaremos frases do português. A título de exemplo, as frases ' "(a gasolina acabou $\vee$ o motor fundiu)", "(a gasolina não acabou $\supset$ o motor fundiu)" e "$\neg$ a gasolina acabou" são frases bem formadas para a entrada de dados de nosso *prompt*.

Tarefa 1: defina uma função `is_wff(input)` para determinar se uma frase da lógica proposicional é bem formada ou não.

Tarefa 2: defina uma função `get_tree(input)` que retorna a árvore sintática de uma frase bem formada. Para isso, construa uma classe `Node` que representa um nó da árvore. A classe nó deve conter um campo para guardar uma informação (que será um conectivo ou uma frase atômica) e uma lista de nós-filho.

Tarefa 3: implemente a técnica de tableaux semânticos ensinada em sala de aula.

Tarefa 4: defina uma função `is_tautology(formula)` que retorna `True` se *formula* é uma tautologia, e `False` caso contrário. Use, para isso, a técnica de tableaux semânticos.

Tarefa 5: defina uma função `is_contradiction(formula)` que retorna `True` se *formula* é uma contradição, e `False` caso contrário. Use, para isso, a técnica de tableaux semânticos.

Tarefa 6: defina uma função `follows_from(conclusion, premisses)` que retorna `True` se `conclusion` é uma consequência de `premisses`, e `False` caso contrário. Use, para isso, a técnica de tableaux semânticos.

Retornando ao nosso prompt...

Tarefa 7: defina uma função `get_sentences_with_atoms(beliefs, atoms)` que retorna uma lista dos `beliefs` que contêm ao menos um átomo proposicional de `atoms`.

Tarefa 8: use todas as funções definidas anteriormente para definir a função `main` de seu programa.