Trabalho prático individual nº 2

Inteligência Artificial
Ano Lectivo de 2023/2024

8 de Dezembro de 2023

I Observações importantes

1. This assignment should be submitted via *GitHub* within 28 hours after its publication. The assignment can be submitted after 28 hours, but will be penalized at 5% for each additional hour.

2. Complete the requested methods in module `"tpi2.py"`, provided together with this description. Keep in mind that the language adopted in this course is Python3.

3. Include your name and number and comment or delete non-relevant code (e.g. test cases, print statements); submit only the mentioned module `"tpi2.py"`.

4. You can discuss this assignment with colleagues, but you cannot copy their programs neither in whole nor in part. Limit these discussions to the general understanding of the problem and avoid detailed discussions about implementation.

5. Include a comment with the names and numbers of the colleagues with whom you discussed this assignment. If you turn to other sources, identify these sources as well.

6. All submitted code must be original; although trusting that most students will do this, a plagiarism detection tool will be used. Students involved in plagiarism will have their submissions canceled.

7. The submitted programs will be evaluated taking into account: performance; style; and originality / evidence of independent work. Performance is mainly evaluated concerning correctness and completeness, although efficiency may also be taken into acount. Performance is evaluated through automatic testing. If necessary, the submitted modules will be analyzed by the teacher in order to appropriately credit the student's work.

II Exercícios

Together with this description, you can find modules semantic_network and constraintsearch . They are similar to the ones used in practical lectures, but some changes and additions were introduced.

Module `tpi2` contains some derived classes. In the following exercices, you are asked to complete certain methods in these classes. Any other code that you develop and integrate in other modules will be ignored.

The module `tpi2_tests` contains some test code. You can add other test code in this module. Don't change the semantic_network and constraintsearch modules.

You can find the intended results of `tpi2_tests` in the file `tpi2_results.txt`

The responses to the main questions asked by students during this TPI will be collected in section III below.

1. The attached module `semantic_network` was designed based on the one you already know, from practical classes, but has some changes were introduced:

   - All network data is know in the form of a dictionary of dictionaries (see code).
   - The following convention was introduced: object names start with upper case letter (e.g. 'Mary') and type names start with lower case letter (e.g. 'woman'). Convenience functions are provided to check if a name is an object or type.
   - There is a new type of association, `AssocOne`, which allows only one value (e.g. hasMother: each person has only one mother).

   You are expected to implement the following methods in class `MySN`

   a) `query_local(user,e1,rel,e2)` - Equivalent to the method with the same name that you know from practical classes.

   b) `query(entity,assoc)` - Equivalent to the method with the same name that you developed in practical classes.

   c) `update_assoc_stats(assoc,user=None)` - This method stores and/or updates frequency statistics in a dictionary where keys are tuples `(assoc,user)` and values are tuples `(stats_assoc_e1,stats_assoc_e1)`, i.e. frequency statistics for the types of the first and second arguments of the association `assoc`. If `user==None`, the statistics are computed based on the declarations of all users. Otherwise, the statistics are computed based on the declarations of the specified user. For each of the arguments, the method will collect all objects used in that argument, in the available declarations, check their types, compute frequencies of occurrence of each type, and propagate the information upwards in the type hierarchy. For `N` such objects, where `K` of them are of unknown type, the total number of objects to be used for calculating relative frequencies is given by `N-K+K**(1/2)`. See the results file for examples and details.

2. Implement in class `MyCS` the method `search_all()`. Instead of a normal `search()` method, this one will return all possible solutions without repetitions. This exercice will be evaluated based on correctness, completeness and time efficiency. In order to improve efficiency, consider including constraint propagation and some form of variable ordering.

III Clarification of doubts

This work will be followed through `http://detiuaveiro.slack.com`. The clarification of the main doubts will be placed here.

1. ...