# Trabalho prático individual nº 2

# Inteligência Artificial
# Ano Lectivo de 2024/2025

13 de Dezembro de 2024

## 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, bayes_net and constraintsearch . They are similar to the ones used in practical lectures, but some changes and additions were introduced.

The attached module `semantic_network` contains new types of relations, namely `AssocSome`, `AssocOne` and `AssoNum`. `AssocSome` is a normal association with no cardinality restrictions. In turn, `AssocOne` allows only one value (e.g. in `hasFather`, each person has only one father).

In module `bayes_net`, mother variables are split into true and false lists.

In module `constraintsearch`, constraint propagation is given implemented.

Module `tpi2` contains some derived classes. In the following exercices, you are asked to complete certain methods/functions in this module. 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, bayes_net 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.

You are expected to implement the following methods:

1. Implement the method `query(entity,relname)` in class `MySN`, which returns a list of values for a given relation in a given entity, possibly including inherited values. The different types of relations are handled in different ways. In which concerns `Member` and `Subtype` relations, only the local ones are relevant. Inheritance of `AssocOne` and `AssocNum` is processed with cancelling, i.e. the existance of one of these associations in a given entity cancels the effects of a similar association with the same name in a predecessor entity. When there are several declarations of an `AssocOne` association, the most common value should be returned. When there are several declarations of an `AssocNum` association, the average of all values should be returned. In the case of `AssocSome`, inheritance is processed without cancelling. Therefore, all such associations found in predecessors are relevant. Finally, because different users may use a given association name for different types of associations, the type of association most frequently used with a given name should be considered the correct type of the association. For example, `hasHeight` was used four times as `AssocNum` and only once as `AssocOne`, therefore the type of this association is considered to be `AssocNum` and the only declaration in which it was used as `AssocOne` is ignored.

2. Implement the method test_independence(v1,v2,given) in class `MyBN`, which determines if two variables, `v1` and `v2`, in a Bayesian network, are conditionally independent of each other when some other variables are given. In order to test independence, the method builds an undirected graph containing the following edges:

   - All edges in the network connecting the input variables (i.e. `v1`, `v2` and those in `given`) to all their ancestors.
   - For each variable covered in the previous point, add edges connecting all pairs of mother variables.
   - As a last step, remove all edges containing any of the variables in `given`.

   The graph is represented as a list of pairs of variables. Variables `v1` and `v2` are independent if there is a path connecting them in the graph. The method receives as input the two variables and a list of other (given) variables, and returns as result a boolean (`True` if independent, false otherwise) and the used graph. The network represented in Figure **??** is included in the `tpi2_tests` module for testing purposes.

3. Implement in class `MyCS` the method `search_all()` which returns all possible solutions without repetitions. This exercice will be evaluated based on correctness, completeness

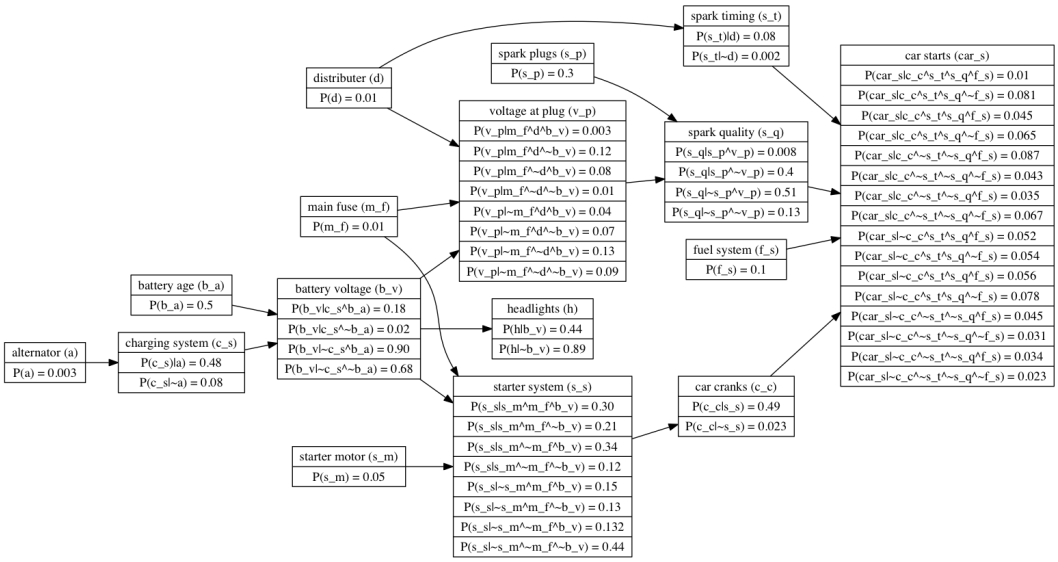


Figura 1: Bayesian network for a car that does not start

and time efficiency. In order to improve efficiency, consider including some form of variable ordering.

4. Develop the function handle_ho_constraint (domains, constraints , variables , constraint ) which, given the `domains` of variables in a problem, the binary `constraints` between these variables, the `variables` involved in a higher-order constraint and a unary `constraint` (which represents the higher-order constraint as presented in classes), will add appropriate auxiliary variable(s) and binary constraint(s) in the constraint graph.

# 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. Podemos adicionar argumentos às funções?

   **Resposta**: Only optional arguments located after the argments already included; for testing, I will call the functions with the mandatory arguments that are identified in the provided code, and no others.