Department of Applied Mathematics and Computer Science

02180 Introduction to Artificial Intelligence

Belief Revision Assignment

Authors:

Dario Cannistrà 194830

Stella Maria Saro 196563

Konstantinos Spyrikos 200240

Ghassen Lassoued 196609

Federico Crespo 200161

Course responsible:
Nina Gierasimczuk

May 17, 2020

1 Introduction

This paper is concerned with the developing of AGM belief revision agent. Belief revision is one of the main topic in artificial intelligence. Since the beginning of computing, the updating of the belief base with new information is a long discussed argument. Thanks to the use of the AGM belief revision it is had been making sure that our belief base is consistent when a new sentence is added. The algorithm that has been used operates according to the Levi identity.

$$\mathbf{K} * \mathbf{p} = (\mathbf{K} \div \neg \mathbf{p}) + \mathbf{p} \tag{1}$$

In the next chapter, the features and the implementation of the agent will be discussed , which are mainly divided into three steps: revision (*), contraction ($\div$) and expansion (+).

The AGM belief revision agent was implemented by using Python as the programming language. More specifically, the SymPy library was used. It is very convenient when dealing with logical propositions because thanks to its built- functions it allows you to operate with Boolean expressions just with standard python operators e.g. & (And), | (Or), (Not)

2 Design and Implementation of belief base

First of all, it is essential to mention how every belief state is been modeled. Beliefs are stored in a class that we have created called 'Belief', inside this class, the SymPy built-in function 'to_cnf' is used to assure that every belief being inputted is stored in CNF form. The data structure used for the belief base is a list of elements of the class belief. In order to assure that the Belief Base remains consistent, every time we add a new belief we perform the Belief Revision algorithm. During this algorithm we check that the sentence being added is consistent, then if it is already entailed (via our resolution based entailment algorithm) in the Belief Base (since if it this were to be the case, we would not need to add it as it would be redundant), and if the new belief being added makes sense with every other belief in the Belief Base. This last aspect is implemented through a selection function that will discard those sentences that do not have sense. This aspect will be discussed later in the paper. Computationally speaking , the elements of the belief base are formulas expressed in propositional logic and not in natural language.

The design of the belief base has been performed in the AGM model, in that way every sentence that follows from the belief set is already in it.

Therefore, the belief base consists of sentences reduced to the CNF form and it will be *input assimilating* and *action priority update* i.e. our belief will change according to the new input sentences and the new incoming information will have more priority than my belief.
The belief base is being printed as a list of single sentences revised every time a new information is added.

3 AGM Belief Revision

The AGM revision is one of the possible way to revision a sentence it is named after the names of their proponents, Alchourrón, Gärdenfors, and Makinson. This method is different from the others because the revision function is considered 'appropriate'. With appropriate it is meant that it satisfies the six rational postulates and for this the rationality is granted. To follow an image with the pipeline that the algorithm ensues from and the description of the various step

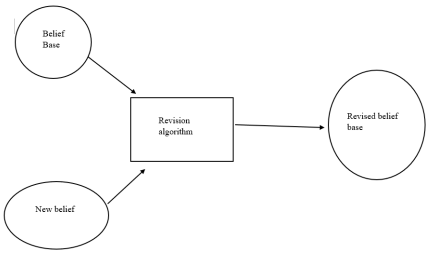

Figure 1: Naive AGM simplification

In the code, it has been chosen to deal with an external revision. So, the argument is first added and then the contraction is executed.

The algorithm used to implement this, is based on the AGM revision. In the algorithm, after having defined a class of Belief, the user can start inputting the sentences into the knowledge. The Belief base will obviously be empty at the beginning and then, after having initialized it, it is possible to add a new sentence. At the end the user can see how the list will be updated and after some steps it will also be possible to empty the list.

3.1 Logical Entailment

The entailment is going to be derived using Propositional Resolution. The entailment is being verified before the addition of the new sentence, inside the belief base. In fact, if the information in the new belief is already entailed in the belief base, it does not need to be added. The basic principle of resolution is that, given a clause containing a literal a and another clause containing the literal $Not(a)$, the clause containing all (and only) the other literals can be inferred by the previous two.

This needs to be done considering only one pair of complementary clauses at a time, in order to avoid an incorrect evaluation of the inconsistency that would give an empty clause. If the derivation produces an empty clause this implies that the function for which we are checking the entailment is in fact entailed in the Belief Base. If this is the case, the new belief does not need to be added to the Belief Base, if not it has to be added.

For a better understanding, consider the following example:
Consider that you have a clause composed of a,b and the clause $\neg a$. From these two 'a' can be derived. The final database is not empty and from the Deduction Semantic Theorem is possible to state that '$\neg a$' is not entailed in the database because it is not returning the empty clause.

In order to implement the Propositional Resolution for a new sentence, after decomposing the belief base into clauses, a conjunction of all of them is being performed. The negation of the new

sentence is then been added to the conjunction.

At this point the clauses have been resolved into pairs. For every two complementary elements in a couple of clauses, a new clause containing all the literals of the pair, except the complementary ones, has been created. If a newly created clause is empty, then if in the belief base we would include the negation of the new belief it would be inconsistent this means that in this belief base the negation of the new sentence is unsatisfiable. The resolution is then complete and the entailment is verified, if the generated clause is not empty then the clause is added to the conjunction.

If the new clause has elements that are already present in the clauses, those elements are deleted in order to minimize the number of pairs to resolve. In fact, these clauses would not contain any new useful information and would just be a repetition.

3.2 Revision and Order

The revision is the core of the project. In order to duplicate the human belief, the agent has to adapt to the new input sentences. The main purpose of the agent is to make the beliefs consistent and to modify or discard all the sentences that make it inconsistent.

It is important to point out the difference between knowledge and belief. The knowledge is what is known to be true, it is an axiom; hence, it can not be inconsistent. Regarding the latter, it is something that the agent thinks to be True. In our case the new belief is always considered with a greater priority over the others, because it is assumed to be deriving from a reliable source. Thus the Belief base prior to the addition of this last belief has to be modified following the Levi identity 1.

Once a new belief is added, in order to implement the revision of the belief base, for each belief in it, it has been verified if the conjunction with the new belief is satisfied. If the new belief is in contradiction with one of the other beliefs this would be removed from the belief base. After doing this a new belief base is been created in which the last belief is consistent with each of the others but it is still possible that the conjunction of all the beliefs is unsatisfiable. At this point something needs to be deleted to make the revised belief base consistent.The choice on what to delete from the belief base has been made based on the plausibility order.

The order has been defined as the sum of all the true sentences into the belief base. First it has been calculated a list of all the possible combinations obtained by associating the Values True and False to the different literals present in the Belief Base. Then the order has been calculated for every one of these combinations substituting the corresponding values of the literals in everyone of the sentences of the belief base and storing the value assumed by the function and counting the True values.

The combination with maximum order for which the last added belief is True, has then been selected and the corresponding values of the literals have been substituted in the beliefs. Given these values, the beliefs that resulted unsatisfiable have been removed.

In case of more combinations corresponding to the maximum order, these would both be acceptable but a choice has to be made. The choice made in this implementation is keeping the first combination selected by the loop.

To make it more clear, let's consider an example like the one discussed in slide 10. As you can see there could have been two plausible outputs and both of them would be acceptable.
How can decide if validate one choice over another?

The goal in this case is to discard the minimum amount of information and a general solution to the problem is the partial contraction that operates on remainders which maximizes the number of propositions retained. It iterates through a selection function that indicates which statements need to be selected by the program inside the remainder set. The selection cannot be rational in this case because the human mind is very complex and is not possible to predict a rational order to decide which sentence is more plausible over the other.

Priority order on formulas in the belief base:
The idea behind the plausibility order is that the most plausible world is the one with the highest number of true propositional sentences in it. Taking into account that the update of belief base is very credible, this update has to be in the world.
An example is given:
Fede' s belief: a, b, a $\rightarrow b$
$He learns from Kostas that: \neg b$
After revision

World	$\neg a$,$\neg b$	$\neg a$,b	a,$\neg b$	a,b
Plausibility (number of truths):	2	1	2	2

We are looking for the most plausible world that does not satisfy b.
A possible answer, after revision is: Fede' s belief base: a, $\neg b$

4 Further improvements

The belief base is implemented in propositional logic language using CNF sentences. First-order logic is a more powerful language. Hence, possible optimization would be to implement the knowledge base in first-order logic to examine relations between objects. In general, someone could consider the possibility of implementing the belief base with a natural language that is more powerful than propositional logic.

Another possible improvement could be being able to read sentences directly from natural language, but in order to do so we would need a natural language interpreter that could derive propositions from it.

Even though lists have been used, we consider that sets could have been more beneficial both computationally and logically since subsets and uniqueness are part of their definition as a data structure.

5 Conclusion

This project gave us the opportunity to enrich our knowledge regarding belief base, entailment, revision, resolution and the SymPy package. Using python as our programming language made us also

discover some new interesting aspects regarding its capabilities and functionality. All things considered, it is possible to apply the methods encountered to a specific domain such as Mastermind game. It has been difficult to understand how to implement the algorithms seen in the lectures, maybe because of the lack of experience working with sets which in this kind of problems would have been more efficient.

References

[1] Nina Gierasimczuk's lecture notes

[2] Stuart Russell Peter Norvig Artificial Intelligence - A Modern Approach. 3rd edition, Global Edition.

[3] Propositional Resolution, http://intrologic.stanford.edu/notes/chapter_05.html

[4] SymPy documentation, https://www.sympy.org/en/index.html