Non Prioritized Belief Revision with AnsProlog*

Gerardo I. Simari [1] Marcelo A. Falappa

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA) [2]
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur. Av. Alem 1253, (8000) Bahía Blanca, Argentina
Tel: ++54 291 4595135 - Fax: ++54 291 4595136
{gis,mfalappa}@cs.uns.edu.ar

1 Introduction and Background

One of the key aspects in the design of an architecture for autonomous agents is the way in which beleifs are revised in the face of new information. An intelligent agent must be prepared to handle new information (be it sensory input, communication with other agents, etc). The areas of *Knowledge Representation and Reasoning* and *Belief Revision* deal with the issues of how beliefs are represented, how they are used in reasoning processes, and how a current body of beliefs reacts to the appearance of new information. There are many problems that must be solved in order to effectively manage a body of knowledge which, more often than not, may be incomplete or even inconsistent. The adequate solution to these problems is an essential feature of autonomy, because an agent must always be up to date in order to behave accordingly [12].

Belief Revision was mainly introduced by Gärdenfors and later extended by Alchourrón, Gärdenfors, and Makinson [1, 4]. Its main goal is to model the dynamics of knowledge, *i.e.*, the way in which an agent's knowledge reacts before new information. Daily life offers many examples in which humans perform belief revision: watching the stock quotes on TV may inform us that a certain stock is doing well, or a phone call from an old friend may tell us that he is in town for the holidays, and that he is now married. In this example, the basic operations of belief revision, known as *expansion, contraction*, and *revision*, are mapped respectively into finding out that the stock is doing well, that our friend is no longer out of town, and that he is now married. These operations are also related to environments closer to the computational realm, such as the use of communication protocols where new protocols can be added (expansion), withdrawn (contraction), or a subset of them be replaced by one or more (revision).

Revisions are the most commonly used change operators because they allow a sentence α to be included into a set K, generating a new set K', preserving consistency in the new set. The traditional revision models are *prioritized*, that is, they give priority to new information over the information that is already part of the knowledge. This property does not seem plausible

This work is partially supported by the *Agencia Nacional de Promoción Científica y Tecnológica* (PICT 13096)

[1]Partially supported by *Comisión de Investigaciones Científicas, Gobierno de la Provincia de Buenos Aires.*

[2]Member of the IICyTI (Instituto de Investigación en Ciencia y Tecnología Informática).

in the real world, because in many cases it is not reasonable to give priority to information just because it is new.

In *non prioritized* models, it is possible for new information *not* to be totally accepted. Such new information can be rejected or accepted only after a debate process. In this sense, there exists a variety of different non prioritized belief revision models, among which are David Makinson's Screened Revision [11, 10], Sven Ove Hansson's semirevision operators [9], André Fuhrmann's merge operations [3] and Falappa *et al.*'s [2] recently formulated revisions by sets of sentences. In this last work, a new kind of non prioritized revision operator based on the use of explanations is presented. It is argued that an agent, before incorporating information which is inconsistent with its knowledge, should request an explanation that supports this information. An explanation is characterized as being composed of an *explanans* (set of sentences supporting a certain belief), and an *explanandum* (the final conclusion).

An example of this situation is the following: Suppose a person believes that (α) *all felines can climb* and (β) *Moebius is a feline*. Thus, he will believe (δ) *Moebius can climb*. Later on, another person states that *Moebius cannot climb*. If δ is dropped, then α or β will have to be dropped as well. However, it does not seem rational to incorporate external beliefs without pondering them. An explanation should be required in order to incorporate such information, especially in the case in which it contradicts the previously maintained set of beliefs.

In our case, the person should demand an explanation for $\neg\delta$. One possibility is that he is given an explanation such as: *Moebius is a cat, but he cannot climb because he is hurt*, and *cats that are hurt cannot climb*. Now, the sentences in the explanans can be used to evaluate the new piece of information before it is incorporated. In [12], we proposed the use of AnsProlog* as a language for the representation of beliefs, as well as for explanations.

2 Belief Revision with respect to Explanations

An agent's beliefs can be represented by means of *belief sets* (sets of sentences closed under logical consequence) or *belief bases* (arbitrary sets of sentences). In computational applications, one must opt for finite belief bases. New information, which is called *epistemic input*, is sometimes represented by a sentence of the language or an arbitrary set of sentences. The revision operator proposed in [2] allows a non prioritized revision in the following way:

- The epistemic input is a single sentence with an explanation for it.
- The explanation (a set of sentences with some constraints) is added to the original set, maybe resulting in a temporarily inconsistent set.
- The consistency is then restored by a contraction by falsum.

It should be noted that this operator incorporates the possibility of *partial acceptance*, *i.e.*, even though the proposed explanation for α may be rejected, parts of it may still be added to the knowledge base in the process.

An *explanation* can be defined as follows [2]. The set A is an explanation for the sentence α if, and only if, the following conditions are satisfied,

1. Deduction: $A \vdash \alpha$.
2. Consistency: A $\nvdash \bot$
3. Minimality: if $B \subset A$ then $B \nvdash \alpha$.
4. Informational Content: $Cn(A) \not\subseteq Cn(\alpha)$.

Deduction guarantees that the explanans (support for a given belief) implies the explanandum (the belief being explained). Consistency prevents having an inconsistent explanation (which would explain *any* belief). Minimality establishes that every belief in the explanation is needed to obtain the explanandum, and Informational Content avoids cases in which the explanandum implies every sentence in the explanans. In particular, it avoids a sentence being an explanation for itself. As we can see from this description, and explanation can be seen as an external *argument* supporting a given belief.

Assume we want to revise a given belief base Π with respect to a given explanation A for a belief α. The revision involves:

1. Construction of counter-explanations for A from Π. These counter-explanations are minimal subsets of Π which are inconsistent with A.
2. A is compared with respect to its counter-explanations.
3. If A is (in some sense) "better" than its counter-explanations, then A is incorporated into Π and its counter-explanations (or part of them) are eliminated. In any other case, Π will remain unaltered.

The last step of the revision involves a decision between the proposed explanation, and the counter-explanations that can be built from Π. This decision will depend upon the particular (typically partial) ordering that is imposed over the set of beliefs.

3 AnsProlog* and the Answer Set Semantics

We will now briefly introduce the language of logic programming with respect to the answer set semantics, which is usually referred to as AnsProlog*. This name is short for "**Pro**gramming in **Log**ic with **Ans**wer sets" [6]. An AnsProlog* program is a finite collection of rules of the form:

L_0 `or` ... `or` $L_k \leftarrow L_{k+1}$, ..., L_m, `not` L_{m+1}, ..., `not` L_n.

where the L_i's are literals (in the sense of classical logic). Intuitively, a rule of this form means that if L_{k+1}, ..., L_m are true and if L_{m+1}, ..., L_n can be assumed to be false, then the sentence L_0 `or` ... `or` L_k is true (*i.e.*, at least one of its literals is true). The symbol '*' in AnsProlog* means that no restriction is applied to the structure of the program's rules; a variety of syntactic sub-classes can be defined when such rules are restricted. The importance of defining a set of sub-classes lies in the varying degree of complexity of the rules in each class. This complexity has a profound impact on the computational cost of operations that may be performed on programs, such as Answer Set checking and verifying if a given belief is entailed by a given program [7]. Part of our work lies in the study of how the computational complexity of these and other problems is related to belief dynamics under this representation.

An AnsProlog* program can be used to represent an agent's beliefs by means of its *answer set semantics*. Such semantics can be defined as follows [5]:

1. A program Π *cautiously entails* a literal L ($\Pi \models L$) if L belongs to *all* answer sets of Π.
2. A program Π *bravely entails* a literal L ($\Pi \models_b L$) if L belongs to *some* answer sets of Π.

Where *answer sets* are minimal and consistent sets of literals that obey the rules of Π. It should be noted that, for programs having only *one* answer set, there is no difference between these two relations.

In the same way, any *explanation* for a given belief can be represented by an AnsProlog program; a subset of the program's rules can be interpreted as the explanans, and the desired explanandum will be entailed by the program.

An example of a belief base is the following program Π_1:

$$climbs(X) \leftarrow feline(X).$$
$$feline(X) \leftarrow cat(X).$$
$$cat(moebius) \leftarrow .$$

Π_1 has as its only answer set:

$$S_1 = \{feline(moebius), cat(moebius), climbs(moebius)\}$$

An explanation for $climbs(moebius)$ represented by an AnsProlog* program, related to this example, is the following program Ψ_1:

$$climbs(X) \leftarrow feline(X).$$
$$feline(fluffy) \leftarrow .$$

This program explains that *Fluffy* climbs because he is a feline, and has the sole answer set: $E_1 = \{cat(fluffy), climbs(fluffy)\}$. This program can then be used as an explanation for $climbs(fluffy)$, where Ψ_1 is the explanans and $climbs(fluffy)$ is the explanandum. An example of an explanation that conflicts with Π_1 is the following program Ψ_2:

$\neg climbs(X) \leftarrow hurt(X).$
$hurt(moebius) \leftarrow .$

which has as its only answer set: $E_2 = \{hurt(moebius), \neg climbs(moebius)\}$. This answer set clearly disagrees with Π_1, which entails the belief $climbs(moebius)$.

4 Research Goals

Our work involves research with the goal of defining revision operators based on this framework. This includes defining *preference criteria* that allows an agent to decide between a proposed explanation and a counter-explanation that can be obtained from its own knowledge. Furthermore, we are also studying how *kernel contractions* [8] can be constructed using this framework, in order to eliminate inconsistencies that arise due to the acceptance of explanations that contradict prior knowledge.

References

[1] Alchourrón, C., Gärdenfors, P., and Makinson, D. *On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. The Journal of Symbolic Logic 50* (1985), 510–530.

[2] Falappa, M. A., Kern-Isberner, G., and Simari, G. R. *Belief Revision, Explanations and Defeasible Reasoning. Artificial Intelligence Journal 141* (2002), 1–28.

[3] Fuhrmann, A. *An Essay on Contraction.* Studies in Logic, Language and Information, CSLI Publications, Stanford, California, 1997.

[4] Gärdenfors, P. *Knowledge in Flux: Modelling the Dynamics of Epistemic States.* The MIT Press, Bradford Books, Cambridge, Massachusetts, 1988.

[5] Gelfond, M., and Leone, N. Logic programming and knowledge representation—the A-prolog perspective. *Artificial Intelligence 138*, 1–2 (2002), 3–38.

[6] Gelfond, M., and Lifschitz, V. Logic Program with Classical Negation. In *Proceedings of the 7th Int. Conf. on Logic Programming* (June 1990), D. H. D. Warren and P. Szeredi, Eds., MIT, pp. 579–597.

[7] Gottlob, G. Complexity and expressive power of disjunctive logic programming. In *Logic Programming - Proceedings of the 1994 International Symposium* (Massachusetts Institute of Technology, 1994), M. Bruynooghe, Ed., The MIT Press, pp. 23–42.

[8] Hansson, S. O. *Kernel Contraction. The Journal of Symbolic Logic 59* (1994), 845–859.

[9] Hansson, S. O. *Semi-Revision. Journal of Applied Non-Classical Logic 7* (1997), 151–175.

[10] Hansson, S. O. *Theoria: Special Issue on Non-Prioritized Belief Revision.* Department of Philosophy, Uppsala University, 1997.

[11] Makinson, D. *Screened Revision.* In **Theoria** [10].

[12] Simari, G. I., and Falppa, M. A. Belief dynamics and explanations in ansprolog*. In *Proceedings of the IX Argentine Congress on Computer Science (CACIC)* (La Plata, Buenos Aires, Argentina, 2003), M. Naiouf, Ed., Universidad Nacional de La Plata, pp. 589–600.