Belief Dynamics and Explanations in AnsProlog*

Gerardo I. Simari [†] Marcelo A. Falappa

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)[‡]
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

Abstract

Knowledge representation models are very important in the design of intelligent agents because they provide with mechanisms to manage beliefs and their dynamics. In this paper, we propose the use of AnsProlog* as a knowledge representation language, and develop a Non Prioritized Belief Revision operator based on the Answer Set semantics and the use of explanations. This operator is suitable for multiagent environments, in which agents can exchange information by having dialogues which explain their respective beliefs. A simple, yet complete example follows the presentation of this operator.

Keywords: Belief Change, Knowledge Representation, Logic Programming, Answer Set Semantics, AnsProlog*.

1 Introduction and Motivation

The design of intelligent agents is greatly influenced by the many different models that exist to represent knowledge. It is essential that such agents have computationally adequate mechanisms to manage its knowledge which, more often than not, may be incomplete and even inconsistent. It is also important for an agent to be able to obtain new conclusions that allow it to reason about the state of the world in which it is embedded [FS03].

It has been shown that this problem cannot be solved within the realm of *Classic Logic*. This situation has triggered the development of a series of logical formalisms that extend the classic ones. These proposals often carry the names of *Nonmonotonic*, or *Defeasible Reasoning*. Some examples of such models are Reiter's Default Logic, Moore's Autoepistemic Logic, McCarthy's Circumscription, McDermott and Doyle's Nonmonotonic Logics, and Belief Revision (also called Belief Change). This last formalism was introduced by Gärdenfors and later extended by Alchourrón, Gärdenfors, and Makinson [AGM85, Gä88].

In particular, *Belief Revision* has as its main objective to model the dynamics of knowledge, that is, the way in which an agent's knowledge must be updated when it finds new information.

[†]Partially supported by *Comisión de Investigaciones Científicas* (CIC) - Gobierno de la Provincia de Buenos Aires, Argentina.

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

In other words, in what way must an agent's model of the world be updated when its sensors obtain up to date information about the environment? In this sense, it is in the field of *Cognitive Robotics* where belief revision finds its most appropriate application. Agents (be them physical or software) must have the necessary flexibility in order to change their beliefs because they are the main motivators of their actions. The idea of belief revision finds an example close to our basic intuitions in the legal environment, where the promulgation of a new law demands that some of the rest be "revised" and others removed to keep the set of laws consistent. In this example, the three basic operations of belief revision known as *expansion, contraction*, and *revision* are respectively mapped into promulgation, derogation, and amendment of laws. These same ideas are also applicable to environments that are 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 their knowledge. This property does not seem plausible 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 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 [Mak97, Han97b], Sven Ove Hansson's semirevision operators [Han97a], André Fuhrmann's merge operations [Fuh97] and Falappa *et al.*'s [FKIS02] 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).

A classic example of this situation is the following: Suppose that a person believes that (α) *all mammals can jump* and (β) *Titus is a mammal.* Thus, he will believe (δ) *Titus can jump.* Later on, another person states that *Titus cannot jump.* If δ is dropped, then α or β will have to be dropped as well. However, it does not seem to be rational to incorporate external beliefs without pondering them. An explanation should be required in order to incorporate such information, especially in the case where 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: *Titus cannot jump because he is an elephant*, and *elephants are mammals, but they cannot jump.* Now, the sentences in the explanans can be used to evaluate the new piece of information before it is incorporated. We propose the use of AnsProlog* as a language for the representation of beliefs, as well as for explanations.

The rest of this work is organized as follows: section 2 presents a review of the AnsProlog* language. The description of some of its most important syntactic subclasses, and examples for them, are also included. Section 3 describes the use of explanations in belief dynamics, and introduces the concepts used in the construction of the Answer Set Revision operator. The use of AnsProlog* as a knowledge representation language is introduced in section 4, along with a series of examples that illustrate it. Next, section 5 introduces the Answer Set Revision operator, and section 6 exemplifies it by means of the formalization of a multiagent domain in AnsProlog*. Finally, future work is discussed in section 7.

2 A Brief Review of the AnsProlog* Language

We now consider 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" [GL90]. An AnsProlog* program is a finite collection of rules of the form:

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

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. Some of these sub-classes are described next.

2.1 AnsProlog

AnsProlog programs are sets of rules in which the L_is are atoms and $k = 0$. The following is an example of an AnsProlog program:

$$jumps(X) \leftarrow mammal(X), \texttt{not } ab(X).$$
$$ab(X) \leftarrow elephant(X).$$
$$mammal(X) \leftarrow elephant(X).$$
$$mammal(larry) \leftarrow .$$
$$elephant(titus) \leftarrow .$$

From this program, we can conclude that *Larry* jumps, while *Titus* does not (because he is an elephant, and elephant's are "abnormal" in this sense).

2.2 AnsProlog$^{\neg not}$

In this class, the L_i's are atoms, $k = 0$, and $m = n$. These programs are similar to Prolog programs that do not make use of negation as failure, or the *cut* predicate. Another difference between Prolog and AnsProlog$^{\neg not}$ (and AnsProlog* in general) is that Prolog's semantics is defined with respect to a fixed inference mechanism, whereas AnsProlog*'s is not. In AnsProlog*, a program is interpreted as a *set* of rules, each of which is a *set* of literals (or literals preceded by `not`). Hence, the ordering of rules or literals does not affect the semantics of a given AnsProlog* program.

The following is an example of an AnsProlog$^{\neg not}$ program:

$flight(X, Z) \leftarrow flight(X, Y), flight(Y, Z).$
$flight(bhi,\ eze).$
$flight(eze,\ jfk).$

This program is an example of the *transitive closure* of a given relation. We can conclude from it that there is a flight from *bhi* to *jfk* (the facts also express that there are flights from *bhi* to *eze* and from *eze* to *jfk*).

2.3 AnsProlog$^{\neg}$

Programs in this class allow the presence of "¬" in their rules; the only restriction they impose is $k = 0$. The following is an example of an AnsProlog$^{\neg}$ program, similar to the one presented for AnsProlog:

$jumps(X) \leftarrow mammal(X),$ `not` $\neg jumps(X).$
$\neg jumps(X) \leftarrow elephant(X).$
$mammal(X) \leftarrow elephant(X).$
$mammal(larry) \leftarrow .$
$elephant(titus) \leftarrow .$

As in the example for the AnsProlog class, from this program we can conclude that *Larry* jumps and *Titus* does not.

2.4 AnsProlog^{or}

This class introduces disjunction in the head of the rules; however, it does impose the restriction that the L_i's must be atoms. It must be noted that the "`or`" operator is different from "$\vee$". The first one is called *epistemic disjunctions* [GL91]. The sentence "A `or` B" means *"A is believed to be true or B is believed to be true"*. Therefore, A `or` $\neg A$ is not always true, as is $A \vee \neg A$.

A simple example of an AnsProlog^{or} program is the following, from which we can conclude that *Larry* is a bird or an insect, but we cannot conclude both.

$insect(X)$ `or` $bird(X) \leftarrow flies(X)$
$flies(larry) \leftarrow .$

This is only a subset of the different classes that have been studied. The importance of defining such 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 [Got94]. The study of how the computational complexity of these and other problems is related to belief dynamics under this representation is left as future work.

3 Belief Revision by Sets of Sentences

The theory of belief change assumes that an agent's beliefs are represented by *belief sets* (sets of sentences closed under logical consequence) or *belief bases* (arbitrary sets of sentences). It is clear that, 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 [FKIS02] 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.
- Then the consistency is restored by a contraction by falsum.

An *explanation* can be defined as follows [FKIS02]. 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.

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. The first approach for this operator, described here, does not take this possibility into account. Therefore, an explanation is either fully accepted or fully rejected, as we will see below. An Answer Set revision operator that incorporates partial acceptance is left for future work.

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 process, and how it can be implemented, will not be treated here, and is the topic of future work.

This mechanism could be applied in some real life situations or dialogues between agents. In this work, we develop a new operator of this type, based on explanations, for multiagent systems.

4 An AnsProlog* Representation

An agent's beliefs can be directly represented by an AnsProlog program. This program's *answer set semantics* will represent such beliefs. Such semantics can be defined as follows [GL02]:

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.

For programs having only one answer set, there is no difference between these two relations. As will be noted later, the Answer Set Revision operator is not affected by this choice in belief entailment. In the same way, any explanation for a given belief can be represented by an AnsProlog program; the program's rules can be interpreted as the explanans, and the desired explanandum will be entailed by the program. A simple example of a belief base is the following program Π_1:

$jumps(X) \leftarrow mammal(X).$
$mammal(X) \leftarrow elephant(X).$
$mammal(titus) \leftarrow .$

Π_1 has as its only answer set: $S_1 = \{mammal(titus), elephant(titus), jumps(titus)\}$. An explanation represented by an AnsProlog* program, related to this example, is the following program φ_1:

$jumps(X) \leftarrow mammal(X).$
$mammal(larry) \leftarrow .$

Like Π_1, this program explains that *Larry* jumps because he is a mammal, and has the sole answer set: $E_1 = \{mammal(larry), jumps(larry)\}$. This program can then be used as an explanation for $jumps(larry)$, where φ_1 is the explanans and $jumps(larry)$ is the explanandum. An example of an explanation that conflicts with Π_1 is the following program φ_2:

$\neg jumps(X) \leftarrow elephant(X).$
$elephant(titus) \leftarrow .$

which has as its only answer set: $E_2 = \{elephant(titus), \neg jumps(titus)\}$. This answer set clearly disagrees with Π_1, which entails the belief $jumps(titus)$. In the next section, we introduce a non prioritized belief revision operator based on the representation described.

5 Non Prioritized Answer Set Revision

An agent's belief base can be represented by an AnsProlog* program, and its beliefs by considering the program's *answer set semantics* [GL02]. Therefore, a belief will belong to a given belief base if and only if it is entailed by its associated logic program. The answer sets shown in the previous section are examples of knowledge bases generated from logic programs. Because *explanations* can be seen as special cases of belief bases (they are sets of beliefs specifically designed to entail a given belief), they can be represented by AnsProlog* programs in the same way. It must be noted that, even though the answer set semantics yields a unique set, the complete set of answer sets is available at any time because it can be computed directly from the program. This fact plays an important role in the construction of the Answer Set Revision operator, as we will see below.

When an agent with belief base Π is faced with an explanation φ for a given belief α, the agent must establish the status of the given information. There are three possible scenarios:

(i). α is consistent with *every* answer set of Π; *i.e.*,
for every S such that S is an answer set of Π, $\neg\alpha \notin S$.

(ii). α is only consistent with *some* of the answer sets of Π; *i.e.*, there is at least one answer set S of Π such that $\neg\alpha \notin S$.

(iii). α is *not* consistent with *any* of the answer sets of Π; *i.e.*, for every S such that S is an answer set of Π, $\neg\alpha \in S$.

Situation (i) is the simplest because the explanation agrees completely with the beliefs in Π. Therefore, new knowledge can be safely incorporated by adding φ to Π, *i.e.*, the new knowledge base will be $\Pi_{rev} = \Pi \cup \varphi$.

In case (ii), the proposed explanation must be evaluated with respect to the counter-explanations that can be constructed from Π. Such counter-explanations are subsets of Π that have $\neg\alpha$ as their explanandum. If the new knowledge is to be incorporated, the new knowledge base will be $\Pi_{rev} = \Pi \cup \varphi$, as in ($i$). This addition to Π will automatically remove those answer sets that were in disagreement with φ, and will incorporate new ones. In case φ is rejected, Π is left unaltered.

The last situation is in some sense the "opposite" of case (i); it is the most difficult because the explanation presented is in complete disagreement with the beliefs in Π. As in (ii), the explanation for the new information must be evaluated with respect to the counter-explanations that can be constructed from Π. If φ is accepted, the inconsistency must be removed from $\Pi \cup \varphi$ by traditional means. The belief base will remain unaltered if the agent rejects the explanation. As was mentioned earlier, the choice in semantics (cautious or brave) is irrelevant for this operator. This situation arises because the beliefs that could only be bravely entailed (and not cautiously entailed) are not considered to be "stable" beliefs, in the sense that a given belief α and $\neg\alpha$ could be entailed by such semantics. An extension where this semantics is considered is left for future work.

6 An Example of Answer Set Revision

In this section, we will show a simple (but complete) example of the concepts introduced above. The domain is intentionally simple to keep the example from running too long.

Suppose an agent wishes to make an investment of up to $20,000, so he goes out and scouts the market to see where he can invest his money. Before going in search of options, his knowledge base could be the following program Π:

$$\bot \leftarrow buy(X), price(X, PX), invest(M), greater(PX, M).$$

This rule is called a *restriction*, and it states that the agent cannot buy X if its price is greater than the amount intended for investment. Another restriction, similar to this one, states that the agent cannot invest more money than it owns:

$$\bot \leftarrow invest(I), my\_money(M), greater(I, M).$$

The following rule represents that the agent does not wish to buy an item if it has a feature that is considered negative.

$$\neg buy(X) \leftarrow feature(X, Y), negative\_feature(Y).$$

The belief base is completed with the following facts:

$invest(20000).$
$my\_money(28000).$
$negative\_feature(lose\_value).$
$negative\_feature(needs\_improvement).$
$negative\_feature(expensive).$

The first fact states that the agent is willing to invest $20,000, and the second one declares that the agent actually has $28,000. The last three facts mean that the agent considers the features mentioned to be undesirable in an item.

The agent then goes shopping, and a variety of salesmen offer him: a car for $16,000, a piece of land for $21,000, and an apartment for $30,000. After seeing the options, the belief base has been revised; it has been extended to include what the agent has observed. These inclusions can be directly made because they are in agreement with the answer sets of Π:

$price(car, 16000).$
$price(land, 21000).$
$price(apartment, 30000).$

Furthermore, the agent has also gathered information on each of the products:

$feature(car, loses\_value).$
$feature(car, low\_price).$

The car will lose value immediately and by the day after purchase, but it is at a very low price.

$feature(land, needs\_improvement).$
$feature(land, increases\_value).$

On the other hand, the piece of land will increase its value and could be worth twice in just a few years. Nevertheless, to make a more immediate profit, it needs improvement.

$feature(apartment, expensive).$
$feature(apartment, collect\_rent).$

The apartment is well over the price the agent intended to invest. Nevertheless, it has a very attractive feature: the agent can make money quite easily by renting it to others.

Now, the agent has to make a decision among the three options (if any); each of the salesmen gave him good reasons to buy their item. The salesmen basically built *explanations* that point out the advantages of the item being offered by them. The first salesmen offered the car, and built the following explanation:

$$\begin{aligned}\varphi_c = \{ \quad & buy(X) \leftarrow advantage(X,Y), important\_feature(Y)., \\ & advantage(car, low\_price)., \\ & important\_feature(low\_price).\}\end{aligned}$$

which minimally entails $buy(car)$. The rest of the salesmen built similar explanations φ_l and φ_a to try and convince the agent that their product is the one to buy.

Of course, the agent is able to build *counter-explanations* for each of the explanations proposed. The counter-explanation for φ_c is the following explanation for $\neg buy(car)$:

$$\begin{aligned}\varphi_{\neg c} = \{ \quad & \neg buy(X) \leftarrow feature(X,Y), negative\_feature(Y). \\ & feature(car, lose\_value), \\ & negative\_feature(lose\_value).\}\end{aligned}$$

Similar counter-explanations $\varphi_{\neg l}$ and $\varphi_{\neg a}$ can be built to explain $\neg buy(land)$ and $\neg buy(apartment)$, respectively. In order to gather the information on all the products, the agent initially rejected all of these explanations, and therefore his knowledge base has not been modified with respect to what to buy.

The agent is now faced with a revision of his knowledge. Before he went shopping, he expected to spend $20,000, even though he actually has $28,000 to spend. There are two possible situations that can arise after the agent revises his beliefs:

- He decides not to change his spending limit.
- He decides to raise his spending limit to try and buy one of the more expensive offers.

In the first case, the agent might decide to buy the car, which is the only option left if the spending limit is not changed. Therefore, he accepts the salesman's offer, and his new knowledge base will entail $buy(car)$, which will involve removing $negative\_feature(loses\_value)$ from Π.

In the second case, the agent may decide that he wants to buy the land, in which case his belief base will be revised to include $invest(21000)$ instead of $invest(20000)$. If this isn't done, there would be an inconsistency because both $buy(land)$ and $\neg buy(land)$ could be entailed. Furthermore, as before, $negative\_feature(needs\_improvement)$ must be removed from Π.

The agent might also decide to purchase the apartment. Even though this is initially inconsistent with his knowledge (he has $28,000, and the apartment costs $30,000), he adopts the wish anyway, because he is also planning on bargaining with the salesman in order to get a better price.

7 Summary

In this work, we have proposed the use of AnsProlog* as a knowledge representation language that is useful in modeling belief dynamics. By interpreting a logic program under its Answer Set semantics, the associated entailment operator can be used to test membership to a belief base. Likewise, an explanation for a given belief can be represented by a logic program that entails such belief. This representation is then used in the definition of the Answer Set Revision operator, described and exemplified throughout this work.

We regard the topic of Computational Complexity as important future research, because the computational cost of the various operations that can be performed on AnsProlog* programs is directly related to the computational cost of the defined operator.

References

[AGM85] Carlos Alchourrón, Peter Gärdenfors, and David Makinson. *On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. The Journal of Symbolic Logic*, 50:510–530, 1985.

[FKIS02] Marcelo A. Falappa, Gabriele Kern-Isberner, and Guillermo R. Simari. *Belief Revision, Explanations and Defeasible Reasoning. Artificial Intelligence Journal*, 141:1–28, 2002.

[FS03] Marcelo A. Falappa and Gerardo I. Simari. Non prioritized reasoning in intelligent agents. In Nelson Acosta, editor, *Proceedings of the V Workshop de Investigadores en Ciencias de la Computación*, pages 237–240. Universidad del Centro de la Provincia de Buenos Aires, Tandil, Buenos Aires, Argentina, 2003.

[Fuh97] André Fuhrmann. *An Essay on Contraction.* Studies in Logic, Language and Information, CSLI Publications, Stanford, California, 1997.

[Gä88] Peter Gärdenfors. *Knowledge in Flux: Modelling the Dynamics of Epistemic States.* The MIT Press, Bradford Books, Cambridge, Massachusetts, 1988.

[GL90] Michael Gelfond and Vladimir Lifschitz. Logic Program with Classical Negation. In David H. D. Warren and Peter Szeredi, editors, *Proceedings of the 7th Int. Conf. on Logic Programming*, pages 579–597. MIT, June 1990.

[GL91] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[GL02] Michael Gelfond and Nicola Leone. Logic programming and knowledge representation—the A-prolog perspective. *Artificial Intelligence*, 138(1–2):3–38, 2002.

[Got94] Georg Gottlob. Complexity and expressive power of disjunctive logic programming. In Maurice Bruynooghe, editor, *Logic Programming - Proceedings of the 1994 International Symposium*, pages 23–42, Massachusetts Institute of Technology, 1994. The MIT Press.

[Han97a] Sven Ove Hansson. *Semi-Revision*. *Journal of Applied Non-Classical Logic*, 7:151–175, 1997.

[Han97b] Sven Ove Hansson. *Theoria: Special Issue on Non-Prioritized Belief Revision*. Department of Philosophy, Uppsala University, 1997.

[Mak97] David Makinson. *Screened Revision*. In **Theoria** [Han97b].