"Classical" Negation
in
Non-Monotonic Reasoning and Logic Programming*

José Alferes†
DM, U. Évora and A.I. Centre, UNL
2825 Monte da Caparica, Portugal
(jja@di.fct.unl.pt)

Luís Moniz Pereira*
A.I. Centre, U.Nova de Lisboa and CRIA Uninova
2825 Monte da Caparica, Portugal
(lmp@di.fct.unl.pt)

Teodor C. Przymusinski‡
Department of Computer Science
University of California
Riverside, CA 92521, USA
(teodor@cs.ucr.edu)

1 Introduction

Logic programs, deductive databases, and more generally non-monotonic theories, use various forms of *default negation*, *not* F, whose major distinctive feature is that *not* F is assumed "by default", i.e., it is assumed in the absence of sufficient evidence to the contrary. The meaning of *"sufficient evidence"* depends on the specific semantics used. For example, in Reiter's *Closed World Assumption*, *CWA* [Rei78], *not* A is assumed for atomic A if A is not provable, or, equivalently, if there is a minimal model in which A is false. On the other hand, in Minker's *Generalized Closed World Assumption*, *GCWA* [Min82, GPP89], or in McCarthy's Circumscription, *CIRC* [McC80], *not* A is assumed only if A is false in *all* minimal models. In Clark's *Predicate Completion* semantics for logic programs [Cla78], this form of negation is called *negation-as-failure* because *not* A is derivable whenever attempts to prove A finitely fail.

The more recent semantics proposed for logic programs and deductive databases, such as the *stable semantics* [GL88], *well-founded semantics* [VGRS90], *partial stable* or *stationary semantics* [Prz90] and *static semantics* [Prz95b], propose even more sophisticated meanings for default negation, closely related to more general non-monotonic formalisms such as *Default Logic*, *DL* [Rei80], *AutoEpistemic Logic*, *AEL* [Moo85], and *AutoEpistemic logic of Beliefs*, *AEB* [Prz97a]. Under all of these semantics, however, default negation "*not* " significantly differs from classical negation "$\neg$".

*Extended abstract of this paper appeared in the Proceedings of the European Workshop on Logic in Artificial Intelligence (JELIA'96), Lecture Notes on Artificial Intelligence, Springer Verlag, 1996, vol. 1126, pp. 143-163.
†Partially supported by JNICT-Portugal
‡Partially supported by the National Science Foundation grant #IRI-9313061.

For example, the formula:

$$charged(x) \wedge \neg guilty(x) \supset acquitted(x)$$

says that a person charged with a crime should be acquitted if he or she is actually proven to be not guilty. On the other hand, the formula

$$charged(x) \wedge not\ guilty(x) \supset acquitted(x)$$

says that one should be acquitted of any charges by default unless sufficient evidence of that person's guilt is demonstrated. In particular, one should be acquitted if there is no information available about one's guilt or innocence.

1.1 Default negation does not suffice

Although default negation proved to be quite useful in various domains and application frameworks, it is not the only form of negation that is needed in non-monotonic formalisms. Indeed, while default negation *not* A of an atomic formula A is always assumed "by default", we often need to be more careful before jumping to negative conclusions. For example, it would make little sense to say

$$not\ innocent(x) \supset guilty(x)$$

to express the fact that being guilty is the opposite of being innocent because it would imply that people should be considered guilty "by default". Similarly, we would not want to say:

$$crossing(x), not\ train(x) \supset drive(x)$$

to express the fact that one may safely cross a railway track if (s)he first makes sure that there is no train approaching, because such a clause would imply that (s)he should cross the railway even if (s)he did not bother to look around at all.

1.2 Classical negation does not provide a solution

One could argue that the above feature constitutes the main difference between default negation and classical negation and therefore if default negation *not* F does not work in some particular context then classical negation $\neg F$ should be used instead. One problem with such an approach is that classical negation $\neg F$ is not part of the language of standard logic programs and deductive databases and thus it is not immediately available without an appropriate extension of their languages and semantics, such as the one proposed in [Prz95b]. More importantly, however, classical negation $\neg F$ has some specific characteristics which often make it unsuitable or at least insufficient in the context of non-monotonic reasoning:

- Classical negation $\neg F$ satisfies the so called *"law of the excluded middle"*, $F \vee \neg F$, for every formula F. While it is a natural axiom in the domain of formal logic and mathematics, it is not suitable for *commonsense reasoning* where we are often faced with situations in which some things are true, some are false and yet some others are not (perhaps, yet) determined to be either true or false. For example, an employer may be faced with some candidates who are clearly qualified for the job, and thus should be hired, some who are clearly not qualified, and thus should be rejected, and yet some other candidates whose qualifications are not clear and who therefore should be interviewed in order to determine their suitability for the job (see [Gel92]). However, if we attempt to describe the first two statements using real classical negation "$\neg$" in the clauses:

$$qualified(x) \supset hire(x), \qquad \neg qualified(x) \supset reject(x)$$

then, by virtue of the law of the excluded middle, we will immediately conclude that every candidate must either be hired or rejected without leaving any room for those who need to be further interviewed.

The either/or character of the law of the excluded middle presupposes that we have sufficient information to determine, at least in principle, whether any given property or its opposite is true. However, when dealing with *incomplete information* or with properties involving *gradual change* we often encounter "gray areas" in which neither the property nor its opposite seem to hold.

The law of the excluded middle leads to particularly unintuitive results when used with reference to non-existing or non-applicable properties or objects, e.g., in statements like "the pink elephant in my pocket is either old or not old" or "the chair is either happy or unhappy". It would be clearly more appropriate to say "the chair is neither happy nor unhappy".

Moreover, the law of the excluded middle is highly *non-constructive* and thus does not fit well within the somewhat vague "spirit" of logic programming and deductive databases which seems to rely heavily on *"directional reasoning"*, i.e., on first establishing the validity of premises of a clause before deducing its consequents and not the other way around. As a result of its non-constructive character, the law of the excluded middle is also computationally expensive.

- Classical literals A and $\neg A$ are not treated symmetrically by default negation. More precisely, in the absence of evidence to the contrary, default negation *not* A of *positive* literals (atoms) A is always assumed, while the same does not apply to negative literals $\neg A$. For example, even though GCWA (or circumscription) applied to the theory

$$charged(tom)$$

$$charged(x) \wedge guilty(x) \supset convicted(x)$$

implies *not* $convicted(tom)$[1] this is no longer the case if we perform a simple syntactic substitution of $\neg innocent$ for $guilty$ thus obtaining

$$charged(tom)$$

$$charged(x) \wedge \neg innocent(x) \supset convicted(x)$$

because the new theory has minimal models in which $convicted(tom)$ is true.

As we can see, a simple syntactic substitution of $\neg innocent$ for $guilty$ turns out to have a significant impact on the semantics of the resulting theory. As a consequence, default negation, *not* F, is heavily dependent on the syntactic form of the formula F, and, in particular, it is *not invariant* under the *equivalence* or *renaming* of predicates.

Often times, however, we would like to apply negation by default symmetrically to both positive and negative literals. For instance, in the preceding example, given no information about Tom's qualifications we would like to conclude that Tom was neither found qualified nor unqualified, i.e., that both *not* $qualified(Tom)$ and *not* $(\neg qualified(Tom))$ hold. This would provide us with an important information that Tom has to be interviewed.

1.3 Symmetric negation

Based on the above discussion, we conclude that, in addition to default negation, "*not* ", and classical negation, "$\neg$", which are already used in non-monotonic reasoning, we need a new form of negation, denoted here by "$\sim$", which has the following properties:

- Similarly to classical negation, it satisfies the basic properties of negation such as the double-negation law $\sim(\sim F) \equiv F$ and the distributive laws $\sim(F \vee G) \equiv \sim F \wedge \sim G$ and $\sim(F \wedge G) \equiv \sim F \vee \sim G$;
- However, as opposed to classical negation, symmetric negation "$\sim$" is to be treated *symmetrically* by the default negation "*not*", i.e., in the absence of evidence to the contrary, both *not* A and *not* $(\sim A)$ are to be assumed. More generally, the non-monotonic semantics should be invariant under the renaming of any predicates from A to $\sim A$ and vice versa;
- Moreover, also in contrast to classical negation, "$\sim$" should not be required to satisfy the law of the excluded middle, $A \vee \sim A$;
- It also should substantially differ from default negation in that the negation $\sim A$ of atomic formulae A should not be assumed by default.

In view of the second property, we will call the form of negation "$\sim$" satisfying the above three properties *symmetric negation*. It is important to point out that the above conditions are *not* intended to be jointly sufficient to fully characterize symmetric negation and thus they do not uniquely determine this notion but rather represent constraints which may be met by several different notions.

Gelfond and Lifschitz [GL90] were the first to point out the need for such negation in logic programming and to propose a specific semantics for such negation for logic programs with the stable semantics. Somewhat unfortunately, they called their negation *"classical negation"*. Independently, Pearce and Wagner in [PW90a] studied Nelson's strong negation [Nel49] and pointed out the need for using strong negation in non-monotonic reasoning. However their strong negation is different from the strong negation defined in this paper (see Remark 3.2).

Subsequently, several researchers proposed different, often incompatible, forms of symmetric negation for various semantics of logic programs and deductive databases [DR91, PA92, Prz91, Prz95b, Pea90, Wag93]. To the best of our knowledge, however, no systematic study of symmetric negation in non-monotonic reasoning was ever attempted in the past[2]. In this paper we conduct such a systematic study of symmetric negation:

- We introduce and discuss two natural, yet different, definitions of symmetric negation: one is called *strong negation* and the other is called *explicit negation*. For logic programs with the stable semantics, both symmetric negations coincide with Gelfond-Lifschitz's "classical negation".
- We study properties of strong and explicit negation and their mutual relationship as well as their relationship to default negation "*not* " and classical negation "$\neg$".
- We show how one can use symmetric negation to provide natural solutions to various knowledge representation problems, such as theory and interpretation update and belief revision.
- Rather than to limit our discussion to some narrow class of non-monotonic theories, such as the class of logic programs with some specific semantics, we conduct our study so that it is applicable to a broad class of non-monotonic formalisms, which includes the well-known formalisms of circumscription, autoepistemic logic and all the major semantics recently proposed for logic programs (stable, well-founded, stationary, static and others).
- In order to achieve the desired level of generality, we define the notion of symmetric negation in the knowledge representation framework of *AutoEpistemic logic of Beliefs*, *AEB*, introduced by Przymusinski in [Prz97a], which was shown to isomorphically contain all of the above mentioned formalisms as special cases. As a result, we automatically provide the corresponding notions of symmetric negation for all formalisms embeddable into *AEB*.

[2] For logic programs, an extensive study was carried out in [AP92].

The paper is organized as follows. In the next section we briefly recall the definition and basic properties of the Autoepistemic Logic of Beliefs, AEB, as well as the translation of logic programs into belief theories in AEB. In Section 3 we introduce strong negation, the first form of symmetric negation, discuss its basic properties and illustrate its simple application to the problem of interpretation and theory update.

In Section 4 we introduce explicit negation (the second form of symmetric negation), we establish its basic properties and in Section 5 we discuss applications of explicit negation to knowledge representation. In Section 6 we briefly discuss some issues involved in the implementation of explicit negation. We conclude with some final remarks.

2 Autoepistemic Logic of Beliefs

First we briefly recall the definition and basic properties of the *Autoepistemic Logic of Beliefs*, AEB, introduced by Przymusinski in [Prz97a]. Consider a fixed propositional language $\mathcal{L}$ with standard connectives ($\vee$, $\wedge$, $\supset$, $\neg$) and the propositional letter $\perp$ (denoting *false*). The language of AEB is a propositional modal language $\mathcal{L}_{AEB}$ obtained by augmenting the language $\mathcal{L}$ with a modal operator $\mathcal{B}$, called the *belief* operator. The atomic formulae of the form $\mathcal{B}F$, where F is an arbitrary formula of $\mathcal{L}_{AEB}$, are called *belief atoms*. The formulae of the original language $\mathcal{L}$ are called *objective*. Observe that arbitrarily deep level of *nested beliefs* is allowed in belief theories. Any theory T in the language $\mathcal{L}_{AEB}$ is called a *belief theory*.

Definition 2.1 (Belief Theory) *By an* autoepistemic theory of beliefs, *or just a* belief theory, *we mean an arbitrary theory in the language $\mathcal{L}_{AEB}$, i.e., a (possibly infinite) set of arbitrary clauses of the form:*

$$B_1 \wedge ... \wedge B_k \wedge \mathcal{B}G_1 \wedge ... \wedge \mathcal{B}G_l \wedge \neg\mathcal{B}F_1 \wedge ... \wedge \neg\mathcal{B}F_n \supset A_1 \vee ... \vee A_m$$

where $k, l, m, n \geq 0$, $A_i s$ and $B_i s$ are objective atoms and $F_i s$ and $G_i s$ are arbitrary formulae of $\mathcal{L}_{AEB}$. Such a clause says that if the $B_i s$ are true, the $G_i s$ are believed, and the $F_i s$ are not believed then one of the $A_i s$ is true.

By an affirmative belief theory *we mean any belief theory all of whose clauses satisfy the condition that $m > 0$.* □

We assume the following two simple axiom schemata and one inference rule describing the arguably obvious properties of belief atoms:

(D) Consistency Axiom: $\neg\mathcal{B}\perp$.

(K) Normality Axiom: For any formulae F and G: $\mathcal{B}(F \supset G) \supset (\mathcal{B}F \supset \mathcal{B}G)$.

(N) Necessitation Rule: For any formula F: $\frac{F}{\mathcal{B}F}$

The first axiom states that tautologically false formulae are *not* believed. The second axiom states that if we believe that a formula F implies a formula G and if we believe that F is true then we believe that G is true as well. The necessitation inference rule states that if a formula F has been proven to be true then F is believed to be true.

Definition 2.2 (Formulae Derivable from a Belief Theory)
For any belief theory T, we denote by $Cn_{AEB}(T)$ the smallest set of formulae of the language $\mathcal{L}_{AEB}$ which contains the theory T, all the (substitution instances of) the axioms (K) and (D) and is closed under both standard propositional consequence and the necessitation rule (N).

We say that a formula F is derivable *from theory T in the logic AEB if F belongs to $Cn_{AEB}(T)$. A belief theory T is* consistent *if the theory $Cn_{AEB}(T)$ is consistent.* □

In the presence of the axiom (K), the axiom (D) is equivalent to the axiom $\mathcal{B}F \supset \neg\mathcal{B}\neg F$, stating that if we believe in a formula F then we do *not* believe in its negation $\neg F$. Moreover, it is easy to see that the axioms imply that the belief operator $\mathcal{B}$ obeys the *distributive law for conjunctions* $\mathcal{B}(F \wedge G) \equiv \mathcal{B}F \wedge \mathcal{B}G$ (see [Prz97a]). For readers familiar with modal logics it should be clear by now that we are, in effect, considering here a *normal* modal logic with one modality $\mathcal{B}$ which satisfies the consistency axiom (D) [MT94a].

Let us recall that throughout the paper we view the modal language $\mathcal{L}_{AEB}$ simply as a propositional language extending the underlying *objective* language $\mathcal{L}$. In particular, formulae of the form $\mathcal{B}F$ are considered to be propositional atoms in the extended language $\mathcal{L}_{AEB}$ and models of belief theories, i.e., models of theories in the language $\mathcal{L}_{AEB}$, are just classical models of propositional theories.

Definition 2.3 (Minimal Models) *[Prz97a]*
By a minimal model *of a belief theory T we mean a model M of T with the property that there is* no *smaller model N of T which coincides with M on belief atoms $\mathcal{B}F$. If a formula F is true in all minimal models of T then we write: $T \models_{\min} F$ and say that F is* minimally entailed *by T.* □

For readers familiar with *circumscription*, this means that we are considering circumscription $CIRC(T;\mathcal{K})$ of the theory T in which atoms from the objective language $\mathcal{L}$ are minimized while the belief atoms $\mathcal{B}F$ are fixed, i.e., $T \models_{\min} F \equiv CIRC(T;\mathcal{L}) \models F$. In other words, minimal models are obtained by first assigning *arbitrary* truth values to the belief atoms and then *minimizing* objective atoms.

2.1 Static Autoepistemic Expansions

The intended meaning of belief atoms $\mathcal{B}F$ is based on the principle of *predicate minimization*:

$$\mathcal{B}F \text{ if } F \text{ is minimally entailed } \equiv F \text{ is true in all minimal models.}$$

Accordingly, beliefs in AEB can be called *minimal beliefs* (see [Min82, GPP89] and [McC80]). As in Moore's Autoepistemic Logic, also in the Autoepistemic Logic of Beliefs the intended meaning of belief atoms is implemented by defining plausible sets of beliefs that an ideally rational and introspective agent may hold, given a set of premises T.

Definition 2.4 (Static Autoepistemic Expansion) *[Prz97a]*
A belief theory $T^\diamond$ is called a static autoepistemic expansion *of a belief theory T if it satisfies the following fixed-point equation:*

$$T^\diamond = Cn_{AEB}(T \cup \{\mathcal{B}F : T^\diamond \models_{\min} F\}),$$

where F ranges over all formulae of $\mathcal{L}_{AEB}$. □

The definition of static autoepistemic expansions is based on the idea of building an expansion $T^\diamond$ of a belief theory T by closing it with respect to: (i) the derivability in the logic AEB, and, (ii) the addition of belief atoms $\mathcal{B}F$ satisfying the condition that the formula F is minimally entailed by $T^\diamond$. Consequently, the definition of static expansions *enforces* the intended meaning of belief atoms described above. Note that negations $\neg\mathcal{B}F$ of the remaining belief atoms are not *explicitly* added to the expansion although some of them will be forced in by the Normality and Consistency Axioms.

It turns out that every belief theory T in AEB has the *least* (in the sense of set-theoretic inclusion) static expansion $T^\diamond$ which has an *iterative* definition as the *least fixed point* of the monotonic[3] belief closure operator:

$$\Psi_T(S) = Cn_{AEB}(T \cup \{\mathcal{B}F : S \models_{\min} F\}),$$

where S is an arbitrary belief theory and the F's range over all formulae of $\mathcal{L}_{AEB}$.

[3] Strictly speaking the operator is only monotone on the lattice of all theories of the form $T \cup M_{bel}$ where T is fixed and M_{bel} ranges over all belief formulae.

Theorem 2.1 (Least Static Expansion) *[Prz97a]*
Every belief theory T in AEB has the least *static expansion, namely, the least fixed point $T^\diamond$ of the monotonic belief closure operator Ψ_T.*

More precisely, the least static expansion $T^\diamond$ of a belief theory T can be constructed as follows. Let $T^0 = T$ and suppose that T^α has already been defined for any ordinal number $\alpha < \beta$.

- *If $\beta = \alpha + 1$ is a successor ordinal then define:*

$$T^{\alpha+1} = \Psi_T(T^\alpha) = Cn_{AEB}(\ T \cup \{\mathcal{B}F : T^\alpha \ \models_{\min}\ F\} \text{ and } F \in \mathcal{L}_{AEB}\ ).$$

- *Otherwise, define $T^\beta = \bigcup_{\alpha<\beta} T^\alpha$.*

The sequence $\{T^\alpha\}$ is monotonically increasing and has a unique fixed point $T^\diamond = T^\lambda = \Psi_T(T^\lambda)$, for some ordinal λ. □

According to the following recent result established in [BDP96], for any *finite* belief theory only *one iteration* of the operator Ψ_T is needed.

Theorem 2.2 *[BDP96] For every finite belief theory T, the least static expansion $T^\diamond$ of T is given by:*

$$T^\diamond = \Psi_T(T) = Cn_{AEB}(T \cup \{\mathcal{B}F : T \models_{min} F\}). \ \square$$

Observe that the *least* static autoepistemic expansion $T^\diamond$ of T contains those and only those formulae which are true in *all* static autoepistemic expansions of T. It defines the so called *static semantics* of a belief theory T. It is easy to verify that a belief theory T either has a *consistent* least static expansion $T^\diamond$ or it does *not* have any consistent static expansions at all. Moreover, least static expansions of *affirmative* belief theories are always consistent [Prz97a].

Example 2.1 Consider the following belief theory T describing a simple diagnosis of car problems:

$$\begin{array}{lcl} \mathcal{B}\neg Broken & \supset & Runs \\ \mathcal{B}\neg Runs & \supset & Fix. \end{array}$$

Let us first observe that $T \models_{min} \neg Broken$, $T \models_{min} (Runs \Leftrightarrow \mathcal{B}\neg Broken)$ and $T \models_{min} (Fix \Leftrightarrow \mathcal{B}\neg Runs)$. Indeed, in order to find minimal models of T we just need to assign *arbitrary* truth values to the two belief atoms $\mathcal{B}\neg Broken$ and $\mathcal{B}\neg Runs$, and then *minimize* the objective atoms Fix, $Broken$ and $Runs$. We easily see that T has the following four minimal models[4]:

$$\begin{array}{lcl} M_1 & = & \{\mathcal{B}\neg Broken,\ \mathcal{B}\neg Runs,\ \neg Broken,\ Runs,\ Fix\}; \\ M_2 & = & \{\neg\mathcal{B}\neg Broken,\ \neg\mathcal{B}\neg Runs,\ \neg Broken,\ \neg Runs,\ \neg Fix\}; \\ M_3 & = & \{\neg\mathcal{B}\neg Broken,\ \mathcal{B}\neg Runs,\ \neg Broken,\ \neg Runs,\ Fix\}; \\ M_4 & = & \{\mathcal{B}\neg Broken,\ \neg\mathcal{B}\neg Runs,\ \neg Broken,\ Runs,\ \neg Fix\}. \end{array}$$

Since in all of them the formulae $\neg Broken$, $Runs \Leftrightarrow \mathcal{B}\neg Broken$ and $Fix \Leftrightarrow \mathcal{B}\neg Runs$ are satisfied, we infer that $T \models_{min} \neg Broken$, $T \models_{min} (Runs \Leftrightarrow \mathcal{B}\neg Broken)$ and $T \models_{min} (Fix \Leftrightarrow \mathcal{B}\neg Runs)$.

Consequently, since $T^1 = \Psi_T(T) = Cn_{AEB}(T\ \cup\ \{\mathcal{B}F : T \models_{\min} F\})$, we obtain:

$$\{\mathcal{B}\neg Broken, \mathcal{B}(Runs \Leftrightarrow \mathcal{B}\neg Broken), \mathcal{B}(Fix \Leftrightarrow \mathcal{B}\neg Runs)\} \subseteq T^1.$$

Since $T^1 \models Runs$, from the Necessitation rule it follows that $\mathcal{B}Runs \in T^1$. By the Consistency axiom we infer that $\neg\mathcal{B}\neg Runs \in T^1$ and thus also $\mathcal{B}(\neg\mathcal{B}\neg Runs) \in T^1$. Since

$$T^1 \models \mathcal{B}(\neg Fix \Leftrightarrow \neg\mathcal{B}\neg Runs)$$

[4] For simplicity, when describing static expansions of this and other examples we list only those elements of the expansion that are "relevant" to our discussion. In particular, we usually omit nested beliefs.

by the Normality axiom:

$$T^1 \models \mathcal{B}(\neg Fix) \Leftrightarrow \mathcal{B}(\neg\mathcal{B}\neg Runs)$$

and therefore $\mathcal{B}\neg Fix \in T^1$. It is easy to check that T^1 is (clearly, the least) fixed point of Ψ_T and therefore $T^\diamond = T^1 = Cn_{AEB}(T \cup \{\mathcal{B}\neg Broken, Runs, \mathcal{B}\neg Fix\})$ is the least static expansion of T. The static semantics of T asserts our belief that the car is not broken and thus runs fine and does not need fixing. One easily verifies that T does not have any other (consistent) static expansions. □

2.2 Embeddability of Circumscription and Autoepistemic Logic

One easily sees that propositional circumscription (and thus also CWA, $GCWA$ and $ECWA$ [Rei78, Min82, GPP89]) can be properly embedded into AEB.

Proposition 2.1 (Embeddability of Circumscription) *[Prz97a] Propositional circumscription, CWA, $GCWA$ and $ECWA$ are all properly embeddable into the Autoepistemic Logic of Beliefs, AEB. More precisely, if T is any objective theory then T has a unique static expansion $T^\diamond$ and any objective formula F is logically implied by the circumscription $CIRC(T)$ of T if and only if $T^\diamond$ logically implies the belief atom $\mathcal{B}F$:*

$$CIRC(T) \models F \;\equiv\; T^\diamond | = \mathcal{B}F. \;\Box$$

Moreover, a simple extension, $AELB$, of AEB, obtained by adding an additional *knowledge operator*, has been shown [Prz97a] to isomorphically contain Moore's Autoepistemic Logic, AEL.

2.3 Logic Programs as Belief Theories

Major semantics defined for normal and disjunctive *logic programs* are also embeddable into AEB [Prz97a]. In particular, this is true for the *stable semantics* [GL88], *well-founded semantics* [VGRS90], *partial stable* or *stationary semantics* [Prz90] and *static semantics* [Prz95b] of normal and disjunctive logic programs.

Suppose that P is a disjunctive logic program consisting of rules:

$$A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m \wedge not\, C_1 \wedge \ldots \wedge not\, C_n.$$

The translation of P into the affirmative belief theory $T_{\mathcal{B}\neg}(P)$ is given by the set of the corresponding clauses:

$$B_1 \wedge \ldots \wedge B_m \wedge \mathcal{B}\neg C_1 \wedge \ldots \wedge \mathcal{B}\neg C_n \supset A_1 \vee \ldots \vee A_k \tag{1}$$

obtained by replacing the default negation, *not* F, by the belief atom, $\mathcal{B}\neg F$, and by replacing the rule symbol $\rightarrow$ by the standard material implication $\supset$.

The translation, $T_{\mathcal{B}\neg}(P)$, gives therefore the following meaning to the default negation:

$$not\, F \;\stackrel{def}{\equiv}\; \mathcal{B}\neg F \;\equiv\; F \text{ is believed to be false } \;\equiv\; \neg F \text{ is minimally entailed.} \tag{2}$$

Theorem 2.3 (Embeddability of Stationary and Stable Semantics) *[Prz97a] There is a one-to-one correspondence between stationary (or, equivalently, partial stable) models $\mathcal{M}$ of the program P and consistent static autoepistemic expansions $T^\diamond$ of its translation $T_{\mathcal{B}\neg}(P)$ into a belief theory. Namely, for any objective atom A we have:*

$$A \in \mathcal{M} \quad \textit{iff} \quad A \in T^\diamond \quad \textit{iff} \quad \mathcal{B}A \in T^\diamond$$
$$\neg A \in \mathcal{M} \quad \textit{iff} \quad \mathcal{B}\neg A \in T^\diamond.$$

In particular, the well-founded model $\mathcal{M}_{WF}$ of the program P corresponds to the least static expansion of $T_{\mathcal{B}\neg}(P)$. Moreover, stable models (or answer sets) $\mathcal{M}$ of P correspond to those consistent static autoepistemic expansions $T^\diamond$ of $T_{\mathcal{B}\neg}(P)$ that satisfy the condition that for all objective atoms A, either $\mathcal{B}A \in T^\diamond$ or $\mathcal{B}\neg A \in T^\diamond$, i.e., that every atom is either believed or disbelieved in $T^\diamond$. □

Example 2.2 It is easy to see that the belief theory T considered in Example 2.1 can be viewed as a translation $T_{\mathcal{B}\neg}(P)$ of the stratified logic program P given by:

$$\begin{array}{lcl} Runs & \leftarrow & notBroken \\ Fix & \leftarrow & notRuns. \end{array}$$

The unique consistent static expansion,

$$T^{\diamond} = Cn_{AEB}(T \cup \{\mathcal{B}\neg Broken, Runs, \mathcal{B}\neg Fix, \})$$

of T corresponds therefore to the unique perfect model,

$$M = \{not\ Broken, Runs, not\ Fix\}$$

of P, which is also its unique well-founded and stable model [Prz90]. □

3 Strong Negation

In the Introduction we concluded that in addition to default negation, *not* F, non-monotonic reasoning requires a new type of negation which is similar to classical negation $\neg F$, in the sense that it is not assumed by default, but does not satisfy the law of the excluded middle and ensures a symmetric treatment of positive and negative information by default negation. In this section we define the so called *strong negation*, $\sim F$, and argue that it is a natural candidate for such a negation. We study properties of strong negation and its relationship to default negation and classical negation. We also illustrate how strong negation can be applied to provide a natural solution to an important knowledge representation problem.

We introduce strong negation within the broad framework of the Autoepistemic Logic of Beliefs, AEB, described in Section 2. As a result, the definition of strong negation applies to all non-monotonic formalisms embeddable into AEB, including circumscription, autoepistemic logic and all the major semantics recently proposed for logic programs. The definition is patterned after the original definition given in [GL90] and therefore, when applied to logic programs with stable semantics, strong negation coincides with Gelfond-Lifschitz's so called "classical negation" .

Strong negation is introduced to belief theories T in the Autoepistemic Logic of Beliefs, AEB, by:

(1) augmenting the underlying objective language $\mathcal{L}$ with a set $\widehat{\mathcal{S}} = \{\sim A : A \in \mathcal{S}\}$ of new *objective* propositional symbols, called *strong negation atoms*, where $\mathcal{S}$ is any fixed set of propositional symbols from $\mathcal{L}$. As a result we obtain an extended objective language $\widehat{\mathcal{L}}$ and an extended language of beliefs $\widehat{\mathcal{L}}_{AEB}$.

(2) adding to the belief theory T the following *strong negation constraint*:

$$(S_A) \qquad A \wedge \sim A \supset \bot \quad \text{or, equivalently,} \quad \sim A \supset \neg A,$$

for every strong negation atom $\sim A$ in $\widehat{\mathcal{S}}$.

The strong negation constraint S_A states that A and $\sim A$ cannot be both true and thus ensures that the intended meaning of strong negation $\sim A$ is "*$\sim A$ is the opposite of A*". For example, a proposition A may describe the property of being *"qualified"* while the proposition $\sim A$ describes the property of being *"unqualified"*. The strong negation constraint states that a person cannot be *both* qualified and unqualified. We do not assume, however, that everybody is already known to be either qualified or unqualified.

In practice, given a theory T, the set $\widehat{\mathcal{S}}$ is usually defined as $\widehat{\mathcal{S}} = \{\sim A : \sim A \text{ occurs in } T\}$. As a result, theories which do not contain strong negation atoms $\sim A$ are not affected by the strong negation constraints in any way. We want to emphasize that rather than introducing strong negation at the *meta-level*, by including strong negation constraints in the logical closure operator Cn_{AEB}, we instead make strong negation part of the *object-level* language, by making the constraints part of any belief theory in which the corresponding atoms $\sim A$ occur. In the sequel, we will implicitly assume that the strong negation constraint S_A is implicitly included in any belief theory T which contains the atom $\sim A$.

For the sake of readability, instead of using somewhat cumbersome names, like $\sim good\_teacher$, to denote the strong negation of the predicate $good\_teacher$, we use more readable names, such as $bad\_teacher$. Furthermore, even though we only consider propositional languages, we often use variables as a shortcut for all ground instantiations of a given formula.

Example 3.1 Consider a university that periodically evaluates faculty members based on their research and teaching performance. Faculty members who are known to be strong researchers and who are believed to be good teachers (as we all know, it is difficult to objectively evaluate teaching) receive positive evaluation. On the other hand, those who are believed not to be strong researchers as well as those that are believed to be poor teachers receive negative evaluation. Anyone who received a teaching award is considered to be a good teacher and anyone who published at least 10 reviewed papers during the evaluation period is considered a good researcher. The individuals whose evaluation status is not yet determined undergo some further (unspecified) review process. This leads us to the following belief theory in AEB:

$$\begin{array}{l} good\_researcher(x) \wedge \mathcal{B}\, good\_teacher(x) \supset good\_evaluation(x) \\ \mathcal{B}\neg good\_researcher(x) \supset bad\_evaluation(x) \\ \mathcal{B}\, bad\_teacher(x) \supset bad\_evaluation(x) \\ teaching\_award(x) \supset good\_teacher(x) \\ many\_publications(x) \supset good\_researcher(x). \end{array}$$

The predicates $bad\_teacher$, $bad\_evaluation$ and $bad\_researcher$ are intended to represent strong negation of the predicates $good\_teacher$, $good\_evaluation$ and $good\_researcher$, respectively, and, therefore, we need to add to our theory strong negation constraints (S_A) for each one of them:

$$\begin{array}{l} good\_researcher(x) \wedge bad\_researcher(x) \supset \bot \\ good\_teacher(x) \wedge bad\_teacher(x) \supset \bot \\ good\_evaluation(x) \wedge bad\_evaluation(x) \supset \bot. \end{array}$$

Suppose that both Ann and Tom published at least 10 papers. Suppose, further, that Ann is a good teacher, both Mary and Keith received teaching awards but Tom is considered to be a bad teacher. Moreover, Keith and Paul have a lot of joint publications so at least one of them must be a good researcher:

$$\begin{array}{l} many\_publications(Ann) \\ many\_publications(Tom) \\ good\_teacher(Ann) \\ teaching\_award(Mary) \\ teaching\_award(Keith) \\ bad\_teacher(Tom) \\ good\_researcher(Keith) \vee good\_researcher(Paul). \end{array}$$

The resulting belief theory has one consistent static expansion in which Ann receives a positive evaluation because she is a good researcher and is believed to be a good teacher. Tom receives a negative evaluation because even though he is a good researcher, he is also believed to be a bad teacher. Mary receives a negative evaluation as well because even though she is a good teacher, there is no evidence of good research work in her file (i.e., $\mathcal{B}\neg good\_researcher(Mary)$ holds).

Keith's and Paul's status is not yet clear and thus they will have to be further reviewed. Indeed, none of them individually has been shown to be a strong researcher so they are not eligible for positive evaluation. At the same time none of them is believed to be a bad teacher and there is some evidence supporting each one of them being a good researcher (i.e., neither $\mathcal{B}\neg good\_researcher(Paul)$ nor $\mathcal{B}\neg good\_researcher(Keith)$ is true) and therefore they are not subject to negative evaluation, either.

Observe, however, that if we later find out that Paul is in fact a lousy researcher:

$$bad\_researcher(Paul)$$

then Paul will receive a negative evaluation and we will conclude (by virtue of the strong negation constraint) that Keith is a good researcher and thus Keith will receive a positive evaluation. □

Remark 3.1 The correctness of the above example has been verified by using the prototype *interpreter for static semantics* developed by Stefan Brass and based on the characterizations of static semantics obtained in [BDP96]. The interpreter is available from `ftp://ftp.informatik.uni-hannover.de/software/static/static.html` via FTP and WWW. □

Even though the strong negation operator $\sim F$ is defined so far only for atomic formulae F of the language $\mathcal{L}$, we can easily extend it to arbitrary formulae F of the extended language $\widehat{\mathcal{L}}$. This is accomplished by adding the following axiom schemas for all formulae F and G of the language $\widehat{\mathcal{L}}$:

$$\begin{array}{lcl} \sim(\sim F) & \equiv & F \\ \sim(\neg F) & \equiv & \neg(\sim F) \\ \sim(F \vee G) & \equiv & \sim F \wedge \sim G \\ \sim(F \wedge G) & \equiv & \sim F \vee \sim G. \end{array}$$

Using the above axioms, strong negation $\sim F$ of an arbitrary formula F of the language $\widehat{\mathcal{L}}$ can be shown to be equivalent to a formula of the language $\widehat{\mathcal{L}}$ itself. Moreover, if two formulae F and G of the language $\widehat{\mathcal{L}}$ are equivalent then so are their strong negations $\sim F$ and $\sim G$. Readers interested in the details of this extension of the strong negation operator $\sim F$ to arbitrary formulae F and in its properties should consult the paper [Prz97b]. In the remainder of this section, whenever the operator $\sim F$ is used with reference to a non-atomic formula F it is implicitly assumed that the above axiom schemas are part of the belief theory.

Remark 3.2 The underlying logic with strong negation, $\sim F$, considered in this paper does *not* coincide with the so called *"classical logic with strong negation"* [Vak77, PW90b], i.e., with classical logic augmented with Nelson's strong negation [Nel49]. The difference lies in the mutual relationship between strong and classical negations. In classical logic with strong negation, which is known to be equivalent to 3-valued Lukasiewicz's logic, the two negations $\neg$ and $\sim$ mutually cancel each other because of the axiom $\sim(\neg F) \equiv F$. On the other hand, the logic presented in this paper contains the axiom $\sim(\neg F) \equiv \neg(\sim F)$ ensuring that the two negations commute and thus are mutually symmetric.

The consequences of this seemingly small difference are profound. It is well-known that in the classical logic with strong negation two formulae F and G may be tautologically equivalent and yet their strong negations $\sim F$ and $\sim G$ not be equivalent. For example, A and $\neg\neg A$ are tautologically equivalent and yet $\sim A$ is not equivalent to $\sim(\neg\neg A)$ because the latter formula is equivalent to $\neg A$. This problem can be avoided only if $\sim A$ is always equivalent to $\neg A$, i.e., when strong negation actually coincides with classical negation. On the other hand, in the logic presented in this paper, one can prove that if two formulae F and G of the language $\widehat{\mathcal{L}}$ are equivalent then so are their strong negations $\sim F$ and $\sim G$. Readers should consult the paper [Prz97b] for a more detailed treatment of this subject. In there the strong negation operator is also extended to all formulae of the modal language with beliefs $\widehat{\mathcal{L}}_{AEB}$. □

The above extension preserves the basic property of strong negation, namely, the fact that it is stronger than classical negation. However, this is only true for *positive formulae*, i.e., those that do not use classical negation $\neg$.

Proposition 3.1 (Strong Negation vs. Classical Negation) *Suppose that T is a belief theory and F is a positive formula of $\widehat{\mathcal{L}}$. Then:*

$$T \models (\sim\!F \supset \neg F).$$

Proof. If $F = A$ is atomic then, due to the strong negation constraints, $(\sim\!F \supset \neg F) \in T$. Suppose that the claim is true for positive formulae G and H.

If $F = \sim\!G$ than $T \models (\sim\!G \supset \neg G)$ and therefore $T \models (G \supset \neg\sim\!G)$ which implies that $T \models (\sim\!F \supset \neg F)$.

If $F = G \vee H$ then $T \models (\sim\!G \supset \neg G)$ and $T \models (\sim\!H \supset \neg H)$. Since $T \models \sim\!F \equiv (\sim\!G \wedge \sim\!H)$, we obtain that $T \models (\sim\!F \supset (\neg G \wedge \neg H)$ and therefore $T \models (\sim\!F \supset \neg F)$. The proof for conjunction is completely analogous. □

From the above Proposition one easily derives an important relationship between the strong negation operator and the belief operator:

Proposition 3.2 (Strong Negation vs. Beliefs) *Suppose that T is a belief theory. All static autoepistemic expansions $T^\diamond$ of T are closed under the inference rule:*

$$\frac{\sim\!F}{\mathcal{B}\neg F}$$

where F is any positive formula of $\widehat{\mathcal{L}}$. In particular, this is true for the least static expansion of T.

Proof. By the previous Proposition, if $\sim\!F$ belongs to $T^\diamond$ then so does $\neg F$. By Necessitation, also $\mathcal{B}\neg F$ belongs to $T^\diamond$. □

This result confirms the intuitively desirable fact that (positive) formulae F whose strong negation $\sim\!F$ is known to hold are also believed to be false. Observe that for any positive formula F of $\widehat{\mathcal{L}}$, both F and $\sim\!F$ are assumed false by default, i.e., both $\mathcal{B}\neg F$ and $\mathcal{B}\neg(\sim\!F)$ are true, because, in the absence of any other information, F and $\sim\!F$ are false in all minimal models. This is the intended behavior for symmetric negation.

Remark 3.3 However, whenever needed, for any given positive formula F, one can easily ensure that one of the formulae, F or $\sim\!F$, is *true by default*, and thus that only the other is false by default. This can be accomplished by assuming one of the following *default axioms*:

$$\mathcal{B}\neg F \supset \sim\!F \qquad \text{or} \qquad \mathcal{B}\neg(\sim\!F) \supset F$$

which say that if we disbelieve F (respectively, $\sim\!F$) then $\sim\!F$ (respectively, F) can be assumed to be true. For example, the first default axiom causes $\sim\!F$ to be true by default thus forcing strong negation to behave in a way that is similar (but not identical) to classical negation. This will prove useful later on when we discuss the application of strong negation to theory and interpretation update. Moreover, assuming both by default makes them both undefined, in the absence of other information.

One can also *prevent* any minimization of F and $\sim\!F$ by assuming the *law of the excluded middle*, $F \vee \sim\!F$, for a given formula F, which causes precisely one of F or $\sim\!F$ to be true at all times. □

As we mentioned in the Introduction, non-monotonic formalisms based on some form of *predicate minimization*, such as *CWA*, circumscription and most semantics of logic programs, do not treat atomic formulae A and their classical negations $\neg A$ symmetrically and thus they are *not invariant* under a simple renaming substitution replacing atoms by their negations and vice versa.

The Autoepistemic Logic of Beliefs, a superset of such formalisms, naturally suffers from the same problem. For example, the belief theory $\mathcal{B}\neg A \supset C$ has a unique static expansion in which C is true, and, yet, after substituting A' for $\neg A$, the resulting theory $\mathcal{B}A' \supset C$ has a unique expansion which no longer contains C. However, since strong negation atoms $\sim\!A$ are introduced as regular objective atoms and since renaming of atoms has no effect on the semantics, we immediately conclude that beliefs in *AEB* are invariant under renaming of any predicates from A to $\sim\!A$ and vice versa.

Proposition 3.3 (Strong Negation is Symmetric) *Static semantics of belief theories in AEB is invariant under the renaming of any predicates from A to* $\sim\!A$ *and vice versa.*

More precisely, suppose that S *is a subset of the set of all objective predicates from* $\mathcal{L}$ *and let* Θ *be the operator simultaneously replacing all occurrences of the atom* $\sim\!A$ *by* A *and all occurrences of the atom* A *by* $\sim\!A$*, for all atoms* A *in* S*. Then* $T^\diamond$ *is a static expansion of a belief theory* T *in* AEB *if and only if* $\Theta(T^\diamond)$ *is a static expansion of the theory* $\Theta(T)$*.* □

The above simple proposition illustrates the most basic difference between classical and strong negation.

3.1 Strong Negation and Logic Programming

Since logic programs under major semantics, including *stable semantics* [GL88], *well-founded semantics* [VGRS90], *partial stable* or *stationary semantics* [Prz90] and *static semantics* [Prz95b], can be translated into belief theories in *AEB* via the embedding defined in Section 2.3 [Prz95b, Prz95a], the introduction of strong negation into belief theories immediately introduces strong negation into logic programs with these semantics.

In particular, it follows from [GL90, Prz90] that stable semantics augmented with strong negation is equivalent to stable semantics with the so called "classical negation", originally introduced by Gelfond and Lifschitz [GL90]. We assume here that "classical negation" of an atom A is translated into its strong negation $\sim\!A$.

Theorem 3.1 (Strong Negation Extends "Classical" Negation) *There is a one-to-one correspondence between stable models* $\mathcal{M}$ *of a logic program* P *with "classical negation" and consistent static autoepistemic expansions* $T^\diamond$ *of its translation* $T_{\mathcal{B}\neg}(P)$ *into belief theory that satisfy the condition* $\mathcal{B}A \vee \mathcal{B}\neg A$*, for all objective atoms* A*.* □

From Proposition 3.2 and the fact that default negation, *not* F is translated into $\mathcal{B}\neg F$, one immediately derives an important relationship between strong negation and default negation in logic programs:

Proposition 3.4 (Strong Negation vs. Default Negation) *Suppose that* P *is a logic program. The inference rule:*

$$\frac{\sim\!F}{not\ F}$$

is satisfied by all semantics of P *obtained by the embedding of* P *into* AEB *via the translation* $T_{\mathcal{B}\neg}(P)$ *and for all positive formulae* F *of* $\widehat{\mathcal{L}}$*.* □

Remark 3.4 It is also worth noting that, for atomic objective formulae A, the default axioms discussed above have a particularly simple translation into logic programming rules:

$$\sim\!A \leftarrow not\ A \quad \text{and} \quad A \leftarrow not\ \sim\!A.$$

□

3.2 Application of Strong Negation to Knowledge Representation

Applications of various forms of *symmetric negation* to knowledge representation have been studied in a number of papers and proved to be quite extensive (see e.g. [APP96, GL92, Kow90], [Kow92, PAA93, PDA93a, PDA93b, Prz97a]). In particular, in [Prz97a] strong negation is used to provide a simple embedding of Gelfond's epistemic specifications [Gel92] into *AELB* and in [APP96] strong negation is applied to propose a solution to the problem of belief revision. Here we show how one can apply strong negation to provide a natural solution to an important knowledge representation problem, namely, to the problem of *theory update*. This generalizes to arbitrary belief theories the approach to theory update proposed earlier in [MT94b, PT95].

Katsuno and Mendelzon [KM91] have distinguished two abstract frameworks for reasoning about change: *theory revision* and *theory update*. As opposed to theory revision, which involves a change in knowledge or belief with respect to a static world, theory update involves a change of knowledge or belief in a changing world. Winslett [Win88] showed that reasoning about actions should be done "one model at a time." That is, when reasoning about the outcome of an action, we must consider its effect in each one of the states of the world that are consistent with our (possibly incomplete) knowledge of the current state of the world. This insight is reflected in the general definition of theory update due to Katsuno and Mendelzon, which can be formulated as follows.

Let T and S be sets of propositional formulae. A set S' of formulae is a "theory update" of S by T if

$$\mathrm{Models}(S') = \{\, I' : \exists I \in \mathrm{Models}(S) \;.\; I' \text{ is ``an update of } I \text{ by } T\text{''} \,\}.$$

From this definition we infer that in order to determine "theory update" it suffices to define when an interpretation I' is an update of an interpretation I by a theory T. Below we illustrate how strong negation can be used to define interpretation update by an arbitrary belief theory T in *AEB*. As we already mentioned, this is a generalization to arbitrary belief theories of the approach to interpretation update proposed earlier in [MT94b, PT95].

Suppose that T is the theory discussed in Example 3.1 describing a university that periodically evaluates its faculty based on their research and teaching performance and consisting of formulae:

$$\begin{array}{l}
good_researcher(x) \wedge \mathcal{B}\, good_teacher(x) \supset good_evaluation(x)\\
\mathcal{B}\neg good_researcher(x) \supset bad_evaluation(x)\\
\mathcal{B}\, bad_teacher(x) \supset bad_evaluation(x)\\
teaching_award(x) \supset good_teacher(x)\\
many_papers(x) \supset good_researcher(x)
\end{array}$$

together with strong negation constraints for all the atoms appearing in the theory, e.g.:

$$good\_researcher(x) \wedge bad\_researcher(x) \supset \bot, \ldots,\ etc.$$

Moreover, suppose that I is an interpretation of the objective language $\widehat{\mathcal{L}}$ given by:

$$I = \{good\_evaluation(Jack), good\_evaluation(Lisa), good\_teacher(Scott)\} \cup$$

$$\cup\ \{many\_papers(Lisa), teaching\_award(Jack), many\_papers(Scott)\}.$$

In order to produce an interpretation I' of the objective language $\widehat{\mathcal{L}}$, which can be considered an update of I by the belief theory T, we first augment T with the default axioms $\mathcal{B}\neg(\sim A) \supset A$, for every atom A in I, e.g.:

$$\mathcal{B}\neg bad\_evaluation(Jack) \supset good\_evaluation(Jack), \ldots,\ etc.$$

These axioms can be viewed as natural inertia or *frame axioms* stating that predicates that are initially true in I remain true unless we have some evidence to the contrary. We will denote the set of these default axioms by D_I.

The augmented belief theory $T_I = T \cup D_I$ has a unique static expansion in which Jack receives a negative evaluation because he does not have any research record and Scott receives a positive evaluation because he is a good teacher and a good researcher. Even though Lisa's teaching record is unclear she keeps her positive evaluation due to the default axiom:

$$\mathcal{B}\neg bad\_evaluation(Lisa) \supset good\_evaluation(Lisa).$$

It is therefore natural to consider the interpretation:

$$I' = \{bad\_evaluation(Jack), teaching\_award(Jack), good\_evaluation(Scott)\} \cup$$

$$\cup \ \{good\_researcher(Scott), good\_researcher(Lisa), good\_evaluation(Lisa)\} \cup$$

$$\cup \ \{good\_teacher(Scott), many\_papers(Scott), many\_papers(Lisa), good\_teacher(Jack)\},$$

consisting of all the atomic formulae from the language $\widehat{\mathcal{L}}$ that belong to the unique static expansion $T_I^\diamond$ of T_I, to be the *update of I by T*. In general, we may have more than one static expansion of T_I thus resulting in more than one possible interpretation update of I.

It follows from the results proved in [PT95] that *revision programming*, introduced in [MT94b], is a special case of this approach obtained when the theory T is a *positive logic program*. However, the approach proposed in here is more general because the updating theory T can be an *arbitrary belief theory* in AEB. In particular, T can be an arbitrary propositional theory or it can represent any of the non-monotonic formalisms embeddable into AEB, such as CWA, circumscription, autoepistemic logic and all the major semantics recently proposed for logic programs. In [PT95] it was also shown how to define update in the framework of default logic.

4 Explicit Negation

In this section we introduce an alternative weaker notion of symmetric negation in the Autoepistemic Logic of Beliefs, which we call *explicit negation*. The resulting extension of AEB, called the *Autoepistemic Logic of Beliefs with Explicit Negation*, $AEBX$, demonstrates the flexibility of the AEB framework in expressing different forms of negation.

After providing motivation and formal definitions, we show that static expansions in $AEBX$ are sufficiently expressive to characterize the *well-founded semantics with explicit negation*, $WFSX$, a semantics of logic programs introduced earlier in [PA92]. We then contrast explicit negation with strong negation. Section 5 illustrates an application of explicit negation showing how belief revision in $AEBX$ can directly capture the contradiction removal techniques used for logic programs [AP94].

4.1 The Issue of Relevance

In logic programming, clauses are seen as inference rules rather than material implications. However, for definite and normal programs this distinction is immaterial. Due to the absence of negative facts in such programs, the addition of a contrapositive $\neg Head \rightarrow \neg Body$ of a program rule $Head \leftarrow Body$ has no bearing on the rest of the program.

On the other hand, in normal logic programs extended with a symmetric negation (hereafter, simply called *extended logic programs* or ELPs) some care is needed if one wants to preserve the procedural reading of logic program rules. According to this reading, as in a procedure, the truth value of the head of a rule is solely determined by the truth value of its body. Thus, the procedural reading indicates that computing the truth value of a literal can be done by relying solely on the procedural call graph implicitly defined by the rules for literals. In other words, literals that are not (transitively) called by the rules for another literal should not influence its truth value. This well-known property, called *relevance* [Dix92], is essential to guarantee the availability of (strictly) top-down evaluation procedures for a semantics. It is worth noting that the well-founded semantics for normal logic programs [GRS91] obeys relevance (cf. [Dix92]). On the other hand, neither the Stable Models Semantics, nor AEB when applied to theories resulting from logic programs with the strong negation constraints, satisfy this principle.

One paramount motivation for introducing $AEBX$ is the desire to capture those semantics of logic program that satisfy relevance and thus permit efficient, top-down implementations. As shown in Example 4.1, queries for non-relevant semantics cannot, in general, be evaluated in a top-down fashion using standard logic programming implementation procedures, i.e., by simply following the program's call graph. The other was the desire to ensure that $AEBX$ is more expressive than AEB by providing a meaningful semantics to those consistent programs that appear to have a well-defined intended meaning and yet do not have a consistent semantics when strong negation is used (see Example 4.2).

Example 4.1 Take the following theory T and corresponding logic program P, where $T = T_{\mathcal{B}\neg}(P)$:

$$
\begin{array}{ll}
\mathcal{B}\neg{-god\_exists} \supset god\_exists & god\_exists \leftarrow not\ {-god\_exists} \\
\mathcal{B}\neg god\_exists \supset {-god\_exists} & {-god\_exists} \leftarrow not\ god\_exists \\
\mathcal{B}\neg god\_exists \supset {-go\_to\_church} & {-go\_to\_church} \leftarrow not\ god\_exists \\
go\_to\_church & go\_to\_church
\end{array}
$$

where the first two rules represent two conflicting default axioms, which read "I conclude God exists if I disbelieve Its non-existence" and "I conclude God doesn't exist if I disbelieve Its existence". The third rule states that "If I disbelieve the existence of God then I do not go to church", and the fourth that "I go to church".

If $-$ is understood as strong negation, i.e. if strong negation constraints are added to T, then the theory has one expansion, where god_exists because I go to church. Indeed, $go\_to\_church \supset \neg{-go\_to\_church}$ (strong negation constraint), which by contraposition of the third clause entails $\neg\mathcal{B}\neg god\_exists$. Therefore, since $-god\_exists$ only appears in the second clause, $\neg{-god\_exists}$ holds in all minimal models, and by necessitation rule (N), $\mathcal{B}\neg{-god\_exists}$. Hence, by the first clause of T, god_exists holds in all expansions of the theory.

Note that this conclusion is not *relevant*. Looking at the program P makes it clear that only the first two rules are (transitively) called by god_exists. However, the reader may check that the theory consisting solely of the first two clauses, does not have god_exists in all its expansions.

With explicit negation (to be defined below) we have different conclusions, and the corresponding least expansion includes $\{go\_to\_church, \mathcal{B}\neg{-go\_to\_church}\}$ but not god_exists nor $\mathcal{B}\neg{-god\_exists}$ (cf. Example 4.3). □

Example 4.2 Take now the following theory and corresponding logic program:

$$
\begin{array}{ll}
\mathcal{B}\neg shave(x,x) \supset shave(John,x) & shave(john,X) \leftarrow not\ shave(X,X) \\
shave(y,x) \supset go\_dine\_out(x) & go\_dine\_out(X) \leftarrow shave(Y,X) \\
{-shave(Peter,Peter)} & {-shave(peter,peter)} \\
{-go\_dine\_out(John)} & {-go\_dine\_out(john)}
\end{array}
$$

The first rule states that "John shaves everyone not believed to shave themselves". The second says that "If x has been shaved (by anyone) then x will go out to dine". The third states that "Peter does not shave himself", and the fourth that "John has not gone out to dine". We would like to know whether we believe John has shaved himself given that he has not gone out to dine. Note that believing he has not shaved himself leads to a contradiction, and that the conclusion that he has shaved himself is not true in all minimal models.

If the strong negation constraints are added to this theory then AEB assigns no consistent meaning to it (there is no consistent expansion). On the contrary, if explicit negation, to be defined below, is used instead the theory has one expansion that includes:

$\{-go\_dine\_out(john), \mathcal{B}\neg go\_dine\_out(john)\}$ but not $shave(john, john)$. The absence of consistent expansions by using strong negation is brought about by the use of the contrapositive $\neg go\_dine\_out(X) \supset \neg shave(X, X)$ to conclude $\neg shave(john, john), \mathcal{B}\neg shave(john, john)$, and hence $shave(john, john)$ (contradiction). □

As we have seen, the reason strong negation does not directly capture any semantics for extended logic programs complying with relevance, such as *WFSX*, is because of its very definition. Strong negation constraints, $-A \supset \neg A$, state that strongly negated facts or conclusions entail classically negated ones, thereby permitting the use of the contrapositives of the material implications resulting from the translation of the logic program rules.

What the above two paradigmatic examples have in common is the back propagation of truth values by strong negation, against the logic program rule arrow, into a loop of otherwise undefined literals (i.e., such that neither L nor *not* L hold). In the Example 4.1 we have an even loop over default negation, and in the Example 4.2 an odd one. In the first example, the back propagation decides the loop one way, and in the second it comes up against the impossibility of resolving the loop by imposing a truth value. However, as we shall see below, explicit negation does not affect either loop.

To conclude, because of its use of classical negation contrapositives, strong negation leads both to logic programs without a semantics and to logic program theories with unwarranted (non-relevant) conclusions, i.e. conclusions not solely based on the procedural call graph of the logic program. To be able to capture relevant logic program semantics a weaker notion of symmetric negation is needed. Theories with explicit negation which are translatable into extended logic programs can be efficiently queried by the top-down procedural implementation technology of logic programs.

4.2 Introducing Explicit Negation

Explicit negation is added to *AEB* by means of an explicit negation operator. For the sake of simplicity, and since we will never consider theories with both explcit and strong negation, we use the same negation symbol for both. We thus defining the *Autoepistemic Logic of Beliefs with Explicit Negation*, *AEBX*. Specifically, this is accomplished by:

- Augmenting the underlying objective language $\mathcal{L}$ with a set $\widehat{\mathcal{S}} = \{\overline{A} : A \in \mathcal{S}\}$ of new *objective* propositional symbols, called *dual orexplicit negation atoms*, where $\mathcal{S}$ is any fixed set of propositional symbols from $\mathcal{L}$. As a result we obtain an extended objective language $\widehat{\mathcal{L}}$ and an extended language of beliefs $\widehat{\mathcal{L}}_{AEB}$. The extension of explicit negation to arbitrary positive objective formulae can be done in the same way as for strong negation.

- Extending the logical closure operator Cn_{AEB} with the following *Coherence* inference rule

$$\frac{\overline{A}}{\mathcal{B}\neg A}$$

  for every objective propositional symbol A[5].

A Coherence inference rule says that if one derives the dual, one has to believe its negation, i.e. "$\overline{A}$ *serves as evidence against* A". Since the Coherence inference rules have no effect on belief theories that do not include explicit negation atoms, in the sequel we will assume them as part of the operator Cn_{AEB} without further mention, whenever explicit negation is used.

Example 4.3 The details of this example show the essence of how explicit negation treats both previous examples, and the way it differs from strong negation.

[5]If $A = \overline{F}$ then we have $\frac{F}{\mathcal{B}\neg\overline{\overline{F}}}$.

The theory

$$T^\diamond = Cn_{AEB}(T \cup \{\mathcal{B}gtc, \mathcal{B}\neg\overline{gtc}\})$$

is, with the obvious abbreviations, an expansion of the *AEBX* theory in Example 4.1 (where strong negation is replaced by explicit negation). Its minimal models are:

$$\begin{array}{lcl}
M_1 & = & \{\mathcal{B}gtc,\ \mathcal{B}\neg\overline{gtc},\ \ \mathcal{B}ge,\ \ \mathcal{B}\overline{ge},\ \neg ge,\ \neg\overline{ge},\ \neg\overline{gtc},\ gtc\} \\
M_2 & = & \{\mathcal{B}gtc,\ \mathcal{B}\neg\overline{gtc},\ \ \mathcal{B}ge,\ \mathcal{B}\neg\overline{ge},\ \ ge,\ \neg\overline{ge},\ \neg\overline{gtc},\ gtc\} \\
M_3 & = & \{\mathcal{B}gtc,\ \mathcal{B}\neg\overline{gtc},\ \mathcal{B}\neg ge,\ \ \mathcal{B}\overline{ge},\ \neg ge,\ \ \overline{ge},\ \ \overline{gtc},\ gtc\} \\
M_4 & = & \{\mathcal{B}gtc,\ \mathcal{B}\neg\overline{gtc},\ \mathcal{B}\neg ge,\ \mathcal{B}\neg\overline{ge},\ \ ge,\ \ \overline{ge},\ \ \overline{gtc},\ gtc\}
\end{array}$$

$\mathcal{B}\neg\overline{gtc}$ follows in $T^\diamond$ from gtc and the Coherence inference rule.

This example illustrates why explicit negation does not affect the theory's even loop and, for the same reason, why it does not affect the odd loop of Example 4.2. Indeed, in the general case, the explicit negation of the head of a program rule may be true even though its body is undefined (i.e., such that neither $\mathcal{B}Body$ nor $\mathcal{B}\neg Body$ hold in an expansion). In other words, explicit negation allows the overriding to false of a rule's head when its body is undefined. Because of this feature, there is no backward propagation of falsity of the head into the rule's body. On the other hand, when the rule's body is true, then its head must necessarily be true, which, however, represents a forward, rather than backwards, propagation of truth values.

Notice that an explicit negation model may have evidence for both *go_to_church* and $\overline{go\_to\_church}$. Not so for strong or classical negations. However, when there is overwhelming evidence for $\overline{L}$ (i.e. in all models), but not for L (i.e. only in some of the models), then explicit negation, via the Coherence inference rule, decides in favor of $\overline{L}$. And vice-versa. □

As noted above, the definition of explicit negation, contrary to that of strong negation, does not prevent the existence in a model of both A and $\overline{A}$, for some atom A. However, this kind of "paraconsistency" in models does not spill over to *AEBX* expansions:

Proposition 4.1 *Let $T^\diamond$ be a consistent expansion of a belief theory T in* AEBX. *Then, for no atom A:*

$$T^\diamond \not\models A \wedge \overline{A}$$

Proof. By contradiction, assume that $T^\diamond$ is a consistent expansion of T such that $T^\diamond \models A \wedge \overline{A}$, for a given arbitrary atom A. Since $T^\diamond$ is closed under the necessitation rule (N), and by hypothesis $A \in T^\diamond$, also $\mathcal{B}A \in T^\diamond$. Given that, by hypothesis, $\overline{A} \in T^\diamond$ and that $T^\diamond$ is, by definition of *AEBX*, closed under the Coherence axiom, also $\mathcal{B}A \in T^\diamond$. Thus, according to the distributive law for conjunctions mentioned in Section 2, $\mathcal{B}(A \wedge \neg A) \in T^\diamond$, i.e. $\mathcal{B}\bot \in T^\diamond$. Accordingly, given the consistency axiom (D), $T^\diamond$ is inconsistent (contradiction). □

In other words, if there is overwhelming evidence both for and against some atom A then there are no consistent expansions.

The following result shows that the relevant logic program semantics *WFSX* (defined in [PA92]) is embeddable in *AEBX*. This embeddability result requires, besides the translation defined in Section 2.3, a preliminary *WFSX*-semantics preserving transformation of the logic program. This transformation consists in the complete elimination of objective literal goals from rule bodies by means of unfolding, that is by successively replacing them by their various alternative rule definitions. The obtained programs are said to be in "semantic kernel form". Due to the length of the proof, and given that it requires a significant body of definitions and results not need elsewhere in this paper (such as the definitions of *WFSX* and of "semantic kernel transformation", plus the result showing that this transformation is *WFSX*-semantics preserving [AP96]), the proof of the following theorem is omitted here.

Theorem 4.1 (Explicit Negation and *WFSX***)** *Let P be an extended logic program in the semantic kernel form. There is a one- to-one correspondence between the partial stable models $\mathcal{M}$ of P, as defined in [PA92], and the consistent static autoepistemic expansions $T^\diamond$ of its translation $T_{\mathcal{B}\neg}(P)$ into an* AEBX *belief theory, where "explicit negation" of an atom A is translated into $\overline{A}$.*
□

Up to now, we have mainly considered *AEBX* theories resulting from the translation of (non-disjunctive) logic programs. This is so because we have motivated explicit negation by contrast with strong negation on such programs. Nevertheless, there is motivation to extend *AEB* with explicit negation to general theories, and not just logic programs.

The theory of Example 3.1, with strong negation replaced by explicit negation everywhere, illustrates such general use. In fact, the results will be the same, except when later one adds $\overline{good\_researcher(Paul)}$, expressing that there is evidence against Paul being a good researcher[6]. As shown in Example 3.1, by using strong negation, one additionally concludes that Paul will receive a negative evaluation, and that Keith is a good researcher and so receives a positive evaluation. When explicit negation is used instead, we still conclude that Paul will receive a negative evaluation, but we no longer surmise that Keith is a good researcher. Instead, a milder conclusion follows, that Keith is *believed to be* a good researcher. This conclusion is not enough to give Keith a good evaluation. See Example 4.4, for a detailed explanation of how these results are obtained.

4.3 Explicit and Strong Negation Compared

Explicit and strong negation share some similarities. The (derived) inference rule from strong negation to beliefs shown in Proposition 3.2, also holds for explicit negation. Explicit negation is in fact characterized by taking this inference rule as primitive, since it is no longer derivable in the absence of the strong negation axiom.

Another important similarity between both regards the symmetry between atoms and their negations. The result of Proposition 3.3 still holds if strong negation is replaced by explicit negation. In other words, explicit negation is also symmetric.

Because of the similarities, strong and explicit negation are even equivalent for the class of theories whose expansions capture the semantics of logic programs under Stable Models. Indeed, the result in Theorem 3.1 remains true regardless of whether "classical" negation is translated into strong or into explicit negation.

For logic programs, explicit negation can be seen as an approximation to strong negation, in the sense that any *relevant* consequence of a belief theory with strong negation remain true in *AEBX* after strong negation is replaced everywhere by explicit negation:

Proposition 4.2 *Let T_s be a AEB theory with strong negation obtained from an extended logic program P via the translation described in Section 2.3, T_x be the* AEBX *theory obtained from T_s by replacing strong by explicit negation and deleting all the strong negation constraints in T_s, and let $T_s^\diamond$ be an expansion of T_s. Then there exists an expansion $T_x^\diamond$ of T_x such that, for every objective atom A:*

- $\mathcal{B}A \in T_s^\diamond$ *if and only if* $\mathcal{B}A \in T_x^\diamond$
- $\mathcal{B}\neg A \in T_s^\diamond$ *if and only if* $\mathcal{B}\neg A \in T_x^\diamond$ □

Proof. It is easy to check that, for theories resulting from (non-disjunctive) logic programs, $T_s^\diamond$ can be expressed as:

$$T_s^\diamond = Cn_{AEB}(T_s \cup \{\mathcal{B}A : T_s^\diamond \models A\} \cup \{\mathcal{B}\neg A : T_s^\diamond \models_{min} \neg A\})$$

where A ranges over all objective atoms. Moreover:

[6]In Example 3.1 we added instead $bad\_researcher(Paul)$, standing for $\neg good\_researcher(Paul)$.

- $\mathcal{B}A \in T_s^\diamond$ if and only if $T_s^\diamond \models A$
- $\mathcal{B}\neg A \in T_s^\diamond$ if and only if $T_s^\diamond \models_{min} \neg A$

Thus, it is enough to prove that:

$$T_x^\diamond = Cn_{AEBX}(T_s \cup \{\mathcal{B}A : \mathcal{B}A \in T_s^\diamond\} \cup \{\mathcal{B}\neg A : \mathcal{B}\neg A \in T_s^\diamond\})$$

is an expansion of T_x, i.e.

$$T_x^\diamond = Cn_{AEBX}(T_x \cup \{\mathcal{B}A : T_x^\diamond \models A\} \cup \{\mathcal{B}\neg A : T_x^\diamond \models_{min} \neg A\})$$

where Cn_{AEBX} is the closure operator with the Coherence inference rules.

Let $T_x' = T_x \cup \{\mathcal{B}A : \mathcal{B}A \in T_s^\diamond\} \cup \{\mathcal{B}\neg A : \mathcal{B}\neg A \in T_s^\diamond\}$, and let T_s' be obtained from T_x' by adding the strong negation constraints. Clearly, $T_s^\diamond = Cn_{AEB}(T_s')$ and, by Proposition 3.2, $T_s^\diamond = Cn_{AEBX}(T_s')$. We now prove that the minimal models of T_s' exactly coincide with the minimal models of T_x', thus proving that, given that $Cn_{AEBX}(T_s')$ is an expansion, $Cn_{AEBX}(T_x') = T_x^\diamond$ is an expansion of T_x.

Let $M_1, \ldots, M_n, \ldots$ be all the minimal models of T_s'. Since $T_x' \subseteq T_s'$ all these models are models of T_x'. Moreover, since the strong negation constraints only delete models M such that, for some objective atom A, both A and $-A$ are true in M, then all the M_is are minimal models of T_x'.

It remains to be proven that T_x' has no other minimal models. This is proven in two steps: first we prove that (1) for a given fixed combination of belief propositions in some M_i, the only minimal model of T_x' is M_i; then we prove that (2) no new combinations of belief propositions are possible.

1. Since T_s' results from the translation of a non- disjunctive program, for a given fixed combination of belief propositions, there exists a single minimal model M_i. By removing the strong negation constraints, new models appear where, for some objective atom A, both A and $-A$ hold. These models differ from M_i at least in that one of A or $-A$ became true. Since T_x' resulted from the translation of a non-disjunctive logic program, the addition of atoms can only have as consequence the addition of new atoms[7]. Thus all these models are strictly greater than M_i, and so they are not minimal.

2. By contradiction, assume that T_x' has a minimal model N with a combination of belief propositions that does not exist in any of the M_is, i.e. N differs from any of the M_i in that there exists at least one literal L such that $\mathcal{B}L \in M_i$ and $\neg\mathcal{B}L \in N$, or $\neg\mathcal{B}L \in M_i$ and $\mathcal{B}L \in N$.

   - Assume that $\mathcal{B}L \in M_i$ and $\neg\mathcal{B}L \in N$. If additionally $T_s' \models_{min} L$ then, by construction of T_x', $\mathcal{B}L \in T_x'$, and N is not a model of T_x' (contradiction). Otherwise, i.e. if $T_s' \not\models_{min} L$, then there exists a minimal model of T_s' with $\neg\mathcal{B}L$, and so the combination of belief propositions in N is not new (contradiction).
   - Assume that $\neg\mathcal{B}L \in M_i$ and $\mathcal{B}L \in N$. If $T_s' \models_{min} L$ then, since T_s' is an expansion, all its minimal models have $\mathcal{B}L$, and so M_i is not a model of T_s' (contradiction). If $T_s' \models_{min} \neg L$ then, by construction of T_x', $\mathcal{B}\neg L \in T_x'$, and N is not a model of T_x' (contradiction). Otherwise, i.e. $T_s' \not\models_{min} L$ and $T_s' \not\models_{min} \neg L$, then there exists a minimal model of T_s' with $\mathcal{B}L$, and so the combination of belief propositions in N is not new (contradiction).

□

[7] Note that this is not the case for a disjunctive program where the addition of an atom might cause a new minimal model to appear. For example, the theory $T = \{-a, \;\; a \vee b, \;\; -a \supset \neg a\}$ has the minimal model $\{-a, \neg a, b\}$. By removing the strong negation constraint (the last clause), a new minimal model appears $\{-a, a, \neg b\}$. In this case the addition of a causes the deletion of b.

Corollary 4.1.1 (Explicit Negation Approximates Strong Negation) *Let T_s and T_x be as in Proposition 4.2. If some formula F of the form $\mathcal{B}L$, where L is either an objective atom A or its negation $\neg A$, holds in every expansion of T_x, then F also holds in every expansion of T_s.* □

Consequently, query evaluation procedures for explicit negation in logic programs can be used as sound query evaluation procedures for strong negation in logic programs[8]. However those procedures are not complete for strong negation since, as shown by Examples 4.1 and 4.2, the converse of Corollary 4.1.1 is not true in general.

Another result contrasting explicit and strong negation is:

Theorem 4.2 (Explicit Negation Extends Strong Negation) *There is a one to one correspondence between expansions of a theory T_s with strong negation and expansions of the* AEBX *theory T_x obtained by replacing in T_s strong by explicit negation, and by adding, for every atom A, the clause: $\overline{A} \supset \neg A$.* □

Proof. The only possible difference between expansions of T_x and those of T_s would be that the former are closed under Coherence inference rules. However, as shown by Proposition 3.2, all expansions of T_s are also closed under those rules. □

Disjunctive Syllogism is a derived inference rule applicable to classical and strong negations, but which is not enjoyed by explicit negation. This rule typically allows one to conclude A on the strength of $A \vee B$ and $\neg B$ (or $-B$).

Example 4.4 Consider the theory:

$$\overline{good\_researcher(Paul)}$$
$$good\_researcher(Keith) \vee good\_researcher(Paul)$$

stating that "there is evidence against Paul being a good researcher", and that "at least one of Keith or Paul is a good researcher".

Whereas with strong negation $good\_researcher(Keith)$ follows from the theory by virtue of the Disjunctive Syllogism, with explicit negation it is not so. Instead, a milder conclusion obtains:

First, note that $good\_researcher(Keith) \vee good\_researcher(Paul)$ is equivalent to the formula $\neg good\_researcher(Paul) \supset good\_researcher(Keith)$. Now, applying Necessitation, $\mathcal{B}(\neg good\_researcher(Paul) \supset good\_researcher(Keith))$. By then applying axiom **(K)**, we conclude $\mathcal{B}\neg good\_researcher(Paul) \supset \mathcal{B}good\_researcher(Keith)$. Since the premise of this implication follows from $\overline{good\_researcher(Paul)}$ by Coherence, we extract the milder conclusion $\mathcal{B}good\_researcher(Keith)$.

Intuitively, having evidence against Paul being a good researcher does not ensure that Paul is not a good researcher, and so does not warrant Keith being a good researcher. However, this information is enough for believing Paul is not a good researcher, and consequently to believe Keith is. Contrast this reading with the one of strong negation, where $-good\_researcher(Paul)$ means "Paul is a bad researcher". In this case, and knowing that either Paul or Keith is a good researcher, it is expected that Keith is a good researcher. □

5 Application of Explicit Negation

Applications of the logic programming semantics *WFSX* have been studied in various domains, including hypothetical reasoning [PAA93], model-based diagnosis [PDA93b], declarative debugging of logic programs [PDA93a], and updates [AP97]. Many of these applications require belief revision methods, via contradiction removal techniques [AP94]. Here we show how to capture belief revision

[8] In Section 6, the reader can find a brief discussion on the implementation of explicit negation.

in *AEBX* and we illustrate how to extend the above mentioned applications from the class of logic programs to the significantly broader class of belief theories.

Unlike normal programs, extended logic programs with symmetric negation may be contradictory, i.e., might not have a (consistent) semantics. While for some programs this seems reasonable (e.g., for a program with the two facts a and $\overline{a}$), for some others this may not be justified:

Example 5.1 Consider the program:

$$\begin{array}{rcl} runs & \leftarrow & not\ broken \\ \overline{runs} & & \end{array}$$

stating that "if we assume that a car is not broken, then it runs", and that "the car does not run".

None of the logic programming semantics mentioned in this paper assigns a consistent meaning to this program. Indeed, since there are no rules in the program for $broken$, all of the semantics assume $not\ broken$ true ($\neg broken$ is true in all minimal models of the corresponding AEB theory), and so both $runs$ and $\overline{runs}$ hold. If $\overline{runs}$ in the logic program is understood as the strong negation of $runs$ then, since $runs \wedge \neg runs \supset \bot$, no consistent expansion of the corresponding belief theory exists. The same happens if $\overline{runs}$ is understood as the explicit negation of $runs$, cf. Proposition 4.1.

But one can argue that, since the car does not run, it follows by "reductio ad absurdum" that our assumption that the car is not broken must be incorrect and thus has to be *revised.* □

The Contradiction Removal with Explicit Negation (*CRSX*) technique of [AP94] removes contradiction from logic programs under *WFSX* (wherever possible), by revising any default assumption that would otherwise lead to contradiction. To do so, it adds to programs a rule of the form

$$L \leftarrow not\ L$$

for each such assumption $not\ L$, with the effect that no model of the program can now contain $not\ L$. The alternative minimal contradicition removing sets of such "inhibition rules" can be added to the original program in order to obtain the alternative minimally revised programs.

Example 5.2 The only revision of the program in Example 5.1, is obtained by adding to it

$$broken \leftarrow not\ broken$$

The revised program is no longer contradictory: its *WFSX* is $\{\overline{runs}, not\ runs\}$. Furthermore, this addition is minimal. □

The addition of inhibition rules will be allowed for only some pre-designated set of literals – the *revisables*. These are the literals that can be independently revised. Other literals may have their truth value revised but only as a consequence of changes in the revisables. Restricting the revisables is crucial for controlling the causal level at which assumption withdrawal may be performed. Indeed, in an application domain such as diagnosis, revising the functional normality assumption about a component may be conditional upon revising the functional normality assumption about some subcomponent. Conversely, it may suffice to hypothesize the abnormality of a subcomponent to put in question the normality of the component containing it. We deal next with both issues.

Example 5.3 Consider now the (contradictory) program:

$$\begin{array}{rcl} runs & \leftarrow & not\ broken \\ \overline{runs} & & \\ broken & \leftarrow & flatTire \\ broken & \leftarrow & badBattery \end{array}$$

If addition of inhibition rules is allowed for any literal, then one such minimal addition removing the contradiction is $\{broken \leftarrow not\ broken\}$. This revision can be seen as a diagnosis of the car that simply states the car might be broken. However, in this case one would like the diagnosis to delve deeper into the car's functional structure, and obtain two minimal diagnoses: one suggesting a possible problem with a flat tire, and another suggesting a possible problem with a bad battery. This is achieved by declaring as revisables only $badBattery$ and $flatTire$. □

This declaration of revisables is akin to the declaration of abducible literals in abductive logic programming. There, a solution to an abductive query is obtained by minimally, and consistently, adding to the theory facts needed in order to prove the abductive query literal. Moreover, addition of facts is only allowed for literals declared abducibles[9].

Example 5.4 Consider the program obtained from Example 5.3 by removing the fact $\overline{runs}$, i.e. the program:

$$\begin{array}{rcl} runs & \leftarrow & not\ broken \\ broken & \leftarrow & flatTire \\ broken & \leftarrow & badBattery \end{array}$$

If every literal is abducible, then one abductive solution for $not\ runs$ is $\{broken\}$ (i.e., this is one possible explanation of the fact that the car is not running). To obtain only the deeper diagnosis of the car, simply declare as abducibles $\{flatTire, badBattery\}$: the abductive solutions to that query become then $\{flatTire\}$ and $\{badBattery\}$. □

This technique can easily be imported into *AEBX*. Define the revisions of a theory T which has no expansions (a *contradictory* theory), as the non-contradictory theories obtained by minimally adding clauses of the form

$$\mathcal{B}\neg L \supset L$$

which disallow (consistent) expansions in which $\mathcal{B}\neg L$ is true[10].

It is noteworthy to emphasize that, as opposed to logic programs, in *AEBX* there is no need for recourse to a meta-linguistic declaration of revisables. The language itself is sufficiently expressive to handle the definition of revisables. To realize this let us get back to the car example, which in *AEBX* is rendered as the belief theory T:

$$\begin{array}{l} \mathcal{B}\neg broken \supset runs \\ \overline{runs} \\ badBattery \supset broken \\ flatTire \supset broken \end{array}$$

where $broken$ is non-revisable. In *CRSX* this declaration of non-revisability means that $broken$ may have its truth value revised only indirectly, as a consequence of changes in the revisables, but never directly on account of the addition of an inhibition rule for it. In *AEBX* it means that beliefs about $broken$ may change only indirectly, as a consequence of changes in beliefs about revisables, but never directly because of the addition of an inhibition clause for it. To achieve this, we must require that belief changes about $broken$ be solely determined by the belief changes about the revisables, so that adding a single inhibition clause, for $broken$ only, is insufficient because the resulting theory has no consistent expansions.

By Necessitation and the axiom **(K)**, the closure of T contains:

$$\mathcal{B}flatTire \vee \mathcal{B}badBattery \supset \mathcal{B}broken.$$

[9] A formal comparison of contradiction removal and abduction in logic programs can be found in [DP95].

[10] An expansion with $\mathcal{B}\neg L$ would also contain L and $\mathcal{B}L$, since expansions are closed under necessitation, and would thus be inconsistent.

Thus, belief in the truth of $broken$ follows from belief in $flatTire$ or belief in $badBattery$. For the falsity of belief in $broken$ to be determined by the falsity of beliefs in $flatTire$ and $badBattery$ we need an additional statement, which ensures that if both $flatTire$ and $badBattery$ are disbelieved then $broken$ must be disbelieved as well, i.e.:

$$\mathcal{B}\neg flatTire \wedge \mathcal{B}\neg badBattery \supset \mathcal{B}\neg broken \tag{3}$$

Example 5.5 The belief theory T above, augmented with clause (3), has two revisions:

$$\begin{array}{l} T \ \cup \ \{\mathcal{B}\neg badBattery \supset badBattery\} \\ T \ \cup \ \{\mathcal{B}\neg flatTire \supset flatTire\} \end{array}$$

each corresponding to one of the desiredly deeper diagnoses.

Observe that $T' = T \cup \{\mathcal{B}\neg broken \supset broken\}$ is not a revision. Indeed, all minimal models of the theory have $\neg flatTire$ and $\neg badBattery$, because there are no clauses defined for neither $flatTire$ nor $badBattery$. Thus every expansion of T' must contain $\{\mathcal{B}\neg flatTire, \mathcal{B}\neg badBattery\}$. Now, by clause (3), every expansion of T' includes $\mathcal{B}\neg Broken$, and thus, by the inhibition clause for $broken$ and by Necessitation, is inconsistent. □

Notice the similarity between clause (3) and Clark's completion [Cla78] of $broken$. The latter implies that if both $flatTire$ and $badBattery$ are false then $broken$ is false, whilst (3) states the same about the corresponding beliefs. For this reason (3) is called the *belief completion clause* for $broken$. The formal definition of belief completion rules can be found in [APP96].

The specification of revisables is captured in the language of $AEBX$ by adding for each literal which is *not* a revisable the corresponding belief completion clause. This language level declarative specification achieves the same effect as the declaration of revisables for logic programs, which in the latter case can only be made meta-linguistically. Though not detailed here, it can also be used to explain the intuitive notion that a literal is abducible unless the rules defining it are closed by a completion clause. Moreover it has greater generality, by allowing conditions to be added to belief completion clauses. For example, instead of simply stating that $broken$ is not a revisable, we might want to say that $broken$ is not revisable only if the atom *deeper* is a consequence of the theory. In such a case we would just add:

$$deeper \wedge \mathcal{B}\neg flatTire \wedge \mathcal{B}\neg badBattery \supset \mathcal{B}\neg broken$$

This allows for flexible control over the level of revision: adding the fact *deeper*, or concluding it, results in only causally deeper faults being obtained. Otherwise superficial faults can be obtained. Diverse complex diagnosis preferences and strategies can be encoded in $AEBX$ (see [DNPS95] to find out how this can be accomplished by changing the revisables in logic programs).

We conclude by remarking that either one of symmetric negation in AEB can be used to obtain a solution to belief revision not relying on a theory transformation: instead of the addition of minimal sets of inhibition rules, alternative revisions can be obtained directly by modifying the definition of expansions, leading to the notion of *Careful Autoepistemic Expansions* (see [APP96]).

6 On the Implementation of Explicit Negation

As we argued above, one advantage of explicit negation is that theories that are translatable into logic programs can be implemented and queried by efficient (strictly) top-down methods based on standard logic programming techniques.

Top-down query evaluation procedures for explicit negation in extended logic programs can be obtained by adapting existing procedures for well-founded semantics of normal programs. One such adaptable derivation procedure is defined in [ADP94]. It relies on the definition of two kinds of derivation. One, called verity derivation, proving truth, and another proving non-falsity of literals (where non-falsity of L means that $not\ L \notin WFM$). Shifting from one kind of derivation to the other is required when trying to prove a default literal $not\ L$: the verity derivation of $not\ L$ succeeds if-and-only-if the non-falsity derivation of L fails; the non-falsity derivation of $not\ L$ succeeds if-and-only-if the verity derivation of L fails. Apart from the switch between derivation types at default literals, the derivation procedure relies on the usual SLD-like rewritting techniques.

Two adaptations are required to incorporate explicit negation (described in [ADP94], where soundness and completeness wrt. *WFSX* is proven). The first incorporates one extra method for deriving truth of a default literal $not\ L$ – the verity derivation of $not\ L$ succeeds, additionally, if the verity derivation of $\overline{L}$ succeeds – and the second restricts non-falsity derivations of objective literals – a non-falsity derivation of L fails if a verity derivation of $\overline{L}$ succeeds. The first adaptation directly reflects the Coherence inference rule that defines explicit negation. Also as a consequence of this rule, if some $\overline{L}$ is true then $not\ L$ must be also true and L cannot be false. This justifies the failure required by the second adaptation in non-falsity derivations.

7 Conclusion

We have shown that, in order to represent and reason about knowledge, one requires a form of negation which is not automatically assumed by default and which is symmetric with respect to default negation. Neither classical negation nor default negation satisfy these coonditions.

We then introduced, within the broad framework of Autoepistemic Logic of Beliefs, two types of symmetric negation, strong and explicit, and illustrated their application on a number of knowledge representation examples.

Although similar, strong and explicit negation differ in their use of contrapositives and their amenability to strictly top-down querying procedures. They are alike in that they both capture the answer-sets semantics [GL90], differ from classical negation, and enjoy symmetry. Of the two, explicit negation is weaker and easier to implement. It can be viewed and used as an approximation to strong negation.

Acknowledgments

The authors are grateful to D. Vakarelov and D. Pearce for their helpful comments.

References

[ADP94] J. J. Alferes, C. V. Damásio, and L. M. Pereira. SLX – A top-down derivation procedure for programs with explicit negation. In M. Bruynooghe, editor, *International Symposium on Logic programming*. MIT Press, 1994.

[AP92] J. J. Alferes and L. M. Pereira. On logic program semantics with two kinds of negation. In K. Apt, editor, *Int. Joint Conf. and Symp. on LP*, pages 574–588. MIT Press, 1992.

[AP94] J. J. Alferes and L. M. Pereira. Contradiction: when avoidance equal removal. In R. Dyckhoff, editor, *4th Int. Ws. on Extensions of LP*, volume 798 of *LNAI*. Springer-Verlag, 1994.

[AP96] J. J. Alferes and L. M. Pereira. *Reasoning with Logic Programming.* volume 1111 of LNAI. Springer-Verlag, 1996.

[AP97] J. J. Alferes and L. M. Pereira. Update-programs can update programs In J. Dix, L. M. Pereira and T. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming*, volume 1216 of *LNAI*. Springer-Verlag, 1997.

[APP96] J. Alferes, L. Pereira, and T. C. Przymusinski. Belief revision in non-monotonic reasoning and logic programming. *Fundamenta Informaticae*, 28(1,2):1–22, 1996.

[BDP96] Stefan Brass, Jürgen Dix, and Teodor. C. Przymusinski. Super Logic Programs. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96), Boston, MA*, pages 529–541. San Francisco, CA, Morgan Kaufmann, 1996.

[Cla78] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Dix92] J. Dix. A framework for representing and characterizing semantics of logic programs. In B. Nebel, C. Rich, and W. Swartout, editors, *3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992.

[DNPS95] C. V. Damásio, W. Nejdl, L. M. Pereira, and M. Schroeder. Model-based diagnosis preferences and strategies representation with logic meta-programming. In K. Apt and F.Turini, editors, *Meta-logics and Logic Programming*, pages 311–338. MIT Press, 1995.

[DP95] C. V. Damásio and L. M. Pereira. Abduction over 3-valued extended logic programs. In V. Marek, A. Nerode, and M. Truszczynski, editors, *Proceedings of the Third International Conference on Logic Programming and Non-Monotonic Reasoning*. LPNMR'95, Springer-Verlag, 1995.

[DR91] P. M. Dung and P. Ruamviboonsuk. Well founded reasoning with classical negation. In A. Nerode, W. Marek, and V. S. Subrahmanian, editors, *LP & NMR*, pages 120–132. MIT Press, 1991.

[Gel92] M. Gelfond. Logic programming and reasoning with incomplete information. Technical report, University of Texas at El Paso, 1992.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth Logic Programming Symposium*, pages 1070–1080, Cambridge, Mass., 1988. Association for Logic Programming, MIT Press.

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proceedings of the Seventh International Logic Programming Conference, Jerusalem, Israel*, pages 579–597, Cambridge, Mass., 1990. Association for Logic Programming, MIT Press.

[GL92] M. Gelfond and V. Lifschitz. Representing actions in extended logic programs. In K. Apt, editor, *Int. Joint Conf. and Symp. on LP*, pages 559–573. MIT Press, 1992.

[GPP89] Michael Gelfond, Halina Przymusinski, and Teodor C. Przymusinski. On the relationship between circumscription and negation as failure. *Journal of Artificial Intelligence*, 38(1):75–94, February 1989.

[GRS91] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[KM91] Hirofumi Katsuno and Alberto O. Mendolzon. On the difference between updating a knowledge base and revising it. In *Proc. KR-91*, pages 387–394, 1991.

[Kow90] R. Kowalski. Problems and promises of computational logic. In John Lloyd, editor, *Computational Logic*, pages 1–36. Basic Research Series, Springer–Verlag, 1990.

[Kow92] R. Kowalski. Legislation as logic programs. In *Logic Programming in Action*, pages 203–230. Springer–Verlag, 1992.

[McC80] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Journal of Artificial Intelligence*, 13:27–39, 1980.

[Min82] J. Minker. On indefinite data bases and the closed world assumption. In *Proc. 6-th Conference on Automated Deduction*, pages 292–308, New York, 1982. Springer Verlag.

[Moo85] R.C. Moore. Semantic considerations on non-monotonic logic. *Journal of Artificial Intelligence*, 25:75–94, 1985.

[MT94a] W. Marek and M. Truszczynski. *Non-Monotonic Logic.* Springer Verlag, 1994.

[MT94b] W. Marek and M. Truszczyński. Revision specifications by means of revision programs. In *Logics in AI. Proceedings of JELIA '94.* Lecture Notes in Artificial Intelligence. Springer-Verlag, 1994.

[Nel49] D. Nelson. Constructible falsity. *Journal of Symbolic Logic*, 14:16–26, 1949.

[PA92] L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In B. Neumann, editor, *European Conf. on AI*, pages 102–106. John Wiley & Sons, 1992.

[PAA93] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Non–monotonic reasoning with logic programming. *Journal of Logic Programming. Special issue on Nonmonotonic reasoning*, 17(2, 3 & 4):227–263, 1993.

[PDA93a] L. M. Pereira, C. Damásio, and J. J. Alferes. Debugging by diagnosing assumptions. In P. A. Fritzson, editor, *Automatic Algorithmic Debugging, AADEBUG'93*, volume 749 of *Lecture Notes in Computer Science*, pages 58–74. Springer–Verlag, 1993.

[PDA93b] L. M. Pereira, C. Damásio, and J. J. Alferes. Diagnosis and debugging as contradiction removal. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on LP & NMR*, pages 316–330. MIT Press, 1993.

[Pea90] D. Pearce. Reasoning with negative information II: Hard negation, strong negation and logic programs. In D. Pearce and H. Wansing, editors, *Nonclassical Logics and Information Processing*, pages 63–79. Springer–Verlag, 1990.

[Prz90] T. C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13(4):445–464, 1990.

[Prz91] T. C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing Journal*, 9:401–424, 1991. (Extended abstract appeared in: Extended stable semantics for normal and disjunctive logic programs. *Proceedings of the 7-th International Logic Programming Conference, Jerusalem*, pages 459–477, 1990. MIT Press.).

[Prz95a] T. C. Przymusinski. Semantics of normal and disjunctive logic programs: A unifying framework. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Proceedings of the Workshop on Non-Monotonic Extensions of Logic Programming at the Eleventh International Logic Programming Conference, ICLP'95, Santa Margherita Ligure, Italy, June 1994*, pages 43–67. Springer Verlag, 1995.

[Prz95b] T. C. Przymusinski. Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, Special Issue on Disjunctive Programs(14):323–357, 1995.

[Prz97a] T. C. Przymusinski. Autoepistemic logic of knowledge and beliefs. In preparation, University of California at Riverside, 1997. (Extended abstract appeared in 'A knowledge representation framework based on autoepistemic logic of minimal beliefs' In *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94, Seattle, Washington, August 1994*, pages 952-959, Los Altos, CA, 1994. American Association for Artificial Intelligence, Morgan Kaufmann.).

[Prz97b] T. C. Przymusinski. Plain negation as a basis for strong, weak and classical negations. In preparation, University of California at Riverside, 1997.

[PT95] T. C. Przymusinski and Hudson Turner. Update by means of inference rules. In A. Nerode, editor, *Proceedings of the Third International Conference on Logic Programming and Non-Monotonic Reasoning, Lexington, KY*, pages 156–174. LPNMR'95, Springer Verlag, 1995.

[PW90a] D. Pearce and G. Wagner. Reasoning with negative information I: Strong negation in logic programs. In L. Haaparanta, M. Kusch, and I. Niiniluoto, editors, *Language, Knowledge and Intentionality*, pages 430–453. Acta Philosophica Fennica 49, 1990.

[PW90b] D. Pearce and G. Wagner. Reasoning with negative information I: Strong negation in logic programs. In L. Haaparanta, M. Kusch, and I. Niiniluoto, editors, *Language, Knowledge and Intentionality*, pages 430–453. Acta Philosophica Fennica 49, 1990.

[Rei78] R. Reiter. On closed-world data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978.

[Rei80] R. Reiter. A logic for default theory. *Journal of Artificial Intelligence*, 13:81–132, 1980.

[Vak77] D. Vakarelov. Notes on n-lattices and constructive logic with strong negation. *Studia Logica*, 36:109–125, 1977.

[VGRS90] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 1990. (to appear). Preliminary abstract appeared in Seventh ACM Symposium on Principles of Database Systems, March 1988, pp. 221–230.

[Wag93] G. Wagner. Reasoning with inconsistency in extended deductive databases. In L. M. Pereira and A. Nerode, editors, *2nd Int. Ws. on LP & NMR*, pages 300–315. MIT Press, 1993.

[Win88] M. Winslett. Reasoning about action using a possible models approach. In *Proc. AAAI–88*, pages 89–93, 1988.

{**Last document revision:** April 24, 1997 at 1:40:26 PM. }