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

Gelfond, M.; Przymusinski, H.; and Przymusinski, T. C. 1989. On the relationship between circumscription and negation as failure. *Journal of Artificial Intelligence* 38(1):75–94.

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

Katsuno, H., and Mendolzon, A. O. 1991. On the difference between updating a knowledge base and revising it. In *Proc. KR-91*, 387–394.

Kowalski, R. 1990. Problems and promises of computational logic. In Lloyd, J., ed., *Computational Logic*, 1–36. Basic Research Series, Springer–Verlag.

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

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

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

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

Moore, R. 1985. Semantic considerations on nonmonotonic logic. *Journal of Artificial Intelligence* 25:75–94.

Nelson, D. 1949. Constructible falsity. *JSL* 14:16–26.

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

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

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

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

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

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

Przymusinski, T., and Turner, H. 1995. Update by means of inference rules. In Nerode, A., ed., *Proceedings of the Third International Conference on Logic Programming and Non-Monotonic Reasoning, Lexington, KY*. LPNMR'95.

Przymusinski, T. C. 1991. Stable semantics for disjunctive programs. *New Generation Computing Journal* 9:401–424. (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.).

Przymusinski, T. C. 1994a. 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*, 952–959. Los Altos, CA: American Association for Artificial Intelligence.

Przymusinski, T. C. 1994b. Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence* Special Issue on Disjunctive Programs. (In print).

Przymusinski, T. C. 1995. Autoepistemic logic of knowledge and beliefs. (In preparation), University of California at Riverside.

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

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

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

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

Because of the similarities between both negations, they are equivalent for the class of theories that capture the semantics of logic programs under Stable Models, whether they are extended with strong negation or with explicit negation (viz. Answer-Sets). Indeed, the result in Theorem 2.3 remains true regardless of whether "classical" negation is translated into strong negation or into explicit negation.

For logic programs, explicit negation can be seen as an approximation to strong negation, in the sense that all consequences of a belief theory in $AEBS$ remain true in $AEBX$ after strong negation is replaced everywhere by explicit negation:

Proposition 3.2 *Let T_s be a AEBS theory obtained from an extended logic program P with strong negation via the translation described in (Przymusinski 1994a), T_x be the AEBX theory obtained from T_s by replacing strong by explicit negation, and let $T_s^\diamond$ be an expansion of T_s. Then the theory $T_x^\diamond$, obtained from $T_s^\diamond$ by replacing strong by explicit negation, is an expansion of T_x.* ♢

Corollary 3.1 (Explicit Approximates Strong) *Let T_s and T_x be as in Proposition 3.2. Then, if some formula F holds in every expansion of T_x, 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[4]. However those procedures are not complete for strong negation since, as shown by Examples 3.1 and 3.2, the converse of Corollary 3.1 is not true in general.

Another result contrasting explicit and strong negation is:

Theorem 3.2 (Explicit Extends Strong) *There is a one to one correspondence between expansions of an AEBS theory T_s 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: $\overline{A} \supset \neg A$.* ♢

3.4 Explicit Negation and Logic Programming

As with strong negation, the extension $AEBX$ of AEB obtained by augmenting it with explicit negation immediately leads to the corresponding extensions of the major semantics for logic programs embeddable into AEB. As stated above, stable semantics augmented with explicit negation is equivalent to stable semantics with the so called "classical" negation of (Gelfond & Lifschitz 1990).

The following result shows that the relevant logic program semantics $WFSX$ (defined in (Pereira & Alferes 1992)) is also embeddable in $AEBX$. This embeddability result requires, besides the translation defined in (Przymusinski 1994a), 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". Lack of space prevents us from giving further details of this transformation in this extended abstract.

Theorem 3.3 (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 (Pereira & Alferes 1992), 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 $\overline{A}$.* ♢

References

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

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

Alferes, J.; Pereira, L.; and Przymusinski, T. C. 1995. Belief revision in logic programming and nonmonotonic reasoning. In Mamede, N., and Pinto-Ferreira, C., eds., *Proceedings of the International Artificial Intelligence Conference, EPIA'95*. The MIT Press.

Clark, K. 1978. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases*. New York: Plenum Press. 293–322.

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

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

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

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

[4] In the extended version, the reader can find a brief discussion on the implementation of explicit negation.

Now, the reason $AEBS$ cannot directly capture any semantics for extended logic programs complying with relevance, such as $WFSX$, is because of the strong negation axiom, $\neg A \supset \sim A$, which states 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. In the first example we have an even loop over default negation, and in the second 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. Explicit negation however does not affect either loop, as we shall see in detail when we return to the first example in Example 3.3.

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 different extension of AEB is needed, equipped with a weaker notion of symmetric negation. 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.

3.2 Introducing Explicit Negation

Similarly to strong negation, explicit negation is added to AEB with an explicit negation operator $\overline{F}$, defining *Autoepistemic Logic of Beliefs with Explicit Negation* – $AEBX$, by:

- Augmenting the original objective language $\mathcal{K}$ with new *objective* propositional symbols $\overline{A}$, called *dual or explicit negation atoms*, resulting in a new objective language $\widehat{\mathcal{K}}$ and the new language of beliefs $\widehat{\mathcal{K}}_{\mathcal{B}}$. The extension of explicit negation to arbitrary positive objective formulae can be done in the same way as for strong negation.
- Assuming the following *Coherence* inference rule[3] $\frac{\overline{F}}{\mathcal{D}F}$, which says that if one derives the dual, one has to believe the negation, i.e. "$\overline{A}$ *serves as evidence against* A".

Since the Coherence inference rule has no effect on belief theories that do not include explicit negation atoms, in the sequel we will assume it without further mention, whenever explicit negation is used.

[3] If $F = \overline{G}$ then we have $\frac{G}{\mathcal{D}\overline{G}}$.

Example 3.3 The details of this example show the essence of how explicit negation treats both previous examples, and the way it differs from strong negation.

The theory $T^{\diamond} = Cn_*(T \cup \{\mathcal{B}gtc, \mathcal{B}\neg\overline{gtc}\})$ is, with the obvious abbreviations, an expansion of the $AEBX$ theory in Example 3.1 (where strong negation is replaced by explicit negation). $\mathcal{B}\neg\overline{gtc}$ 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 3.2. 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's no backward propagation of falsity of the head to the rule's body, possibly leading to non–minimal models. 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 negatinns. 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 for $\overline{L}$. And vice-versa. $\diamondsuit$

Indeed, 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 3.1 *Let $T^{\diamond}$ be a consistent expansion of a belief theory T in $AEBX$. Then, for no atom A:*

$$T^{\diamond} \vdash_* A \wedge \overline{A}$$

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

3.3 Explicit and Strong Negation Compared

Explicit and strong negation share some similarities. The (derived) inference rule from strong negation to beliefs shown in Corollary 2.1, also holds for explicit negation. The latter 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 invariance of beliefs. The result of Theorem 2.2 still holds if strong negation is replaced by explicit negation. In other words, explicit negation is also symmetric.

3 Explicit Negation

In this section we introduce an alternative 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 contrast $AEBX$ with $AEBS$ and 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 (Pereira & Alferes 1992). In the extended version of this paper, we illustrates an application of $AEBX$ showing how belief revision in $AEBX$ can directly capture the contradiction removal techniques used for logic programs (Alferes & Pereira 1994).

3.1 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 logic programs extended with a symmetric negation (hereafter, simply called *extended logic programs* or *ELP*'s) 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 a literal should not influence its truth value. This well-known property, called *relevance* (Dix 1992), 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 (Gelder, Ross, & Schlipf 1991) obeys relevance (cf. (Dix 1992)). On the other hand, neither the Stable Models Semantics nor $AEBS$ satisfy this principle.

One paramount motivation for introducing $AEBX$ was the desire to capture those semantics of logic program that satisfy relevance and thus permit efficient, top-down implementations. As shown in Example 3.1, 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 the logic $AEBX$ is more expressive than the logic $AEBS$ by providing a meaningful semantics to those programs that appear to have a well-defined intended meaning and yet do not have a consistent semantics under $AEBS$ (see Example 3.2).

Example 3.1 Take the following theory and corresponding logic program:

$$\begin{array}{l} \mathcal{D}\text{-}god\_exists \supset god\_exists \\ \mathcal{D}god\_exists \supset \text{-}god\_exists \\ \mathcal{D}god\_exists \supset \text{-}go\_to\_church \\ go\_to\_church \end{array}$$

$$\begin{array}{rl} god\_exists & \leftarrow not\ \text{-}god\_exists \\ \text{-}god\_exists & \leftarrow not\ god\_exists \\ \text{-}go\_to\_church & \leftarrow not\ god\_exists \\ go\_to\_church & \end{array}$$

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".

Strong negation has one expansion, where god_exists because I go to church. Matter of factly, $go\_to\_church \supset \neg\text{-}go\_to\_church$ (strong negation axiom), which by contraposition entails $\neg\mathcal{D}god\_exists$. Therefore, in all minimal models, $\neg\text{-}god\_exists$ and $\mathcal{D}\text{-}god\_exists$ and hence god_exists. With explicit negation (to be defined below) we have different conclusions, and the corresponding least expansion includes $\{go\_to\_church, \mathcal{D}\text{-}go\_to\_church\}$ but not god_exists nor $\mathcal{D}\text{-}god\_exists$ (cf. Example 3.3) $\diamondsuit$

Example 3.2 Take now the following theory resulting from the translation of a corresponding logic program:

$$\begin{array}{l} \mathcal{D}shave(X,X) \supset shave(john,X) \\ shave(Y,X) \supset go\_dine\_out(X) \\ \text{-}shave(peter,peter) \\ \text{-}go\_dine\_out(john) \end{array}$$

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.

$AEBS$ assigns no semantics to this theory (there is no consistent expansion) whereas with $AEBX$, to be defined below, the theory has one expansion that includes: $\{\text{-}go\_dine\_out(john), \mathcal{D}go\_dine\_out(john)\}$ but not $shave(john,john)$. The absence of $AEBS$ expansion is brought about by the use of the contrapositive $\neg go\_dine\_out(X) \supset \neg shave(X,X)$ to conclude $\neg shave(john,john), \mathcal{D}shave(john,john)$, and hence $shave(john,john)$. *Contradiction!* $\diamondsuit$

From Corollary 2.1 and the fact that default negation, *not* F is translated into $\mathcal{D}F$, one immediately derives an important relationship between strong negation and default negation in logic programs:

Corollary 2.4 (Strong vs. Default Negation) *The inference rule:*

$$\frac{\neg F}{not\ F}$$

is satisfied by all semantics of logic programs embeddable into AEBS *via the translation* $T_{\mathcal{B}\neg}(P)$. $\diamondsuit$

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:

$$A \leftarrow not\ \neg A \quad \text{and} \quad \neg A \leftarrow not\ A.$$

2.2 Strong Negation and 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 (Alferes, Pereira, & Przymusinski 1995; Gelfond & Lifschitz 1992; Kowalski 1990; 1992; Pereira, Aparício, & Alferes 1993; Pereira, Damásio, & Alferes 1993a; 1993b; Przymusinski 1994a). Here we show how one can apply strong negation to provide a natural solution to an important knowledge representation problem, namely, theory and interpretation update. Many other applications are possible, e.g., in (Przymusinski 1994a) strong negation is used to provide a simple embedding of Gelfond's epistemic specifications (Gelfond 1992) in *AELBS* and in (Alferes, Pereira, & Przymusinski 1995) strong negation is applied to propose a natural solution to belief revision.

Katsuno and Mendelzon (Katsuno & Mendolzon 1991) 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 (Winslett 1988) 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 Γ and T be sets of propositional formulae. A set T' of formulae is a "theory update" of T by Γ if Models(T') equals

$$\{\ I' : \exists I \in \text{Models}(T)\ .\ I' \text{ is "an update of } I \text{ by } \Gamma"\ \}.$$

We see by the form of this definition 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 Γ. Below we illustrate how strong negation can be used to define interpretation update by an arbitrary belief theory in *AEBS*. This generalizes an approach to interpretation update proposed earlier in (Marek & Truszczyński 1994; Przymusinski & Turner 1995).

Suppose that Γ is the theory discussed in Example 2.1 describing an employer who periodically reappoints (hires) or fires (turns down) current employees (new candidates) based on their qualifications and physical appearance. Moreover, suppose that I is an interpretation of the language $\widehat{\mathcal{K}}$ given by:

$$\{hire(Ann), hire(John), \neg goodL(John), \neg hire(Mary), qualified(Tom), nice(Kathy)\}$$

whose intended meaning is that both Ann and John are current employees, John is bad looking, Mary was already fired, Tom is considered qualified, and Kathy is nice. In order to produce an interpretation I' of the language $\widehat{\mathcal{K}}$ which is an update of I by the belief theory Γ, we first augment Γ with the default axioms $\mathcal{D}(\neg A) \supset A$, for every atom A in I (e.g. $\mathcal{D}(\neg hire(Ann)) \supset hire(Ann)$). These axioms can be viewed as natural *inertia axioms* stating that predicates that are initially true in I remain true unless we have some evidence to the contrary. We will denote them by $Def(I)$.

The augmented belief theory $\Delta(\Gamma, I) = \Gamma \cup Def(I)$ has a unique static expansion in which Ann is reappointed because there is no reason to fire her, John is fired because he was found to have a bad appearance, Tom is hired because he is qualified and there is no indication his appearance is lacking and Mary remains fired. There is not enough information about Kathy to determine her status so she will have to be further screened.

It is therefore natural to consider the interpretation I':

$$\{hire(Ann), \neg hire(John), \neg goodlook(John), \neg hire(Mary), hire(Tom), qualif(Tom), nice(Kathy)\},$$

of $\widehat{\mathcal{K}}$ consisting of all the atomic formulae from the language $\widehat{\mathcal{K}}$ that belong to the unique static expansion of $\Delta(\Gamma, I)$, to be the *update of* I *by* Γ. In general, we may have more then one static expansion of $\Delta(\Gamma, I)$ thus resulting in more than one possible interpretation update of I.

It follows from the results proved in (Przymusinski & Turner 1995) that *revision programming*, introduced in (Marek & Truszczyński 1994), is a special case this approach obtained when the theory Γ is a *positive logic program*. However, the approach proposed in here is much more general because the updating theory Γ can be an *arbitrary belief theory* in *AEBS*. In particular, Γ can be an arbitrary propositional theory or it can represent any of the non-monotonic formalisms embeddable into *AEBS*, such as *CWA*, circumscription, autoepistemic logic and all the major semantics recently proposed for logic programs.

thus $\mathcal{D}(\text{-}goodL(John))$ is not true. However, there is also some indication that he is *not* bad looking because there exist minimal models in which $\text{-}goodL(John)$ is false and thus $\mathcal{B}(\text{-}goodL(John))$ is not true. He will therefore have to undergo further screening. The same applies to Tom.

Observe, however, that if we later find out that it was Tom who was bad looking (i.e. $\text{-}goodL(Tom)$) then $\mathcal{D}(\text{-}goodL(John))$ will become true and thus John will be hired whereas Tom will be fired (or rejected). $\diamondsuit$

Even though strong negation $\text{-}F$ so far is defined only for atomic objective formulae F, we can easily extend it to all *positive* objective formulae F of the language $\widehat{\mathcal{K}}$, i.e., those formulae which do *not* contain classical negation $\neg$. The following definition is recursive and F and G are assumed to be positive objective formulae: $\text{-}(\text{-}F) \equiv F$; $\text{-}(F \vee G) \equiv \text{-}F \wedge \text{-}G$; $\text{-}(F \wedge G) \equiv \text{-}F \vee \text{-}G$.

The above extension preserves the basic property of strong negation, namely, the fact that it is stronger than classical negation:

Proposition 2.1 *For any positive objective formula* F: $\vdash_* (\text{-}F \supset \neg F)$. $\diamondsuit$

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

Corollary 2.1 (Strong Negation vs. Beliefs) *The inference rule:*

$$\frac{\text{-}F}{\mathcal{B}\neg F} \quad \textit{or, equivalently,} \quad \frac{\text{-}F}{\mathcal{D}F}$$

is satisfied in every static autoepistemic expansion $T^\diamond$ *and every positive, objective formula* F. $\diamondsuit$

Observe that for any objective positive formula F, both F and $\text{-}F$ are *independently minimized* in the sense that, in the absence of any information to the contrary, both $\mathcal{D}F$ and $\mathcal{D}\text{-}F$ are true because F and $\text{-}F$ are false in all minimal models.

However, one can easily ensure that one of the formulae, F or $\text{-}F$, is *true by default*, and thus only the other one is minimized, by assuming one of the *default axioms*: $\mathcal{D}\text{-}F \supset F$ or $\mathcal{D}F \supset \text{-}F$ which say that if we disbelieve F (respectively, $\text{-}F$) then $\text{-}F$ (respectively, F) is true. For example, the second default axiom causes $\text{-}F$ to be true by default thus forcing strong negation to behave similarly to classical negation. This will prove useful later on when we discuss the application of strong negation to theory and interpretation update.

One can also *prevent* any minimization of F and $\text{-}F$ by assuming the *law of the excluded middle*, $F \vee \text{-}F$, for this particular formula F, which causes precisely one of F or $\text{-}F$ to be true at all times.

As we mentioned in the Introduction, nonmonotonic 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, as the following result shows, static expansions in *AEBS* are fully symmetric with respect to strong negation.

Theorem 2.2 (Invariance of Beliefs) *Suppose that* S *is a subset of the set of all objective predicates in* $\mathcal{K}$ *and let* Ψ *be the operator simultaneously replacing all occurrences of the atom* $\text{-}A$ *by* A *and all occurrences of the atom* A *by* $\text{-}A$*, for all atoms* A *in* S*. Then* $T^\diamond$ *is a static expansion of the theory* T *in* AEBS *if and only if* $\Psi(T^\diamond)$ *is a static expansion of the theory* $\Psi(T)$. $\diamondsuit$

2.1 Strong Negation and Logic Programming

Since major semantics for logic programs, including stable semantics, *STABLE*, well-founded semantics, *WFS*, and stationary semantics, *STAT*, are embeddable into *AEB* via the embedding defined in (Przymusinski 1994a), the extension *AEBS* of *AEB* obtained by augmenting it with strong negation, immediately leads to the corresponding extensions of these semantics. We denote these extended semantics augmented with strong negation by *STABLES*, *WFSS* and *STATS*, respectively.

As the following result shows, stable semantics augmented with strong negation is equivalent to stable semantics with the so called "classical negation", originally introduced by Gelfond-Lifschitz (Gelfond & Lifschitz 1990).

Theorem 2.3 *There is a one-to-one correspondence between stable models* $\mathcal{M}$ *of an extended logic program* P *with "classical negation" and those consistent static autoepistemic expansions* $T^\diamond$ *of its translation* $T_{\mathcal{B}\neg}(P)$ *into belief theory that satisfy the condition* $\mathcal{B}A \vee \mathcal{D}A \in T^\diamond$*, for all objective atoms* A*. We assume here that "classical negation" of an atom* A *is translated into its strong negation* $\text{-}A$. $\diamondsuit$

Example 2.2 The belief theory:

$$\begin{aligned} qualified(x) \wedge \mathcal{D}(\text{-}goodL(x)) &\supset hire(x) \\ \text{-}qualified(x) &\supset \text{-}hire(x) \\ \text{-}goodL(x) &\supset \text{-}hire(x) \end{aligned}$$

is the translation $T_{\mathcal{B}\neg}(P)$ of the logic program P:

$$\begin{aligned} hire(x) &\leftarrow qualified(x), not\ {\sim}goodL(x) \\ {\sim}hire(x) &\leftarrow {\sim}qualified(x) \\ {\sim}hire(x) &\leftarrow {\sim}goodL(x) \end{aligned}$$

where we denote "classical negation" of A by ${\sim}A$. $\diamondsuit$

tionship to default negation and to classical negation.

- We show how symmetric negation can be applied to provide natural solutions to various knowledge representation problems, such as theory and interpretation update. An application to belief revision is shown in the extended version of this paper.

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 and others).

The knowledge representation framework of *AutoEpistemic logic of Beliefs*, *AEB*, introduced by Przymusinski in (Przymusinski 1994a; 1995), was shown to isomorphically contain all of the above mentioned formalisms as special cases. In order to achieve that level of generality, we define the notion of symmetric negation in the logic *AEB* thus automatically providing the corresponding notion of symmetric negation for all formalisms embeddable into *AEB*. The description of the Autoepistemic Logic of Beliefs, *AEB*, as well as the translation of logic programs into belief theories in *AEB* can be found in (Przymusinski 1994a).

2 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, in some respects, is similar to classical negation $\neg F$ and yet it does not satisfy the law of the excluded middle and allows a symmetric treatment of positive and negative information. 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 by augmenting the Autoepistemic Logic of Beliefs, *AEB* (Przymusinski 1994a) with a strong negation operator, $\sim F$, thus defining the extension of *AEB* called the *Autoepistemic Logic of Beliefs with Strong Negation*, *AEBS*. As a result, all non-monotonic formalisms embeddable into *AEB*, including circumscription, autoepistemic logic and all the major semantics recently proposed for logic programs, admit corresponding extensions allowing the use of strong negation. In particular, strong negation added to logic programs with stable semantics coincides with Gelfond-Lifschitz' so called "classical negation" (Gelfond & Lifschitz 1990).

Strong negation is added to the Autoepistemic Logic of Beliefs, *AEB*, by:

- Augmenting the original objective language $\mathcal{K}$ with new *objective* propositional symbols $\sim A$, called *strong negation atoms*, resulting in a new objective language $\widehat{\mathcal{K}}$ and the new language of beliefs $\widehat{\mathcal{K}}_{\mathcal{B}}$;
- Assuming the following *strong negation* axiom: (S) $\quad A \wedge \sim A \supset \bot$ or, equivalently, $\sim A \supset \neg A$, which says that A and $\sim A$ cannot be both true and thus ensuring that the intended meaning of $\sim A$ is "$\sim A$ *is the opposite of* A". More precisely, we are requiring that the set $Cn_*(T)$ of formulae derivable from a given belief theory T in *AEB* be also closed under the strong negation axiom *(S)*.

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 axiom 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. Since the axiom (S) has no effect on those belief theories T that do not include strong negation atoms $\sim A$, in the sequel we will assume the axiom *(S)* without further mention whenever strong negation is used.

Example 2.1 Consider an employer who hires (and fires) employees based on their qualifications and physical appearance. Employees (job applicants) who are believed to be qualified and not to have a poor appearance will be (re)hired. Employees (job applicants) who are believed to be unqualified or to have a poor appearance will be fired (rejected). The remaining individuals (i.e., those whose status is not yet determined) will undergo some further selection process. This leads us to the following belief theory in *AEBS*:

$$\mathcal{B}(qualified(x)) \wedge \mathcal{D}(\sim goodL(x)) \supset hire(x)$$
$$\mathcal{B}(\sim qualified(x)) \supset \sim hire(x)$$
$$\mathcal{B}(\sim goodL(x)) \supset \sim hire(x).$$

Suppose, moreover, that both Ann and John are qualified, Mary is unqualified and either John or Tom, but we are not sure who, is bad looking:

$$qualified(Ann) \qquad \sim qualified(Mary)$$
$$qualified(John) \qquad \sim goodL(John) \vee \sim goodL(Tom).$$

The resulting belief theory has one static expansion in which Ann will be hired because she is believed to be qualified (i.e., $\mathcal{B}(qualified(Ann))$ holds) and there is no reason to believe she is bad looking (i.e., $\mathcal{D}(\sim goodL(Ann))$ holds). Mary will be fired (rejected) because she is believed to be unqualified (i.e., $\mathcal{B}(\sim qualified(Mary))$ holds). On the other hand, John will neither be hired nor fired (rejected). Indeed, even though John is believed to be qualified there is some indication that he has a poor appearance because there exist minimal models in which $\sim goodL(John)$ holds and

First of all, real 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 (Przymusinski 1994b). There are, however, other important reasons why classical negation $\neg F$ is often not suitable for logic programming, deductive databases and other non-monotonic formalisms:

- 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 others are false and yet some others are not (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 others 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 (Gelfond 1992)). However, if we attempt to describe the first two statements using 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.

- As we pointed out before, default negation *not* A does not treat positive and negative literals symmetrically because only negation *not* A of *positive* literals (atoms) is assumed by default whereas the same is not true for default negation *not* ($\neg A$) applied to negative literals. Somewhat imprecisely, one can say that atomic formulae are minimized while their negations are not. Often times, however, we would like to minimize (by default) both positive and negative information. For instance, in the preceding example, given no information about *Tom*'s qualifications we would like to conclude (using, e.g., $GCWA$ or circumscription) that Tom was neither found qualified nor unqualified, i.e., that both *not* $(qualified(Tom))$ and *not* $(\neg qualified(Tom))$ hold. This would tell us that Tom has to be interviewed. However, this is not possible with classical negation $\neg F$ because the law of the excluded middle does not allow us to minimize both A and $\neg A$ since one of them must always be true.

- 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 and thus difficult to implement.

1.3 Symmetric negation

Based on the preceding discussion, we conclude that, in addition to default negation, *not* F, we need a new type of negation, which, in some respects, is similar to classical negation, $\neg F$, in the sense that it is not assumed "by default" and, yet, in some others, it is very different from classical negation. In particular, it does not satisfy the law of the excluded middle and thus allows for a symmetric treatment of positive and negative information. We will call it *symmetric negation*.

Gelfond and Lifschitz (Gelfond & Lifschitz 1990) were the first to point out the need for such negation in logic programming and they also proposed a specific semantics for such negation for logic programs with the stable semantics. Somewhat unfortunately, they called their negation *"classical negation"*[1]. Subsequently, several researchers proposed different, often incompatible, forms of symmetric negation for various semantics of logic programs and deductive databases (Dung & Ruamviboonsuk 1991; Pereira & Alferes 1992; Przymusinski 1991; 1994b; Pearce 1990; Wagner 1993). 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*. Both coincide with Gelfond-Lifschitz' "classical negation" for logic programs with the stable semantics.

- We study properties of both symmetric negations and their mutual relationship as well as their rela-

[1] Independently, in (Pearce & Wagner 1990) the authors also pointed out the need for such a negation in logic programs. They called it "strong" negation, due to the desired similarities with Nelson's strong negation (Nelson 1949). However their "strong" negation should not be confused with the strong negation defined in this paper.

[2] For logic programs, an extensive study was carried out in (Alferes & Pereira 1992).

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

José Alferes* and **Luís Moniz Pereira***
DM, U. Évora and CRIA Uninova
DCS, U.Nova de Lisboa and CRIA Uninova
2825 Monte da Caparica, Portugal
(jja—lmp@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* A, whose major distinctive feature is the fact that *not* A *is assumed in the absence of* "sufficient evidence" *supporting the atomic formula* A. The meaning of *"sufficient evidence"* depends on the specific semantics used. For example, in Reiter's original *Closed World Assumption*, *CWA* (Reiter 1978), *not* A is assumed 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* (Minker 1982; Gelfond, Przymusinski, & Przymusinski 1989), or in McCarthy's Circumscription, *CIRC* (McCarthy 1980), *not* A is assumed only if A is false in *all* minimal models. In Clark's original *Predicate Completion Semantics* (Clark 1978) for logic programs, this form of negation has been called *negation-by-failure* because *not* A was derivable whenever attempts to prove A finitely failed. For example, the clause $acquitted(x) \leftarrow charged(x), not\ guilty(x)$ says that one should be acquitted of any charges unless he is proven guilty, i.e., unless "sufficient evidence" of his guilt is demonstrated.

The recent semantics proposed for logic programs and deductive databases, such as the *stable semantics*, *well-founded semantics*, and *stationary semantics*, propose even more sophisticated meanings for default negation, closely related to more general non-monotonic formalisms such as *Default Logic*, *DL* (Reiter 1980), *AutoEpistemic Logic*, *AEL* (Moore 1985), and *AutoEpistemic logic of Beliefs*, *AEB* (Przymusinski 1994a; 1995).

1.1 Default negation does not suffice

Although default negation proved to be quite useful in various domains and application frameworks, there are at least two important reasons why it is not the only form of negation that is needed in non-monotonic formalisms:

- Negation *not* A of an atomic formula A is always assumed "by default". However, we often need to be more careful before jumping to negative conclusions. For example, it would make little sense to say $guilty(x) \leftarrow not\ innocent(x)$ 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: $drive(x) \leftarrow crossing(x), not\ train(x)$ to express the fact that one may safely cross a railway track if he first makes sure that there is no train approaching, because such a clause would imply that he should cross the railway even if he did not bother to look around at all.
- Although negation *not* F is always assumed "by default" for atomic formulae F, the same is not true for negated atoms $F = \neg A$ and thus default negation does not treat literals A and $\neg A$ symmetrically. For example, even though GCWA (or circumscription) applied to the theory

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

  implies *not* $convicted(tom)$ this is no longer the case if we replace $guilty(x)$ by $\neg innocent(x)$, because the new theory has minimal models in which $convicted(tom)$ is actually true.

  As we can see, just replacing *guilty* by $\neg$*innocent* has a significant impact on the semantics of the resulting theory. As a consequence, default negation, *not* A, is *not invariant* under simple equivalence or *renaming* of predicates.

1.2 Classical negation does not solve the problem

One can argue that the above features of default negation were intentionally chosen to differentiate default negation *not* F from classical negation $\neg F$ and therefore if *not* F does not work in some particular context then $\neg F$ should be used instead. While the first part of this statement is certainly correct the second is not.

*Partially supported by JNICT-Portugal and ESPRIT project Compulog 2 (no. 6810).

†Partially supported by the National Science Foundation grant #IRI-9313061.