A Knowledge Representation Framework Based on Epistemic Logic

Teodor Przymusinski*

UCR-CS-93-2

*Department of Computer Science, University of California, Riverside, CA 92521. Email address: *teodor@cs.ucr.edu.*
Research partially supported by the National Science Foundation grant #IRI-89-10729.

Abstract

We introduce a uniform non-monotonic framework for knowledge representation based on epistemic logic which is sufficiently general to encompass several non-monotonic formalisms, including circumscription, autoepistemic logic, various semantics proposed for logic programs and deductive databases (stable semantics, well-founded semantics and stationary semantics) as well as Gelfond's epistemic specifications.

The existence of such a uniform framework allows us not only to provide simpler and perhaps more natural definitions of various formalisms but it also enables us to better understand mutual relationships existing between them.

Keywords: Non-monotonic reasoning, semantics of logic programs, disjunctive logic programs and deductive databases.

1 Introduction

In recent years, various approaches to non-monotonic reasoning and different semantics for normal and disjunctive logic programs have been proposed, including:

- Autoepistemic Logic [Moo85];
- Circumscription, CWA, GCWA, ECWA, etc. [McC80, Rei78, Min82, GPP89].
- Circumscriptive Epistemic Logic [Prz91a];
- Epistemic Specifications [Gel92];
- Stable Semantics (with "classical" negation) [GL88, GL90];
- Well-Founded and Stationary Semantics [VGRS90, Prz90];
- Disjunctive Stable Semantics [Prz91c];
- Disjunctive Stationary Semantics [Prz91b].

In this paper we introduce a uniform non-monotonic knowledge representation framework which incorporates the following features:

- It includes all of the non-monotonic formalisms mentioned above as special cases. More precisely, each one of these formalisms can be properly *embedded* into the new framework[1].
- It is more *expressive* than each one of these formalisms considered individually.
- It is defined in the language of *propositional logic* by means of a simple epistemic-style fixed-point equation.

Remark 1.1 *The proposed framework does* not *include Reiter's default logic [Rei80] as a special case. It is however closely related to the formalism of* stationary default logic *proposed earlier in [PP92] which includes Reiter's default logic as a special case. It should be possible to combine both formalisms into one thus obtaining a yet more general framework.* □

2 Propositional Language

Before introducing the formalism we need to define the propositional language $\mathcal{K}$ in which our formalism is defined. The language $\mathcal{K}$ includes three types of propositional constants corresponding to three types of reasoning: *objective reasoning*, *positive introspection* and *default introspection*:

[1] Only propositional circumscription is considered here.

Objective Reasoning: *Objective* propositional symbols "A" ;

Positive Introspection: Propositional symbols "$\mathcal{L}F$", where F is any propositional formula expressible in the language $\mathcal{K}$, with the intended meaning that the formula F is *true* in a given theory T, or, more precisely, that F is logically implied by T, i.e., $T \models F$;

Default Introspection: Propositional symbols "$\mathcal{D}F$", where F is any propositional formula expressible in the language $\mathcal{K}$, with the intended meaning that the formula F is *true by default* in a given theory T, or, more precisely, that it is implied by circumscription of T, i.e., $T \models_{circ} \mathrm{F}$.

In general, the choice of the *default formalism* used is application dependent. In this paper we use *circumscription* $T \models_{circ} F$ which *minimizes* objective propositional symbols[2] and *fixes* all the remaining (introspective) ones, i.e.:

$$T \models_{circ} F \equiv CIRC(T; Obj) \models F.$$

The above definition states that the propositional language $\mathcal{K}$ has the property (see also [Lif89]) that for any formula F in $\mathcal{K}$ its alphabet also contains the corresponding *propositional constants* $\mathcal{L}F$ and $\mathcal{D}F$, i.e., propositional symbols whose names consist of a string beginning with the symbol $\mathcal{L}$ or $\mathcal{D}$ followed by the formula F. It is important to realize that from a formal standpoint these new propositional constants $\mathcal{L}F$ and $\mathcal{D}F$ are *no different* than any other propositional symbols. They have, however, a special intended meaning, namely, intuitively $\mathcal{L}F$ is supposed to mean "*F is true*" while $\mathcal{D}F$ means *"F is true by default"*.

Remark 2.1 *The introduction of* introspective propositions *$\mathcal{L}F$ and $\mathcal{D}F$ is just a technical device, whose purpose is to avoid defining these symbols as new unary connectives (like, e.g., negation $\neg$), which would then force us to provide them with suitable truth valuations, and, to avoid defining them as modal operators, which would then take us out of the realm of propositional calculus. By using these propositions we remain firmly within* classical propositional calculus. *At the same time, the above definition allows arbitrarily deep* nesting *of positive and default introspection, e.g., it allows formulae of the form $\mathcal{L}(\neg\mathcal{D}A \vee \mathcal{L}A)$.*

It is easy to see that any *propositional language $\mathcal{K}^*$ can be expanded to the language $\mathcal{K}$ satisfying these conditions. Namely, it suffices to define $\mathcal{K}^0 = \mathcal{K}^*$,*

$$\mathcal{K}^{n+1} = \mathcal{K}^n \cup \{\mathcal{L}F : F \in \mathcal{K}^n\} \cup \{\mathcal{D}F : F \in \mathcal{K}^n\}$$

and $\mathcal{K} = \bigcup_{n<\omega} \mathcal{K}^n$. In order to preserve the countability of the language, it is tacitly assumed here that unique propositions $\mathcal{L}F$ and $\mathcal{D}F$ correspond to the whole equivalence class of formulae tautologically *equivalent to F.* □

[2]The author is indebted to L. Yuan for pointing out the need to use prioritized circumscription in the definition of circumscriptive epistemic logic [YYar]

3 Static Expansions and Static Epistemic Logic

In this section we introduce *static expansions* $\mathcal{E}$ of a propositional theory T in the language $\mathcal{K}$. They will combine stable autoepistemic expansions of Moore [Moo85] and stable circumscriptive expansions of Przymusinski [Prz91a]. For simplicity, any propositional theory T expressed in the language $\mathcal{K}$ will be called an *epistemic theory*. We will also assume that all theories are implicitly *closed* under logical consequence.

Definition 3.1 *A theory $\mathcal{E}$ is called a* static expansion *of an epistemic theory T if it satisfies the following fixed-point equation:*

$$\mathcal{E} = T \cup \{\mathcal{L}F : \mathcal{E} \models F\} \cup \{\neg\mathcal{L}F : \mathcal{E} \not\models F\} \cup$$

$$\cup \{\mathcal{D}F : \mathcal{E} \models_{circ} F\} \cup \{\neg\mathcal{D}F : \mathcal{E} \models_{circ} \neg F\}.\Box$$

We will call the epistemic logic based on static expansions *static epistemic logic*. Observe that the first part of the definition is identical to the definition of stable expansions in *autoepistemic logic* AEL [Moo85] while the second part coincides[3] with the definition of circumscriptive expansions in *circumscriptive epistemic logic* AEL_{circ} [Prz91a]. Also notice that $\neg\mathcal{D}F \in \mathcal{E}$ if and only if $\mathcal{E} \models_{circ} \neg F$ and *not* when $\mathcal{E} \not\models_{circ}$F. This is a very important feature distinguishing default introspection from positive introspection.

It is easy to see that both autoepistemic logic AEL and circumscriptive epistemic logic AEL_{circ} are *special cases* of the new framework. Indeed, it suffices to observe that for any autoepistemic theory T which uses only (Moore's) positive introspection operator $\mathcal{L}$ there is a one-to-one correspondence between *stable* expansions of T and *static* expansions of T. Since an analogous result holds for circumscriptive epistemic logic we immediately obtain:

Corollary 3.1 *Both autoepistemic logic AEL and circumscriptive epistemic logic AEL_{circ} are* special cases *of static epistemic logic. More precisely, both formalisms can be properly embedded into the formalism based on static expansions.* $\Box$

While both autoepistemic logic and circumscriptive epistemic logic can be embedded in the static epistemic logic, the latter one is clearly more expressive than each one of them considered separately.

It is also clear that (propositional) circumscription (and thus also $GCWA$, $ECWA$ and CWA) is a special case of the proposed formalism. Indeed, if T is a theory which does not contain any introspective propositions $\mathcal{L}F$ and $\mathcal{D}F$ then the unique static expansion of T implies the default proposition $\mathcal{D}F$ if and only if F is implied by circumscription of T, i.e., if $T \models_{circ} F$. Consequently, we obtain:

[3]Except that it uses $\neg\mathcal{D}F$ instead of Not_F.

Corollary 3.2 *Propositional circumscription, CWA, GCWA and ECWA are* special cases *of static epistemic logic. More precisely, all of these formalisms can be properly embedded into the formalism based on static expansions.* □

3.1 Stable Semantics of Logic Programs

Since Moore's autoepistemic logic is a special case of static epistemic logic, it follows from the results of Gelfond and Lifschitz [GL88] that stable semantics can be obtained by means of a suitable translation of a logic program into an epistemic theory. Namely, for a logic program P consisting of clauses:

$$A \leftarrow B_1, ..., B_m, \sim C_1, ..., \sim C_n$$

define $T_{\mathcal{L}}(P)$ to be its translation into the epistemic theory consisting of formulae:

$$B_1 \wedge ... \wedge B_m \wedge \neg\mathcal{L}C_1 \wedge ... \wedge \neg\mathcal{L}C_n \supset A.$$

The translation $T_{\mathcal{L}}(P)$ is obtained therefore by replacing the *negation by default* $\sim C$ appearing in the logic program P by $\neg\mathcal{L}C$.

Theorem 3.3 (cf. [GL88]) *There is a one-to-one correspondence between stable models $\mathcal{M}$ of the program P and static expansions $\mathcal{E}$ of $T_{\mathcal{L}}(P)$. Namely:*

$$A \in \mathcal{M} \quad \textit{iff} \quad \mathcal{L}A \in \mathcal{E}$$

$$\neg A \in \mathcal{M} \quad \textit{iff} \quad \neg\mathcal{L}A \in \mathcal{E}.\Box$$

3.2 Well-Founded and Stationary Semantics

Similarly, since circumscriptive epistemic logic is a special case of static epistemic logic, it follows from the results of Przymusinski [Prz91a] that partial stable semantics can be obtained by means of a suitable translation of a logic program into an epistemic theory. Namely, for a logic program P consisting of clauses:

$$A \leftarrow B_1, ..., B_m, \sim C_1, ..., \sim C_n$$

define $T_{\mathcal{D}}(P)$ to be its translation into the epistemic theory consisting of formulae:

$$B_1 \wedge ... \wedge B_m \wedge \neg\mathcal{D}C_1 \wedge ... \wedge \neg\mathcal{D}C_n \supset A.$$

The translation $T_{\mathcal{D}}(P)$ is obtained therefore by replacing the *negation by default* $\sim C$ appearing in the logic program P by $\neg\mathcal{D}C$.

Theorem 3.4 (cf. [Prz91a]) *There is a one-to-one correspondence between partial stable models $\mathcal{M}$ of the program P and static expansions $\mathcal{E}$ of $T_{\mathcal{D}}(P)$. Namely:*

$$A \in \mathcal{M} \quad \textit{iff} \quad \mathcal{D}A \in \mathcal{E}$$

$$\neg A \in \mathcal{M} \quad \textit{iff} \quad \neg \mathcal{D}A \in \mathcal{E}.$$

Since the well-founded model of P coincides with the least *partial stable model of P [Prz90], it corresponds to the least static expansion of $T_{\mathcal{D}}(P)$.*

Moreover, (total) stable models $\mathcal{M}$ of P correspond to those static expansions $\mathcal{E}$ of $T_{\mathcal{D}}(P)$ that satisfy the condition:

$$\textit{for all } A, \textit{ either } \mathcal{D}A \in \mathcal{E} \textit{ or } \neg \mathcal{D}A \in \mathcal{E},$$

i.e., those static expansions that completely define the truth of all default propositions $\mathcal{D}A$. □

3.3 Combining Stable and Well-founded Negations

As we have seen above, both stable and well-founded negation in logic programs can be obtained by translating $\sim C$ into introspective literals $\neg \mathcal{L}C$ and $\neg \mathcal{D}C$, respectively. However, the existence of both types of introspective literals in static epistemic logic allows us to *combine both types of negation* in one epistemic theory consisting of formulae of the form:

$$B_1 \wedge ... \wedge B_m \wedge \neg \mathcal{L}C_1 \wedge ... \wedge \neg \mathcal{L}C_k \wedge \neg \mathcal{D}C_{k+1} \wedge ... \wedge \neg \mathcal{D}C_n \supset A.$$

Such an epistemic theory may be viewed as representing a more general logic program which permits the simultaneous use of both types of negation. In such logic programs, the first k negative premises represent stable negation and the remaining ones represent the well-founded negation. The ability to use both types of negation significantly increases the expressibility of logic programs. Examples of such programs with mixed negations will be given in the full paper.

3.4 Adding "Classical" Negation

We will now add to static epistemic logic the so called "classical negation" in the sense of Gelfond and Lifschitz [GL90]. As pointed out by many researchers, the form of negation proposed by Gelfond and Lifschitz does not really represent classical negation but rather its weaker form which does not require the law of excluded middle $A \vee \neg A$. Consequently, following Pereira et.al. [PAA91] we will call it *explicit negation*.

In order to use explicit negation in epistemic logic it suffices to add to the language $\mathcal{K}$ new *objective* propositional symbols "$\tilde{A}$" with the intended meaning that $\tilde{A}$ is the "*negation of A*", augmented with the *explicit negation axioms*:

$$A \wedge \tilde{A} \supset \text{false}\,.$$

As we mentioned above, the law of excluded middle $A \vee \tilde{A}$ is not assumed. As pointed out by Bob Kowalski, the proposition A may describe the property of being *"good"* while proposition $\tilde{A}$ describes the property of being *"bad"*. The explicit negation axiom states that things cannot be both good and bad. We do not assume, however, that things must always be either good or bad.

This method of defining explicit (or "classical") negation applies to *all* epistemic theories. In particular, when applied to logic programs, one obtains the following one-to-one correspondence between stable (partial stable) models of logic programs with explicit (or "classical") negation [GL88, Prz90] and static expansions of their translation into epistemic logic which directly generalizes the results obtained before.

Theorem 3.5 *There is a one-to-one correspondence between stable (resp. partial stable) models $\mathcal{M}$ of a logic program P with explicit negation and static expansions $\mathcal{E}$ of $T_{\mathcal{L}}(P)$ (resp. $T_{\mathcal{D}}(P)$). Namely:*

$$L \in \mathcal{M} \quad \textit{iff} \quad \mathcal{L}L \in \mathcal{E} \quad (\textit{resp. } \mathcal{D}L \in \mathcal{E})$$

$$\neg L \in \mathcal{M} \quad \textit{iff} \quad \neg\mathcal{L}L \in \mathcal{E} \quad (\textit{resp. } \neg\mathcal{D}L \in \mathcal{E}).$$

Here L represents either a standard propositional symbol A or its explicit negation $\tilde{A}$. □

3.5 Adding Generalized Closed World Assumption

It is also easy to add a suitable form of *Generalized Closed World Assumption* (GCWA) [Min82, GPP89] to static epistemic theories. It suffices to add the axioms:

$$[GCWA] \quad \mathcal{L}\mathcal{D}(\neg F) \supset \neg F$$

for any positive formula F. The axiom states that negation of a formula F holds in a given theory T if F can be shown to be *false by default*, i.e., if F is false in all minimal models of T.

Observe that this $GCWA$ axiom is different from similar axioms:

$$[CWA] \quad \neg\mathcal{L}F \supset \neg F$$

and

$$[GCWA'] \quad \neg\mathcal{D}F \supset \neg F$$

which were considered earlier.

The second axiom CWA essentially describes Reiter's CWA [Rei78]. It is significantly stronger than both $GCWA$ and $GCWA'$. The third axiom $GCWA'$ is also stronger than $GCWA$ but weaker than CWA.

Example 3.1 *Adding the $GCWA$ axiom to the theory $\neg\mathcal{D}A \supset A$ results in an epistemic theory with one static expansion in which neither $\mathcal{D}A$ nor $\neg\mathcal{D}A$ holds. However, adding either the axiom CWA or the axiom $GCWA'$ to the same theory results in an inconsistent theory, i.e., a theory without static expansions.* □

3.6 Semantics of Disjunctive Logic Programs and Disjunctive Deductive Databases

As it was the case with normal logic programs, static expansions can be used to define the semantics of *disjunctive logic programs* (see [LMR92]). For any disjunctive logic program P:

$$A_1 \vee ... \vee A_l \leftarrow B_1, ..., B_m, \sim C_1, ..., \sim C_n$$

define $T^*_{\mathcal{D}}(P)$ to be its translation into the epistemic theory consisting of formulae:

$$B_1 \wedge ... \wedge B_m \wedge \neg\mathcal{D}C_1 \wedge ... \wedge \neg\mathcal{D}C_n \supset A_1 \vee ... \vee A_l$$

together with the $GCWA$ axioms:

$$\mathcal{L}\mathcal{D}(\neg F) \supset \neg F$$

and *distributive axioms* for default introspection:

$$\mathcal{D}(F \vee G) \equiv \ \mathcal{D}F \vee \mathcal{D}G$$

$$\mathcal{D}(F \wedge G) \equiv \ \mathcal{D}F \wedge \mathcal{D}G.$$

The transformation $T^*_{\mathcal{D}}(P)$ is obtained by replacing the *negation by default* $\sim C$ appearing in the premises of P by $\neg\mathcal{D}C$ and is a generalization of the translation previously used with partial stable models.

It turns out that this transformation immediately leads to the *stationary semantics* of disjunctive logic programs defined[4] in [Prz91b]:

Theorem 3.6 *There is a one-to-one correspondence between stationary expansions of the disjunctive program P and static expansions of its translation $T^*_{\mathcal{D}}(P)$. In particular, the least stationary expansion of P corresponds to the* least *static expansion of $T^*_{\mathcal{D}}(P)$.* □

Stationary semantics has a number of important advantages [Prz90]:

- It is defined for all disjunctive programs and every disjunctive program has the least stationary expansion;
- It can be computed by means of natural iterative minimal model or fixed-point procedures;
- For normal programs it coincides with the partial stable (well-founded) semantics.

The last theorem gives a particularly straightforward characterization of stationary semantics.

[4]We refer here to the definition *not* using the disjunctive inference rule.

3.7 Adding the Localization Axiom

Suppose that P is the program:

$$\begin{array}{lll} A \vee B & \leftarrow & \\ C & \leftarrow & \sim A \\ C & \leftarrow & \sim B \end{array}$$

After transformation by $T^*_{\mathcal{D}}(P)$, this program has three static (or stationary) expansions generated by the formulae listed below:

$$\mathcal{E}_1 = \{A \vee B,\ \neg A \vee \neg B,\ C\}$$

$$\mathcal{E}_2 = \{\neg A,\ \ B,\ \ C\}$$

$$\mathcal{E}_3 = \{A,\ \ \neg B,\ \ C\}.$$

The last two of them can be viewed as representing *models* of the program P while the first one represents the so called *state* or a model consisting of disjunctions [MR90]. States are very important for the proper definition of semantics of disjunctive programs. However, if for some reason we would prefer to limit ourselves to models only, this can be easily achieved in our formalism by introducing the following *localization axiom*:

$$\mathcal{L}(F \vee G) \equiv\ \mathcal{L}F \vee \mathcal{L}G,$$

for any *positive* formulae F and G. The axiom says that if a disjunction is logically implied by the theory then so is one of its components.

After adding localization axioms to the above definition of transformation $T^*_{\mathcal{D}}(P)$ we obtain the *partial stable semantics of disjunctive programs* introduced in [Prz91c]:

Theorem 3.7 *In the presence of localization axioms, there is a one-to-one correspondence between disjunctive partial stable models $\mathcal{M}$ of the program P and static expansions $\mathcal{E}$ of its translation $T^*_{\mathcal{D}}(P)$ into epistemic theory.*

Moreover, (total) disjunctive stable models $\mathcal{M}$ of P correspond to those static expansions $\mathcal{E}$ of $T^*_{\mathcal{D}}(P)$ that satisfy the condition:

$$\textit{for all } A,\ \textit{either}\ \ \mathcal{D}A \in \mathcal{E}\ \ \textit{or}\ \ \neg\mathcal{D}A \in \mathcal{E},$$

i.e., those static expansions that completely define the truth of all default propositions $\mathcal{D}A$. □

Both results characterizing semantics of disjunctive programs extend *verbatim* to the case of disjunctive programs with explicit (or "classical") negation.

3.8 Epistemic Specifications

Epistemic specifications were introduced by Gelfond [Gel92] using his newly defined language of belief sets and world views. We will now show that epistemic specifications constitute a special case

of static epistemic logic and thus, in particular, can be defined entirely in the language of classical propositional logic.

Let G be the database describing Gelfond's epistemic specification. Define $T(G)$ to be its translation into static epistemic logic (with explicit negation) obtained by:

- Replacing, for all objective propositions A, the classical negation symbol $\neg A$ by the explicit negation symbol $\tilde{A}$.

  Motivation: This substitution is caused by the fact that in his paper Gelfond uses the classical negation symbol $\neg A$ when in fact he refers to *explicit negation* $\tilde{A}$. Using the classical negation symbol $\neg A$ to denote explicit negation $\tilde{A}$ makes it impossible to distinguish between the two different types of negation and thus unnecessarily limits the expressive power of resulting theories. It also leads to some technical difficulties. The substitution allows us to reserve the standard negation symbol $\neg A$ for true classical negation. As a result, we are able to express everything that Gelfond can express in his language and more because now we effectively have two negations at our disposal. In particular, we are able to express things like $\neg\tilde{A}$, etc.

- Replacing everywhere Gelfond's "possibility" operator $\mathbf{M}F$ by $\neg\mathbf{K}\neg F$.

  Motivation: Gelfond's "possibility" operator $\mathbf{M}F$ is shown to be in fact *equivalent* to $\neg\mathbf{K}\neg F$, and, vice versa, Gelfond's "belief" operator $\mathbf{K}F$ is shown to be equivalent to $\neg\mathbf{M}\neg F$. Consequently, *only one* of these operators is really needed. This observation was made possible by the fact that we ensured a clear distinction between the two types of negation. Notice that in these equivalences we use the *real* classical negation and not the explicit negation.

- Finally, replacing everywhere Gelfond's "belief" operator $\mathbf{K}F$ by $\mathcal{LD}F$.

  Motivation: The meaning of Gelfond's belief operator $\mathbf{K}F$ is shown to be equivalent to the fact that the formula F can be shown to be *true by default*, i.e., true in all minimal models of the theory.

Now we can show that epistemic specifications are properly embeddable into static epistemic logic:

Theorem 3.8 *Epistemic specifications are a special case of static epistemic expansions, i.e., they can be properly embedded into static epistemic logic.*

More precisely, there is a one-to-one correspondence between static expansions $\mathcal{E}$ of $T(G)$ and Gelfond's world views V of G, i.e., to every static expansion $\mathcal{E}$ of $T(G)$ there corresponds a unique world view V of G, and, vice versa.

Moreover, there is a one-to-one correspondence between minimal models $\mathcal{M}$ of static expansions $\mathcal{E}$ of $T(G)$ and belief sets B of the corresponding world view V of G. □

The above theorem can be summarized as follows:

$$\text{Static expansions } \mathcal{E} \text{ of } T(G) \Leftrightarrow \text{ World views } V \text{ of } G$$

$$\text{Minimal models of } \mathcal{E} \Leftrightarrow \text{ Belief sets of } V.$$

While epistemic specifications can be embedded into static epistemic logic, the latter formalism is strictly *more expressive* than the former. For example, static epistemic logic allows us to express well-founded negation which cannot be naturally defined using epistemic specifications.

Remark 3.1 *The limited size of this abstact does not allow us to provide complete details. In particular, Gelfond's formalism also allows universally and existentially quantified formulae and a suitable form of "negation as failure". However, since the quantification takes place over the* Herbrand universe*, it can be equivalently represented by the usual instantiation of the theory which reduces it to a purely propositional case. Similarly, Gelfond's "negation by failure" can be easily translated into the language of static epistemic logic. Details will be given in the full paper.* □

Example 3.2 *Consider Gelfond's example G given by:*

$$\begin{array}{lll} Pa \vee Pb & \leftarrow & \\ \neg Pa & \leftarrow & \neg \mathbf{M} Pa \\ \neg Pb & \leftarrow & \neg \mathbf{M} Pb \\ \neg Pc & \leftarrow & \neg \mathbf{M} Pc \end{array}$$

After translation, we obtain the epistemic theory $T(G)$:

$$\begin{array}{lll} Pa \vee Pb & \leftarrow & \\ \widetilde{Pa} & \leftarrow & \mathcal{LD} \neg Pa \\ \widetilde{Pb} & \leftarrow & \mathcal{LD} \neg Pb \\ \widetilde{Pc} & \leftarrow & \mathcal{LD} \neg Pc \end{array}$$

The theory $T(G)$ has three static epistemic expansions which are easily shown to be in one-to-one correspondence with the world views of G. □

The fact that epistemic specifications can be properly embedded into static epistemic logic has some important consequences:

- It shows that the formalism of epistemic specifications can be expressed entirely in the language of classical propositional logic. In particular, it allows us to view epistemic specification rules as pure logical implications.
- It ensures a proper distinction between classical and explicit negation which not only increases the expressive power of the formalism but also allows us to establish the equivalences $\mathbf{K}F \equiv \neg \mathbf{M} \neg F$ and $\mathbf{M}F \equiv \neg \mathbf{K} \neg F$ between Gelfond's "belief" and "possibility" operators.

- It allows us to consider arbitrarily complex theories and not just restricted epistemic databases (as defined in [Gel92]).
- Static expansions constitute a strictly stronger (more expressive) framework than epistemic specifications. In particular, they allow us to express well-founded negation which is not naturally expressible in epistemic specifications.
- It simplifies the definition of epistemic specifications.

4 Conclusion

We introduced a simple and uniform framework which encompasses several non-monotonic formalisms and several semantics of normal and disjunctive logic programs. The proposed formalism is defined in the language of classical propositional logic and is more expressive than the individual formalisms that it generalizes. It allows us to better understand mutual relationships existing between different formalisms and leads to simpler and more natural definitions of some of them.

The proposed formalism is quite flexible allowing various extensions and modifications, including:

- Use of a different formalism to define *default introspection* $\mathcal{D}F$. In this paper we used prioritized circumscription $T \models_{circ} F$. For example, by using the *Weak Generalized Closed World Assumption* $WGCWA$ [RLM89] instead of $GCWA$, one can ensure that disjunctions are treated inclusively rather than exclusively.
- Suitable expansion or restriction of the underlying propositional language $\mathcal{K}$ obtained, for example, by allowing default introspection $\mathcal{D}F$ only on a selected class of formulae F.
- Addition of additional axioms, such as the $GCWA$, *Localization* and *Distributive* axioms discussed in the paper.

By using such modifications we may be able to tailor the formalism of static epistemic logic to fulfill the needs of different application domains.

Acknowledgments

The author is grateful to Michael Gelfond and Halina Przymusinska for helpful discussions and comments on the subject of this paper.

References

[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.

[GPP89] M. Gelfond, H. Przymusinska, and T. Przymusinski. On the relationship between circumscription and negation as failure. *Journal of Artificial Intelligence*, 38:75–94, 1989.

[Lif89] V. Lifschitz. Between circumscription and autoepistemic logic. Research report, Stanford University, 1989.

[LMR92] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT Press, Cambridge, Massachusetts, 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.

[MR90] J. Minker and A. Rajasekar. A fixpoint semantics for disjunctive logic programs. *Journal of Logic Programming*, page (to appear), 1990.

[PAA91] L.M. Pereira, J.A. Aparicio, and J.J. Alferes. Non-monotonic reasoning with well-founded semantics. In *Proceedings of the Eigth International Logic Programming Conference, Paris, France*, pages 475–489, Cambridge, Mass., 1991. Association for Logic Programming, MIT Press.

[PP92] H. Przymusinska and T. C. Przymusinski. Stationary extensions of default theories. *Fundamenta Informaticae*, page (In print.), 1992. Special issue devoted to the Fourth Workshop on Non-monotonic Reasoning, Plymouth, Vermont, 1992.

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

[Prz91a] T. C. Przymusinski. Autoepistemic logics of closed beliefs and logic programming. In A. Nerode, W. Marek, and V.S. Subrahmanian, editors, *Proceedings of the First International Workshop on Logic Programming and Non-monotonic Reasoning, Washington, D.C., July 1991*, pages 3–20, Cambridge, Mass., 1991. MIT Press.

[Prz91b] T. C. Przymusinski. Semantics of disjunctive logic programs and deductive databases. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *Proceedings of the Second International Conference on Deductive and Object-Oriented Databases DOOD'81*, pages 85–107, Munich, Germany, 1991. Springer Verlag.

[Prz91c] 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.).

[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.

[RLM89] A. Rajasekar, J. Lobo, and J. Minker. Weak generalized closed world assumption. *Journal of Automated Reasoning*, 5:293–307, 1989.

[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.

[YYar] L. Yuan and J. You. Autoepistemic circumscription and logic programming. Technical Report TR92-18, Dept. of Computing Science, Univerisity of Alberta, 1992. *Journal of Automated Reasoning*, (to appear).