Revisiting Epistemic Specifications

Mirosław Truszczyński

Department of Computer Science
University of Kentucky
Lexington, KY 40506, USA
mirek@cs.uky.edu

In honor of Michael Gelfond on his 65th birthday!

Abstract. In 1991, Michael Gelfond introduced the language of epistemic specifications. The goal was to develop tools for modeling problems that require some form of meta-reasoning, that is, reasoning over multiple possible worlds. Despite their relevance to knowledge representation, epistemic specifications have received relatively little attention so far. In this paper, we revisit the formalism of epistemic specification. We offer a new definition of the formalism, propose several semantics (one of which, under syntactic restrictions we assume, turns out to be equivalent to the original semantics by Gelfond), derive some complexity results and, finally, show the effectiveness of the formalism for modeling problems requiring meta-reasoning considered recently by Faber and Woltran. All these results show that epistemic specifications deserve much more attention that has been afforded to them so far.

1 Introduction

Early 1990s were marked by several major developments in knowledge representation and nonmonotonic reasoning. One of the most important among them was the introduction of *disjunctive logic programs with classical negation* by Michael Gelfond and Vladimir Lifschitz [1]. The language of the formalism allowed for rules

$$H_1 \vee \ldots \vee H_k \leftarrow B_1, \ldots, B_m, not\ B_{m+1}, \ldots, not\ B_n,$$

where H_i and B_i are classical literals, that is, atoms and classical or *strong* negations ($\neg$) of atoms. In the paper, we will write "strong" rather than "classical" negation, as it reflects more accurately the role and the behavior of the operator. The *answer-set* semantics for programs consisting of such rules, introduced in the same paper, generalized the stable-model semantics of normal logic programs proposed a couple of years earlier also by Gelfond and Lifschitz [2]. The proposed extensions of the language of normal logic programs were motivated by knowledge representation considerations. With two negation operators it was straightforward to distinguish between P being *false by default* (there is no justification for adopting P), and P being *strongly false* (there is evidence for $\neg P$). The former would be written as $not\ P$ while the latter as $\neg P$. And with the disjunction in the head of rules one could model "indefinite" rules which, when applied, provide partial information only (one of the alternatives in the head holds, but no preference to any of them is given).

Soon after disjunctive logic programs with strong negation were introduced, Michael Gelfond proposed an additional important extension, this time with a modal operator [3]. He called the resulting formalism the language of *epistemic specifications*. The motivation came again from knowledge representation. The goal was to provide means for the "correct representation of incomplete information in the presence of multiple extensions" [3].

Surprisingly, despite their evident relevance to the theory of nonmonotonic reasoning as well as to the practice of knowledge representation, epistemic specifications have received relatively little attention so far. This state of affairs may soon change. Recent work by Faber and Woltran on *meta-reasoning* with answer-set programming [4, 5] shows the need for languages, in which one could express properties holding across all answer sets of a program, something Michael Gelfond foresaw already two decades ago.

Our goal in this paper is to revisit the formalism of epistemic specifications and show that they deserve a second look, in fact, a place in the forefront of knowledge representation research. We will establish a general semantic framework for the formalism, and identify in it the precise location of Gelfond's epistemic specifications. We will derive several complexity results. We will also show that the original idea of Gelfond to use a modal operator to model "what is known to a reasoner" has a broader scope of applicability. In particular, we will show that it can also be used in combination with the classical logic.

Complexity results presented in this paper provide an additional motivation to study epistemic specifications. Even though programs with strong negation often look "more natural" as they more directly align with the natural language description of knowledge specifications, the extension of the language of normal logic programs with the strong negation operator does not actually increase the expressive power of the formalism. This point was made already by Gelfond and Lifschitz, who observed that there is a simple and concise way to compile the strong negation away. On the other hand, the extension allowing the disjunction operator in the heads of rules is an essential one. As the complexity results show [6, 7], the class of problems that can be represented by means of disjunctive logic programs is strictly larger (assuming no collapse of the polynomial hierarchy) than the class of problems that can be modeled by normal logic programs. In the same vein, extension by the modal operator along the lines proposed by Gelfond is essential, too. It does lead to an additional jump in the complexity.

2 Epistemic Specifications

To motivate epistemic specifications, Gelfond discussed the following example. A certain college has these rules to determine the eligibility of a student for a scholarship:

1. Students with high GPA are eligible
2. Students from underrepresented groups and with fair GPA are eligible

3. Students with low GPA are not eligible
4. When these rules are insufficient to determine eligibility, the student should be interviewed by the scholarship committee.

Gelfond argued that there is no simple way to represent these rules as a disjunctive logic program with strong negation. There is no problem with the first three rules. They are modeled correctly by the following three logic program rules (in the language with both the default and strong negation operators):

1. $eligible(X) \leftarrow highGPA(X)$
2. $eligible(X) \leftarrow underrep(X), fairGPA(X)$
3. $\neg eligible(X) \leftarrow lowGPA(X)$.

The problem is with the fourth rule, as it has a clear meta-reasoning flavor. It should apply when the possible worlds (answer sets) determined by the first three rules do not fully specify the status of eligibility of a student a: neither *all* of them contain $eligible(a)$ nor *all* of them contain $\neg eligible(a)$. An obvious attempt at a formalization:

4. $interview(X) \leftarrow not\ eligible(X), not\ \neg eligible(X)$

fails. It is just another rule to be added to the program. Thus, when the answer-set semantics is used, the rule is interpreted with respect to individual answer sets and not with respect to collections of answer-sets, as required for this application. For a concrete example, let us assume that all we know about a certain student named Mike is that Mike's GPA is fair or high. Clearly, we do not have enough information to determine Mike's eligibility and so we must interview Mike. But the program consisting of rules (1)-(4) and the statement

5. $fairGPA(mike) \vee highGPA(mike)$

about Mike's GPA, has two answer sets:

$$\{highGPA(mike), eligible(mike)\}$$
$$\{fairGPA(mike), interview(mike)\}.$$

Thus, the query $?interview(mike)$ has the answer "unknown." To address the problem, Gelfond proposed to extend the language with a modal operator K and, speaking informally, interpret premises $K\varphi$ as "φ is known to the program" (the original phrase used by Gelfond was "known to the reasoner"), that is, true in all answer-sets. With this language extension, the fourth rule can be encoded as

4′. $interview(X) \leftarrow not\ K\, eligible(X), not\ K\neg eligible(X)$

which, intuitively, stands for "*interview* if neither the eligibility nor the non-eligibility is known."

The way in which Gelfond [3] proposed to formalize this intuition is strikingly elegant. We will now discuss it. We start with the syntax of *epistemic specifications*. As elsewhere in the paper, we restrict attention to the propositional case. We assume a fixed infinite countable set At of *atoms* and the corresponding language $\mathcal{L}$ of propositional logic. A *literal* is an atom, say A, or its *strong* negation $\neg A$. A *simple modal atom* is an expression $K\varphi$, where $\varphi \in \mathcal{L}$, and a *simple modal literal* is defined accordingly. An *epistemic premise* is an expression (conjunction)

$$E_1, \ldots, E_s, not\ E_{s+1}, \ldots, not\ E_t,$$

where every E_i, $1 \leq i \leq t$, is a simple modal literal. An *epistemic rule* is an expression of the form

$$L_1 \vee \ldots \vee L_k \leftarrow L_{k+1}, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n, E,$$

where every L_i, $1 \leq i \leq k$, is a literal, and E is an epistemic premise. Collections of epistemic rules are *epistemic programs*. It is clear that (ground versions of) rules (1)-(5) and (4′) are examples of epistemic rules, with rule (4′) being an example of an epistemic rule that actually takes advantage of the extended syntax. Rules such as

$$a \vee \neg d \leftarrow b, not\ \neg c, \neg K(d \vee \neg c)$$
$$\neg a \leftarrow \neg c, not\ \neg K(\neg(a \wedge c) \rightarrow b)$$

are also examples of epistemic rules. We note that the language of epistemic programs is only a fragment of the language of epistemic specifications by Gelfond. However, it is still expressive enough to cover all examples discussed by Gelfond and, more generally, a broad range of practical applications, as natural-language formulations of domain knowledge typically assume a rule-based pattern.

We move on to the semantics, which is in terms of *world views*. The definition of a world view consists of several steps. First, let W be a consistent set of literals from $\mathcal{L}$. We regard W as a three-valued interpretation of $\mathcal{L}$ (we will also use the term *three-valued possible world*), assigning to each atom one of the three logical values $\mathbf{t}$, $\mathbf{f}$ and $\mathbf{u}$. The interpretation extends by recursion to all formulas in $\mathcal{L}$, according to the following truth tables

$\neg$	
f	**t**
t	**f**
u	**u**

$\vee$	**t**	**u**	**f**
t	**t**	**t**	**t**
u	**t**	**u**	**u**
f	**t**	**u**	**f**

$\wedge$	**t**	**u**	**f**
t	**t**	**u**	**f**
u	**u**	**u**	**f**
f	**f**	**f**	**f**

$\rightarrow$	**t**	**u**	**f**
t	**t**	**u**	**f**
u	**t**	**u**	**u**
f	**t**	**t**	**t**

Fig. 1. Truth tables for the 3-valued logic of Kleene.

By a *three-valued possible-world structure* we mean a non-empty family of consistent sets of literals (three-valued possible worlds). Let $\mathcal{A}$ be a three-valued possible-world structure and let W be a consistent set of literals. For every formula $\varphi \in \mathcal{L}$, we define

1. $\langle \mathcal{A}, W\rangle \models \varphi$, if $v_W(\varphi) = \mathbf{t}$
2. $\langle \mathcal{A}, W\rangle \models K\varphi$, if for every $V \in \mathcal{A}$, $v_V(\varphi) = \mathbf{t}$
3. $\langle \mathcal{A}, W\rangle \models \neg K\varphi$, if there is $V \in \mathcal{A}$ such that $v_V(\varphi) = \mathbf{f}$.

Next, for every literal or simple modal literal L, we define

4. $\langle \mathcal{A}, W\rangle \models \mathit{not}\ L$ if $\langle \mathcal{A}, W\rangle \not\models L$.

We note that neither $\langle \mathcal{A}, W\rangle \models K\varphi$ nor $\langle \mathcal{A}, W\rangle \models \neg K\varphi$ depend on W. Thus, we will often write $\mathcal{A} \models F$, when F is a simple modal literal or its default negation.

In the next step, we introduce the notion of the *G-reduct* of an epistemic program.

Definition 1. *Let P be an epistemic program, $\mathcal{A}$ a three-valued possible-world structure and W a consistent set of literals. The* G-reduct *of P with respect to $\langle \mathcal{A}, W\rangle$, in symbols $P^{\langle \mathcal{A}, W\rangle}$, consists of the heads of all rules $r \in P$ such that $\langle \mathcal{A}, W\rangle \models \alpha$, for every conjunct α occurring in the body of r.*

Let H be a set of disjunctions of literals from $\mathcal{L}$. A set W of literals is *closed* with respect to H if W is consistent and contains at least one literal in common with every disjunction in H. We denote by $\mathit{Min}(H)$ the family of all minimal sets of literals that are closed with respect to H. With the notation $\mathit{Min}(H)$ in hand, we are finally ready to define the concept of a world view of an epistemic program P.

Definition 2. *A three-valued possible-world structure $\mathcal{A}$ is a* world view *of an epistemic program P if $\mathcal{A} = \{W \mid W \in \mathit{Min}(P^{\langle \mathcal{A}, W\rangle})\}$.*

Remark 1. The G-reduct of an epistemic program consists of disjunctions of literals. Thus, the concept of a world view is well defined.

Remark 2. We note that Gelfond considered also inconsistent sets of literals as minimal sets closed under disjunctions. However, the only such set he allowed consisted of *all* literals. Consequently, the difference between the Gelfond's semantics and the one we described above is that some programs have a world view in the Gelfond's approach that consists of a single set of all literals, while in our approach these programs do not have a world view. But in all other cases, the two semantics behave in the same way.

Let us consider the ground program, say P, corresponding to the scholarship eligibility example (rule (5), and rules (1)-(3) and (4′), grounded with respect to the Herbrand universe $\{\mathit{mike}\}$). The only rule involving simple modal literals is

$$\mathit{interview}(\mathit{mike}) \leftarrow \mathit{not}\ K\, \mathit{eligible}(\mathit{mike}), \mathit{not}\ K\neg \mathit{eligible}(\mathit{mike}).$$

Let $\mathcal{A}$ be a world view of P. Being a three-valued possible-world structure, $\mathcal{A} \neq \emptyset$. No matter what W we consider, no minimal set closed with respect to $P^{\langle \mathcal{A}, W\rangle}$ contains $\mathit{lowGPA}(\mathit{mike})$ and, consequently, no minimal set closed with respect to $P^{\langle \mathcal{A}, W\rangle}$ contains $\neg \mathit{eligible}(\mathit{mike})$. It follows that $\mathcal{A} \not\models K\neg \mathit{eligible}(\mathit{mike})$.

Let us assume that $\mathcal{A} \models K\,eligible(mike)$. Then, no reduct $P^{\langle \mathcal{A}, W \rangle}$ contains $interview(mike)$. Let $W = \{fairGP(mike)\}$. It follows that $P^{\langle \mathcal{A}, W \rangle}$ consists only of $fairGPA(mike) \vee highGPA(mike)$. Clearly, $W \in Min(P^{\langle \mathcal{A}, W \rangle})$ and, consequently, $W \in \mathcal{A}$. Thus, $\mathcal{A} \not\models K\,eligible(mike)$, a contradiction.

It must be then that $\mathcal{A} \models not\ K\,eligible(mike)$ and $\mathcal{A} \models not\ K\neg eligible(mike)$. Let W be an arbitrary consistent set of literals. Clearly, the reduct $P^{\langle \mathcal{A}, W \rangle}$ contains $interview(mike)$ and $fairGPA(mike) \vee highGPA(mike)$. If $highGPA(mike) \in W$, the reduct also contains $eligible(mike)$. Thus, $W \in Min(P^{\langle \mathcal{A}, W \rangle})$ if and only if

$W = \{fairGPA(mike), interview(mike)\}$, or
$W = \{highGPA(mike), eligible(mike), interview(mike)\}$.

It follows that if $\mathcal{A}$ is a world view for P then it consists of these two possible worlds. Conversely, it is easy to check that a possible-world structure consisting of these two possible worlds is a world view for P. Thus, $interview(mike)$ holds in $\mathcal{A}$, and so our representation of the example as an epistemic program has the desired behavior.

3 Epistemic Specifications — a Broader Perspective

The discussion in the previous section demonstrates the usefulness of formalisms such as that of epistemic specifications for knowledge representation and reasoning. We will now present a simpler yet, in many respects, more general framework for epistemic specifications. The key to our approach is that we consider the semantics given by *two-valued* interpretations (sets of atoms), and standard *two-valued* possible-world structures (nonempty collections of two-valued interpretations). We also work within a rather standard version of the language of modal propositional logic and so, in particular, we allow only for one negation operator. Later in the paper we show that epistemic specifications by Gelfond can be encoded in a rather direct way in our formalism. Thus, the restrictions we impose are not essential even though, admittedly, not having two kinds of negation in the language in some cases may make the modeling task harder.

We start by making precise the syntax of the language we will be using. As we stated earlier, we assume a fixed infinite countable set of atoms At. The language we consider is determined by the set At, the modal operator K, and by the *boolean connectives* $\bot$ (0-place), and $\wedge$, $\vee$, and $\rightarrow$ (binary). The BNF expression

$$\varphi ::= \bot \mid A \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid K\varphi,$$

where $A \in At$, provides a concise definition of a formula. The parentheses are used only to disambiguate the order of binary connectives. Whenever possible, we omit them. We define the unary *negation* connective $\neg$ and the 0-place connective $\top$ as abbreviations:

$$\neg\varphi ::= \varphi \rightarrow \bot$$
$$\top ::= \neg\bot.$$

We call formulas $K\varphi$, where $\varphi \in \mathcal{L}_K$, *modal atoms* (simple modal atoms that we considered earlier and will consider below are special modal atoms with K-depth equal to 1). We denote this language by $\mathcal{L}_K$ and refer to subsets of $\mathcal{L}_K$ as *epistemic theories*. We denote the modal-free fragment of $\mathcal{L}_K$ by $\mathcal{L}$.

While we will eventually describe the semantics (in fact, several of them) for arbitrary epistemic theories, we start with an important special case. Due to close analogies between the concepts we define below and the corresponding ones defined earlier in the context of the formalism of Gelfond, we "reuse" the terms used there. Specifically, by an *epistemic premise* we mean a conjunction of simple modal literals. Similarly, by an *epistemic rule* we understand an expression of the form

$$E \wedge L_1 \wedge \ldots \wedge L_m \rightarrow A_1 \vee \ldots \vee A_n, \tag{1}$$

where E is an epistemic premise, L_i's are literals (in $\mathcal{L}$) and A_i's are atoms. Finally, we call a collection of epistemic rules an *epistemic program*. It will always be clear from the context, in which sense these terms are to be understood.

We stress that $\neg$ is not a primary connective in the language but a derived one (it is a shorthand for some particular formulas involving the rule symbol). Even though under some semantics we propose below this negation operator has features of default negation, under some others it does not. Thus, we selected for it the standard negation symbol $\neg$ rather than the "loaded" not .

A (two-valued) *possible-world structure* is any nonempty family $\mathcal{A}$ of subsets of At (two-valued interpretations). In the remainder of the paper, when we use terms "interpretation" and "possible-world structure" without any additional modifiers, we always mean a two-valued interpretation and a two-valued possible-world structure.

Let $\mathcal{A}$ be a possible-world structure and $\varphi \in \mathcal{L}$. We recall that $\mathcal{A} \models K\varphi$ precisely when $W \models \varphi$, for every $W \in \mathcal{A}$, and $\mathcal{A} \models \neg K\varphi$, otherwise. We will now define the *epistemic reduct* of an epistemic program with respect to a possible-world structure.

Definition 3. *Let $P \subseteq \mathcal{L}_K$ be an epistemic program and let $\mathcal{A}$ be a possible-world structure. The* epistemic reduct *of P with respect to $\mathcal{A}$, $P^{\mathcal{A}}$ in symbols, is the theory obtained from P as follows: eliminate every rule with an epistemic premise E such that $\mathcal{A} \not\models E$; drop the epistemic premise from every remaining rule.*

It is clear that $P^{\mathcal{A}} \subseteq \mathcal{L}$, and that it consists of rules of the form

$$L_1 \wedge \ldots \wedge L_m \rightarrow A_1 \vee \ldots \vee A_n, \tag{2}$$

where L_i's are literals (in $\mathcal{L}$) and A_i's are atoms.

Let P be a collection of rules (2). Then, P is a propositional theory. Thus, it can be interpreted by the standard propositional logic semantics. However, P can also be regarded as a disjunctive logic program (if we write rules from right to left rather than from left to right). Consequently, P can also be interpreted by the stable-model semantics [2, 1] and the supported-model semantics [8–11]. (For normal logic programs, the supported-model semantics was introduced by Apt et al. [8]. The notion was extended to disjunctive logic programs by Baral and Gelfond [9]. We refer to papers by Brass and

Dix [10], Definition 2.4, and Inoue and Sakama [11], Section 5, for more details). We write $\mathcal{M}(P)$, $\mathcal{ST}(P)$ and $\mathcal{SP}(P)$ for the sets of models, stable models and supported models of P, respectively. An important observation is that *each* of these semantics gives rise to the corresponding notion of an epistemic extension.

Definition 4. *Let $P \subseteq \mathcal{L}_K$ be an epistemic program. A possible-world structure $\mathcal{A}$ is an* epistemic model *(respectively, an* epistemic stable model*, or an* epistemic supported model*) of P, if $\mathcal{A} = \mathcal{M}(P^{\mathcal{A}})$ (respectively, $\mathcal{A} = \mathcal{ST}(P^{\mathcal{A}})$ or $\mathcal{A} = \mathcal{SP}(P^{\mathcal{A}})$).*

It is clear that Definition 4 can easily be adjusted also to other semantics of propositional theories and programs. We briefly mention two such semantics in the last section of the paper.

We will now show that epistemic programs with the semantics of epistemic stable models can provide an adequate representation to the scholarship eligibility example for Mike. The available information can be represented by the following program $P(mike) \subseteq \mathcal{L}_K$:

1. $eligible(mike) \wedge neligible(mike) \rightarrow \bot$
2. $fairGPA(mike) \vee highGPA(mike)$
3. $highGPA(mike) \rightarrow eligible(mike)$
4. $underrep(mike) \wedge fairGPA(mike) \rightarrow eligible(mike)$
5. $lowGPA(mike) \rightarrow neligible(mike)$
6. $\neg K\, eligible(mike), \neg K\, neligible(mike) \rightarrow interview(mike)$.

We use the predicate *neligible* to model the strong negation of the predicate $eligible$ that appears in the representation in terms of epistemic programs by Gelfond (thus, in particular, the presence of the first clause, which precludes the facts $eligible(mike)$ and $neligible(mike)$ to be true together). This extension of the language and an extra rule in the representation is the price we pay for eliminating one negation operator.

Let $\mathcal{A}$ consist of the interpretations

$W_1 = \{fairGPA(mike), interview(mike)\}$
$W_2 = \{highGPA(mike), eligible(mike), interview(mike)\}$.

Then the reduct $[P(mike)]^{\mathcal{A}}$ consists of rules (1)-(5), which are unaffected by the reduct operation, and of the fact $interview(mike)$, resulting from rule (6) when the reduct operation is performed (as in logic programming, when a rule has the empty antecedent, we drop the implication symbol from the notation). One can check that $\mathcal{A} = \{W_1, W_2\} = \mathcal{ST}([P(mike)]^{\mathcal{A}})$. Thus, $\mathcal{A}$ is an epistemic stable model of P (in fact, the only one). Clearly, $interview(mike)$ holds in the model (as we would expect it to), as it holds in each of its possible-worlds. We note that in this particular case, the semantics of epistemic supported models yields exactly the same solution.

4 Complexity

We will now study the complexity of reasoning with epistemic (stable, supported) models. We provide details for the case of epistemic stable models, and only present the results for the other two semantics, as the techniques to prove them are very similar to those we develop for the case of epistemic stable models.

First, we note that epistemic stable models of an epistemic program P can be represented by partitions of the set of all modal atoms of P. This is important as *a priori* the size of possible-world structures one needs to consider as candidates for epistemic stable models may be exponential in the size of a program. Thus, to obtain good complexity bounds alternative polynomial-size representations of epistemic stable models are needed.

Let $P \subseteq \mathcal{L}_K$ be an epistemic program and (Φ, Ψ) be the set of modal atoms of P (all these modal atoms are, in fact, simple). We write $P_{|\Phi,\Psi}$ for the program obtained from P by eliminating every rule whose epistemic premise contains a conjunct $K\psi$, where $K\psi \in \Psi$, or a conjunct $\neg K\varphi$, where $K\varphi \in \Phi$ (these rules are "'blocked" by (Φ, Ψ)), and by eliminating the epistemic premise from every other rule of P.

Proposition 1. *Let $P \subseteq \mathcal{L}_K$ be an epistemic program. If a possible-world structure $\mathcal{A}$ is an epistemic stable model of P, then there is a partition (Φ, Ψ) of the set of modal atoms of P such that*

1. $\mathcal{ST}(P_{|\Phi,\Psi}) \neq \emptyset$
2. *For every $K\varphi \in \Phi$, φ holds in every stable model of $P_{|\Phi,\Psi}$*
3. *For every $K\psi \in \Psi$, ψ does not hold in at least one stable model of $P_{|\Phi,\Psi}$.*

Conversely, if there are such partitions, P has epistemic stable models.

It follows that epistemic stable models can be represented by partitions (Φ, Ψ) satisfying conditions (1)-(3) from the proposition above.

We observe that deciding whether a partition (Φ, Ψ) satisfies conditions (1)-(3) from Proposition 1, can be accomplished by polynomially many calls to an Σ_2^P-oracle and, if we restrict attention to non-disjunctive epistemic programs, by polynomially many calls to an NP-oracle.

Remark 3. If we adjust Proposition 1 by replacing the term "stable" with the term "supported," and replacing $\mathcal{ST}()$ with $\mathcal{SP}()$, we obtain a characterization of epistemic supported models. Similarly, omitting the term "stable," and replacing $\mathcal{ST}()$ with $\mathcal{M}()$ yields a characterization of epistemic models. In each case, one can decide whether a partition (Φ, Ψ) satisfies conditions (1)-(3) by polynomially many calls to an NP-oracle (this claim is evident for the case of epistemic models; for the case of epistemic supported models, it follows from the fact that supported models semantics does not get harder when we allow disjunctions in the heads or rules).

Theorem 1. *The problem to decide whether a non-disjunctive epistemic program has an epistemic stable model is Σ_2^P-complete.*

Proof: Our comments above imply that the problem is in the class Σ_2^P. Let $F = \exists Y \forall Z \Theta$, where Θ is a DNF formula. The problem to decide whether F is true is Σ_2^P-complete. We will reduce it to the problem in question and, consequently, demonstrate its Σ_2^P-hardness. To this end, we construct an epistemic program $Q \subseteq \mathcal{L}_K$ by including into Q the following clauses (atoms w, y', $y \in Y$, and z', $z \in Z$ are fresh):

1. $Ky \rightarrow y$; and $Ky' \rightarrow y'$, for every $y \in Y$
2. $y \wedge y' \rightarrow$; and $\neg y \wedge \neg y' \rightarrow$, for every $y \in Y$
3. $\neg z' \rightarrow z$; and $\neg z \rightarrow z'$, for $z \in Z$
4. $\sigma(u_1) \wedge \ldots \wedge \sigma(u_k) \rightarrow w$, where $u_1 \wedge \ldots \wedge u_k$ is a disjunct of Θ, and $\sigma(\neg a) = a'$ and $\sigma(a) = a$, for every $a \in Y \cup Z$
5. $\neg Kw \rightarrow$.

Let us assume that $\mathcal{A}$ is an epistemic stable model of Q. In particular, $\mathcal{A} \neq \emptyset$. It must be that $\mathcal{A} \models Kw$ (otherwise, $Q^{\mathcal{A}}$ has no stable models, that is, $\mathcal{A} = \emptyset$). Let us define $A = \{y \in Y \mid \mathcal{A} \models Ky\}$, and $B = \{y \in Y \mid \mathcal{A} \models Ky'\}$. It follows that $Q^{\mathcal{A}}$ consists of the following rules:

1. y, for $y \in A$, and y', for $y \in B$
2. $y \wedge y' \rightarrow$; and $\neg y \wedge \neg y' \rightarrow$, for every $y \in Y$
3. $\neg z' \rightarrow z$; and $\neg z \rightarrow z'$, for $z \in Z$
4. $\sigma(u_1) \wedge \ldots \wedge \sigma(u_k) \rightarrow w$, where $u_1 \wedge \ldots \wedge u_k$ is a disjunct of Θ, and $\sigma(\neg a) = a'$ and $\sigma(a) = a$, for every $a \in Y \cup Z$.

Since $\mathcal{A} = \mathcal{ST}(Q^{\mathcal{A}})$ and $\mathcal{A} \neq \emptyset$, $B = Y \setminus A$ (due to clauses of type (2)). It is clear that the program $Q^{\mathcal{A}}$ has stable models and that they are of the form $A \cup \{y' \mid y \in Y \setminus A\} \cup D \cup \{z' \mid z \in Z \setminus D\}$, if that set does not imply w through a rule of type (4), or $A \cup \{y' \mid y \in Y \setminus A\} \cup D \cup \{z' \mid z \in Z \setminus D\} \cup \{w\}$, otherwise, where D is any subset of Z. As $\mathcal{A} \models Kw$, there are no stable models of the first type. Thus, the family of stable models of $Q^{\mathcal{A}}$ consists of all sets $A \cup \{y' \mid y \in Y \setminus A\} \cup D \cup \{z' \mid z \in Z \setminus D\} \cup \{w\}$, where D is an arbitrary subset of Z. It follows that for every $D \subseteq Z$, the set $A \cup \{y' \mid y \in Y \setminus A\} \cup D \cup \{z' \mid z \in Z \setminus D\}$ satisfies the body of at least one rule of type (4). By the construction, for every $D \subseteq Z$, the valuation of $Y \cup Z$ determined by A and D satisfies the corresponding disjunct in Θ and so, also Θ. In other words, $\exists Y \forall Z \Theta$ is true.

Conversely, let $\exists Y \forall Z \Theta$ be true. Let A be a subset of Y such that $\Theta_{|Y/A}$ holds for every truth assignment of Z (by $\Theta_{|Y/A}$, we mean the formula obtained by simplifying the formula Q with respect to the truth assignment of Y determined by A). Let $\mathcal{A}$ consist of all sets of the form $A \cup \{y' \mid y \in Y \setminus A\} \cup D \cup \{z' \mid z \in Z \setminus D\} \cup \{w\}$, where $D \subseteq Z$. It follows that $Q^{\mathcal{A}}$ consists of clauses (1)-(4) above, with $B = Y \setminus A$. Since $\forall Z \Theta_{|A/Y}$ holds, it follows that $\mathcal{A}$ is precisely the set of stable models of $Q^{\mathcal{A}}$. Thus, $\mathcal{A}$ is an epistemic stable model of Q. □

In the general case, the complexity goes one level up.

Theorem 2. *The problem to decide whether an epistemic program $P \subseteq \mathcal{L}_K$ has an epistemic stable model is Σ_3^P-complete.*

Proof: The membership follows from the earlier remarks. To prove the hardness part, we consider a QBF formula $F = \exists X \forall Y \exists Z \Theta$, where Θ is a 3-CNF formula. For each atom $x \in X$ ($y \in Y$ and $z \in Z$, respectively), we introduce a fresh atom x' (y' and z', respectively). Finally, we introduce three additional fresh atoms, w, f and g.

We now construct a disjunctive epistemic program Q by including into it the following clauses:

1. $Kx \rightarrow x$; and $Kx' \rightarrow x'$, for every $x \in X$
2. $x \wedge x' \rightarrow$; and $\neg x \wedge \neg x' \rightarrow$, for every $x \in X$
3. $\neg g \rightarrow f$; and $\neg f \rightarrow g$
4. $f \rightarrow y \vee y'$; and $f \rightarrow z \vee z'$, for every $y \in Y$ and $z \in Z$
5. $f \wedge w \rightarrow z$; and $f \wedge w \rightarrow z'$, for every $z \in Z$
6. $f \wedge \sigma(u_1) \wedge \sigma(u_2) \wedge \sigma(u_3) \rightarrow w$, for every clause $C = u_1 \vee u_2 \vee u_3$ of Θ, where $\sigma(a) = a'$ and $\sigma(\neg a) = a$, for every $a \in X \cup Y \cup Z$
7. $f \wedge \neg w \rightarrow w$
8. $\neg K \neg w \rightarrow$

Let us assume that $\exists X \forall Y \exists Z \Theta$ is true. Let $A \subseteq X$ describe the truth assignment on X so that $\forall Y \exists Z \Phi_{X/A}$ holds (we define $\Phi_{X/A}$ in the proof of the previous result). We will show that Q has an epistemic stable model $\mathcal{A} = \{A \cup \{a' \mid a \in X \setminus A\} \cup \{g\}\}$. Clearly, Kx, $x \in A$, and Kx', $x \in X \setminus A$, are true in $\mathcal{A}$. Also, $K\neg w$ is true in $\mathcal{A}$. All other modal atoms in Q are false in $\mathcal{A}$. Thus, $Q^{\mathcal{A}}$ consists of rules x, for $x \in A$, x', for $x \in X \setminus A$ and of rules (2)-(7) above. Let M be a stable model of $Q^{\mathcal{A}}$ containing f. It follows that $w \in M$ and so, $Z \cup Z' \subseteq M$. Moreover, the Gelfond-Lifschitz reduct of $Q^{\mathcal{A}}$ with respect to M consists of rules x, for $x \in A$, x', for $x \in X \setminus A$, all $\neg$-free constraints of type (2), rule f, and rules (4)-(6) above, and M is a minimal model of this program.

Let $B = Y \cap M$. By the minimality of M, $M = A \cup \{x' \mid x \in X \setminus A\} \cup B \cup \{y' \mid y \in Y \setminus B\} \cup Z \cup Z' \cup \{f, w\}$. Since $\forall Y \exists Z \Phi_{X/A}$ holds, $\exists Z \Phi_{X/A, Y/B}$ holds, too. Thus, let $D \subseteq Z$ be a subset of Z such that $\Phi_{X/A, Y/B, Z/D}$ is true. It follows that $M' = A \cup \{x' \mid x \in X \setminus A\} \cup B \cup \{y' \mid y \in Y \setminus B\} \cup D \cup \{z' \mid z \in Z \setminus D\} \cup \{f\}$ is also a model of the Gelfond-Lifschitz reduct of $Q^{\mathcal{A}}$ with respect to M, contradicting the minimality of M.

Thus, if M is an answer set of $Q^{\mathcal{A}}$, it must contain g. Consequently, it does not contain f and so no rules of type (4)-(7) contribute to it. It follows that $M = A \cup \{a' \mid a \in X \setminus A\} \cup \{g\}$ and, as it indeed is an answer set of $Q^{\mathcal{A}}$, $\mathcal{A} = \mathcal{ST}(Q^{\mathcal{A}})$. Thus, $\mathcal{A}$ is a epistemic stable model, as claimed.

Conversely, let as assume that Q has an epistemic stable model, say, $\mathcal{A}$. It must be that $\mathcal{A} \models K\neg w$ (otherwise, $Q^{\mathcal{A}}$ contains a contradiction and has no stable models). Let us define $A = \{x \in X \mid \mathcal{A} \models Kx\}$ and $B = \{x \in X \mid \mathcal{A} \models Kx'\}$. It follows that $Q^{\mathcal{A}}$ consists of the clauses:

1. x, for $x \in A$ and x', for $x \in B$
2. $x \wedge x' \rightarrow$; and $\neg x \wedge \neg x' \rightarrow$, for every $x \in X$
3. $\neg g \rightarrow f$; and $\neg f \rightarrow g$
4. $f \rightarrow y \vee y'$; and $f \rightarrow z \vee z'$, for every $y \in Y$ and $z \in Z$
5. $f \wedge w \rightarrow z$; and $f \wedge w \rightarrow z'$, for every $z \in Z$

6. $f \wedge \sigma(u_1) \wedge \sigma(u_2) \wedge \sigma(u_3) \rightarrow w$, for every clause $C = u_1 \vee u_2 \vee u_3$ of Φ, where $\sigma(a) = a'$ and $\sigma(\neg a) = a$, for every $a \in X \cup Y \cup Z$.
7. $f, \neg w \rightarrow w$

We have that $\mathcal{A}$ is precisely the set of stable models of this program. Since $\mathcal{A} \neq \emptyset$, $B = X \setminus A$. If M is a stable model of $Q^{\mathcal{A}}$ and contains f, then it contains w. But then, as $M \in \mathcal{A}$, $\mathcal{A} \not\models K\neg w$, a contradiction. It follows that there is no stable model containing f. That is, the program consisting of the following rules has no stable model:

1. x, for $x \in A$ and x', for $x \in X \setminus A$
2. $y \vee y'$; and $z \vee z'$, for every $y \in Y$ and $z \in Z$
3. $w \rightarrow z$; and $w \rightarrow z'$, for every $z \in Z$
4. $\sigma(u_1) \wedge \sigma(u_2) \wedge \sigma(u_3) \rightarrow w$, for every clause $C = u_1 \vee u_2 \vee u_3$ of Θ, where $\sigma(a) = a'$ and $\sigma(\neg a) = a$, for every $a \in X \cup Y \cup Z$.
5. $\neg w \rightarrow w$

But then, the formula $\forall Y \exists Z \Theta_{|X/A}$ is true and, consequently, the formula $\exists X \forall Y \exists Z \Theta$ is true, too. □

For the other two epistemic semantics, Remark 1 implies that the problem of the existence of an epistemic model (epistemic supported model) is in the class Σ_2^P. The Σ_2^P-hardness of the problem can be proved by similar techniques as those we used for the case of epistemic stable models. Thus, we have the following result.

Theorem 3. *The problem to decide whether an epistemic program $P \subseteq \mathcal{L}_K$ has an epistemic model (epistemic supported model, respectively) is Σ_2^P-complete.*

5 Modeling with Epistemic Programs

We will now present several problems which illustrate the advantages offered by the language of epistemic programs we developed in the previous two sections. Whenever we use predicate programs, we understand that their semantics is that of the corresponding ground programs.

First, we consider two graph problems related to the existence of Hamiltonian cycles. Let G be a directed graph. An edge in G is *critical* if it belongs to every hamiltonian cycle in G. The following problems are of interest:

1. Given a directed graph G, find the set of all critical edges of G
2. Given a directed graph G, and integers p and k, find a set R of no more than p new edges such that $G \cup R$ has no more than k critical edges.

Let $HC(vtx, edge)$ be any standard ASP encoding of the Hamiltonian cycle problem, in which predicates vtx and $edge$ represent G, and a predicate hc represents edges of a candidate hamiltonian cycle. We assume the rules of $HC(vtx, edge)$ are written from left to right so that they can be regarded as elements of $\mathcal{L}$. Then, simply adding to $HC(vtx, edge)$ the rule:

$$Khc(X,Y) \rightarrow critical(X,Y)$$

yields a correct representation of the first problem. We write $HC_{cr}(vtx, edge)$ to denote this program. Also, for a directed graph $G = (V, E)$, we define

$$D = \{vtx(v) \mid v \in V\} \cup \{edge(v, w) \mid (v, w) \in E\}.$$

We have the following result.

Theorem 4. *Let $G = (V, E)$ be a directed graph. If $HC_{cr}(vtx, edge) \cup D$ has no epistemic stable models, then every edge in G is critical (trivially). Otherwise, the epistemic program $HC_{cr}(vtx, edge) \cup D$ has a unique epistemic stable model $\mathcal{A}$ and the set $\{(v, w) \mid \mathcal{A} \models critical(u, v)\}$ is the set of critical edges in G.*

Proof (Sketch): Let H be the grounding of $HC_{cr}(vtx, edge) \cup D$. If H has no epistemic stable models, it follows that the "non-epistemic" part H' of H has no stable models (as no atom of the form $critical(x, y)$ appears in it). As H' encodes the existence of a hamiltonian cycle in G, it follows that G has no Hamiltonian cycles. Thus, trivially, every edge of G belongs to every Hamiltonian cycle of G and so, every edge of G is critical.

Thus, let us assume that $\mathcal{A}$ is an epistemic stable model of H. Also, let S be the set of all stable models of H' (they correspond to Hamiltonian cycles of G; each model contains, in particular, atoms of the form $hc(x, y)$, where (x, y) ranges over the edges of the corresponding Hamiltonian cycle). The reduct $H^{\mathcal{A}}$ consists of H' (non-epistemic part of H is unaffected by the reduct operation) and of C', a set of some facts of the form $critical(x, y)$. Thus, the stable models of the reduct are of the form $M \cup C'$, where $M \in S$. That is, $\mathcal{A} = \{M \cup C' \mid M \in S\}$. Let us denote by C the set of the atoms $critical(x, y)$, where (x, y) belongs to every hamiltonian cycle of G (is critical). One can compute now that $H^{\mathcal{A}} = H' \cup C$. Since $\mathcal{A} = \mathcal{ST}(H^{\mathcal{A}})$, $\mathcal{A} = \{M \cup C \mid M \in S\}$. Thus, $HC_{cr}(vtx, edge) \cup D$ has a unique epistemic stable model, as claimed. It also follows that the set $\{(v, w) \mid \mathcal{A} \models critical(u, v)\}$ is the set of critical edges in G. □

To represent the second problem, we proceed as follows. First, we "select" new edges to be added to the graph and impose constraints that guarantee that all new edges are indeed new, and that no more than p new edges are selected (we use here *lparse* syntax for brevity; the constraint can be encoded strictly in the language $\mathcal{L}_K$).

$$vtx(X) \wedge vtx(Y) \rightarrow newEdge(X,Y)$$
$$newEdge(X,Y) \wedge edge(X,Y) \rightarrow \bot$$
$$(p+1)\{newEdge(X,Y) : vtx(X), vtx(Y)\} \rightarrow \bot.$$

Next, we define the set of edges of the extended graph, using a predicate $edgeEG$:

$$edge(X,Y) \rightarrow edgeEG(X,Y)$$
$$newEdge(X,Y) \rightarrow edgeEG(X,Y)$$

Finally, we define critical edges and impose a constraint on their number (again, exploiting the *lparse* syntax for brevity sake):

$edgeEG(X, Y) \wedge Khc(X, Y) \rightarrow critical(X, Y)$
$(k+1)\{critical(X, Y) : edgeEG(X, Y)\} \rightarrow \bot.$

We define Q to consist of all these rules together with all the rules of the program $HC(vtx, edgeEG)$. We now have the following theorem. The proof is similar to that above and so we omit it.

Theorem 5. *Let G be a directed graph. There is an extension of G with no more than p new edges so that the resulting graph has no more than k critical edges if and only if the program $Q \cup D$ has an epistemic stable model.*

For another example we consider the unique model problem: given a CNF formula F, the goal is to decide whether F has a unique minimal model. The unique model problem was also considered by Faber and Woltran [4, 5]. We will show two encodings of the problem by means of epistemic programs. The first one uses the semantics of epistemic models and is especially direct. The other one uses the semantics of epistemic stable models.

Let F be a propositional theory consisting of constraints $L_1 \wedge \ldots \wedge L_k \rightarrow \bot$, where L_i's are literals. Any propositional theory can be rewritten into an equivalent theory of such form. We denote by F^K the formula obtained from F by replacing every atom x with the modal atom Kx.

Theorem 6. *For every theory $F \subseteq \mathcal{L}$ consisting of constraints, F has a least model if and only if the epistemic program $F \cup F^K$ has an epistemic model.*

Proof: Let us assume that F has a least model. We define $\mathcal{A}$ to consist of all models of F, and we denote the least model of F by M. We will show that $\mathcal{A}$ is an epistemic model of $F \cup F^K$. Clearly, for every $x \in M$, $\mathcal{A} \models Kx$. Similarly, for every $x \notin M$, $\mathcal{A} \models \neg Kx$. Thus, $[F^K]^{\mathcal{A}} = \emptyset$. Consequently, $[F \cup F^K]^{\mathcal{A}} = F$ and so, $\mathcal{A}$ is precisely the set of all models of $[F \cup F^K]^{\mathcal{A}}$. Thus, $\mathcal{A}$ is an epistemic model.

Conversely, let $\mathcal{A}$ be an epistemic model of $F \cup F^K$. It follows that $[F^K]^{\mathcal{A}} = \emptyset$ (otherwise, $[F \cup F^K]^{\mathcal{A}}$ contains $\bot$ and $\mathcal{A}$ would have to be empty, contradicting the definition of an epistemic model). Thus, $[F \cup F^K]^{\mathcal{A}} = F$ and consequently, $\mathcal{A}$ is the set of all models of F. Let $M = \{x \in At \mid \mathcal{A} \models Kx\}$ and let

$$a_1 \wedge \ldots \wedge a_m \wedge \neg b_1 \wedge \ldots \wedge \neg b_n \rightarrow \bot \quad (3)$$

be a rule in F. Then,

$$Ka_1 \wedge \ldots \wedge Ka_m \wedge \neg Kb_1 \wedge \ldots \wedge \neg Kb_n \rightarrow \bot$$

is a rule in F^K. As $[F^K]^{\mathcal{A}} = \emptyset$,

$$\mathcal{A} \not\models Ka_1 \wedge \ldots \wedge Ka_m \wedge \neg Kb_1 \wedge \ldots \wedge \neg Kb_n.$$

Thus, for some i, $1 \leq i \leq m$, $\mathcal{A} \not\models Ka_i$, or for some j, $1 \leq j \leq n$, $\mathcal{A} \models Kb_j$. In the first case, $a_i \notin M$, in the latter, $b_j \in M$. In either case, M is a model of rule (3).

It follows that M is a model of F. Let M' be a model of F. Then $M' \in \mathcal{A}$ and, by the definition of M, $M \subseteq M'$. Thus, M is a least model of F. $\square$

Next, we will encode the same problem as an epistemic program under the epistemic stable model semantics. The idea is quite similar. We only need to add rules to generate all candidate models.

Theorem 7. *For every theory $F \subseteq \mathcal{L}$ consisting of constraints, F has a least model if and only if the epistemic program*

$$F \cup F^K \cup \{\neg x \rightarrow x' \mid x \in At\} \cup \{\neg x' \rightarrow x \mid x \in At\}$$

has an epistemic stable model.

We note that an even simpler encoding can be obtained if we use *lparse* choice rules. In this case, we can replace $\{\neg x \rightarrow x' \mid x \in At\} \cup \{\neg x' \rightarrow x \mid x \in At\}$ with $\{\{x\} \mid x \in At\}$.

6 Connection to Gelfond's Epistemic Programs

We will now return to the original formalism of epistemic specifications proposed by Gelfond [3] (under the restriction to epistemic programs we discussed here). We will show that it can be expressed in a rather direct way in terms of our epistemic programs in the two-valued setting and under the epistemic supported-model semantics.

The reduction we are about to describe is similar to the well-known one used to eliminate the "strong" negation from disjunctive logic programs with strong negation. In particular, it requires an extension to the language $\mathcal{L}$. Specifically, for every atom $x \in At$ we introduce a fresh atom x' and we denote the extended language by $\mathcal{L}'$. The intended role of x' is to represent in $\mathcal{L}'$ the literal $\neg x$ from $\mathcal{L}$. Building on this idea, we assign to each set W of literals in $\mathcal{L}$ the set

$$W' = (W \cap At) \cup \{x' \mid \neg x \in W\}.$$

In this way, sets of literals from $\mathcal{L}$ (in particular, three-valued interpretations of $\mathcal{L}$) are represented as sets of atoms from $\mathcal{L}'$ (two-valued interpretations of $\mathcal{L}'$).

We now note that the truth and falsity of a formula form $\mathcal{L}$ under a three-valued interpretation can be expressed as the truth and falsity of certain formulas from $\mathcal{L}'$ in the two-valued setting. The following result is well known.

Proposition 2. *For every formula $\varphi \in \mathcal{L}$ there are formulas $\varphi^-, \varphi^+ \in \mathcal{L}'$ such that for every set of literals W (in $\mathcal{L}$)*

1. *$v_W(\varphi) = \mathbf{t}$ if and only if $u_{W'}(\varphi^+) = \mathbf{t}$*
2. *$v_W(\varphi) = \mathbf{f}$ if and only if $u_{W'}(\varphi^-) = \mathbf{f}$*

Moreover, the formulas φ^- and φ^+ can be constructed in polynomial time with respect to the size of φ.

Proof: This a folklore result. We provide a sketch of a proof for the completeness sake. We define φ^+ and φ^- by recursively as follows:

1. $x^+ = x$ and $x^- = \neg x'$, if $x \in At$
2. $(\neg\varphi)^+ = \neg\varphi^-$ and $(\neg\varphi)^- = \neg\varphi^+$
3. $(\varphi \vee \psi)^+ = \varphi^+ \vee \psi^+$ and $(\varphi \vee \psi)^- = \varphi^- \vee \psi^-$; the case of the conjunction is dealt with analogously
4. $(\varphi \rightarrow \psi)^+ = \varphi^- \rightarrow \psi^+$ and $(\varphi \rightarrow \psi)^- = \varphi^+ \rightarrow \psi^-$.

One can check that formulas φ^+ and φ^- defined in this way satisfy the assertion. □

We will now define the transformation σ that allows us to eliminate strong negation. First, for a literal $L \in \mathcal{L}$, we now define

$$\sigma(L) = \begin{cases} x & \text{if } L = x \\ x' & \text{if } L = \neg x \end{cases}$$

Furthermore, if E is a simple modal literal or its default negation, we define

$$\sigma(E) = \begin{cases} K\varphi^+ & \text{if } E = K\varphi \\ \neg K\varphi^- & \text{if } E = \neg K\varphi \\ \neg K\varphi^+ & \text{if } E = \mathit{not}\ K\varphi \\ K\varphi^- & \text{if } E = \mathit{not}\ \neg K\varphi \end{cases}$$

and for an epistemic premise $E = E_1, \ldots, E_t$ (where each E_i is a simple modal literal or its default negation) we set

$$\sigma(E) = \sigma(E_1) \wedge \ldots \wedge \sigma(E_t).$$

Next, if r is an epistemic rule

$$L_1 \vee \ldots \vee L_k \leftarrow F_1, \ldots, F_m, \mathit{not}\ F_{m+1}, \ldots, \mathit{not}\ F_n, E$$

we define

$$\sigma(r) = \sigma(E) \wedge \sigma(F_1) \wedge \ldots \wedge \sigma(F_m) \wedge \neg\sigma(F_{m+1}) \wedge \ldots \wedge \neg\sigma(F_n) \rightarrow \sigma(L_1) \vee \ldots \vee \sigma(L_k).$$

Finally, for an epistemic program P, we set

$$\sigma(P) = \{\sigma(r) \mid r \in P\}) \cup \{x \wedge x' \rightarrow \bot\}.$$

We note that $\sigma(P)$ is indeed an epistemic program in the language $\mathcal{L}_K$ (according to our definition of epistemic programs). The role of the rules $x \wedge x' \rightarrow \bot$ is to ensure that sets forming epistemic (stable, supported) models of $\sigma(P)$ correspond to consistent sets of literals (the only type of set of literals allowed in world views).

Given a three-valued possible structure $\mathcal{A}$, we define $\mathcal{A}' = \{W' \mid W \in \mathcal{A}\}$, and we regard $\mathcal{A}'$ as a two-valued possible-world structure. We now have the following theorem.

Theorem 8. *Let P be an epistemic program according to Gelfond. Then a three-valued possible-world structure $\mathcal{A}$ is a world view of P if and only if a two-valued possible-world structure $\mathcal{A}'$ is an epistemic supported model of $\sigma(P)$.*

Proof (Sketch): Let P be an epistemic program according to Gelfond, $\mathcal{A}$ a possible-world structure and W a set of literals. We first observe that the G-reduct $P^{\langle \mathcal{A}, W \rangle}$ can be described as the result of a certain two-step process. Namely, we define the *epistemic reduct* of P with respect to $\mathcal{A}$ to be the disjunctive logic program $P^{\mathcal{A}}$ obtained from P by removing every rule whose epistemic premise E satisfies $\mathcal{A} \not\models E$, and by removing the epistemic premise from every other rule in P. This construction is the three-valued counterpart to the one we employ in our approach. It is clear that the epistemic reduct of P with respect to $\mathcal{A}$, with some abuse of notation we will denote it by $P^{\mathcal{A}}$, is a disjunctive logic program with strong negation.

Let Q be a disjunctive program with strong negation and W a set of literals. By the *supp-reduct* of Q with respect to W, $R^{sp}(Q, W)$, we mean the set of the heads of all rules whose bodies are satisfied by W (which in the three-valued setting means that every literal in the body not in the scope of not is in W, and every literal in the body in the scope of not is not in W). A consistent set W of literals is a supported answer set of Q if $W \in Min(R^{sp}(Q, W))$ (this is a natural extension of the definition of a supported model [8, 9] to the case of disjunctive logic programs with strong negation; again, we do not regard inconsistent sets of literals as supported answer sets).

Clearly, $P^{\langle \mathcal{A}, W \rangle} = R^{sp}(P^{\mathcal{A}}, W)$. Thus, $\mathcal{A}$ is a world view of P according to the definition by Gelfond if and only if $\mathcal{A}$ is a collection of all supported answer sets of $P^{\mathcal{A}}$.

We also note that by Proposition 2, if E is an epistemic premise, then $\mathcal{A} \models E$ if and only if $\mathcal{A}' \models \sigma(E)$. It follows that $\sigma(P^{\mathcal{A}}) = \sigma(P)^{\mathcal{A}'}$. In other words, constructing the epistemic reduct of P with respect to $\mathcal{A}$ and then translating the resulting disjunctive logic program with strong negation into the corresponding disjunctive logic program without strong negation yields the same result as first translating the epistemic program (in the Gelfond's system) into our language of epistemic programs and then computing the reduct with respect to $\mathcal{A}'$. We note that there is a one-to-one correspondence between supported answer sets of $P^{\mathcal{A}}$ and supported models of $\sigma(P^{\mathcal{A}})$ (σ, when restricted to programs consisting of rules without epistemic premises, is the standard transformation eliminating strong negation and preserving the stable and supported semantics). Consequently, there is a one-to-one correspondence between supported answer sets of $P^{\mathcal{A}}$ and supported models of $\sigma(P)^{\mathcal{A}'}$ (cf. our observation above). Thus, $\mathcal{A}$ consists of supported answer sets of $P^{\mathcal{A}}$ if and only if $\mathcal{A}'$ consists of supported models of $\sigma(P)^{\mathcal{A}'}$. Consequently, $\mathcal{A}$ is a world view of P if and only if $\mathcal{A}'$ is an epistemic supported model of $\sigma(P)$. □

7 Epistemic Models of Arbitrary Theories

So far, we defined the notions of epistemic models, epistemic stable models and epistemic supported models only for the case of epistemic programs. However, this restriction is not essential. We recall that the definition of these three epistemic semantics consists of two steps. The first step produces the reduct of an epistemic program P with respect to a possible-world structure, say $\mathcal{A}$. This reduct happens to be (modulo a trivial syntactic transformation) a standard disjunctive logic program in the language $\mathcal{L}$ (no modal atoms anymore). If the set of models (respectively, stable models, supported models) of the reduct program coincides with $\mathcal{A}$, $\mathcal{A}$ is an epistemic model (respectively, epistemic stable or supported model) of P. However, the concepts of a model, stable model and supported model are defined for *arbitrary* theories in $\mathcal{L}$. This is obviously well known for the semantics of models. The stable-model semantics was extended to the full language $\mathcal{L}$ by Ferraris [12] and the supported-model semantics by Truszczynski [13]. Thus, there is no reason precluding the extension of the definition of the corresponding epistemic types of models to the general case. We start be generalizing the concept of the reduct.

Definition 5. *Let T be an arbitrary theory in $\mathcal{L}_K$ and let $\mathcal{A}$ be a possible-world structure. The* epistemic reduct *of T with respect to $\mathcal{A}$, $T^{\mathcal{A}}$ in symbols, is the theory obtained from T by replacing each maximal modal atom $K\varphi$ with $\top$, if $\mathcal{A} \models K\varphi$, and with $\bot$, otherwise.*

We note that if T is an epistemic program, this notion of the reduct does not coincide with the one we discussed before. Indeed, now no rule is dropped and no modal literals are dropped; rather modal atoms are replaced with $\top$ and $\bot$. However, the replacements are executed in such a way as to ensure the same behavior. Specifically, one can show that models, stable models and supported models of the two reducts coincide.

Next, we generalize the concepts of the three types of epistemic models.

Definition 6. *Let T be an arbitrary theory in $\mathcal{L}_K$. A possible-world structure $\mathcal{A}$ is an* epistemic model *(respectively, an* epistemic stable model*, or an* epistemic supported model*) of P, if $\mathcal{A}$ is the set of models (respectively, stable models or supported models) of $\mathcal{M}(P^{\mathcal{A}})$.*

From the comments we made above, it follows that if T is an epistemic program, this more general definition yields the came notions of epistemic models of the three types as the earlier one.

We note that even in the more general setting the complexity of reasoning with epistemic (stable, supported) models remains unchanged. Specifically, we have the following result.

Theorem 9. *The problem to decide whether an epistemic theory $T \subseteq \mathcal{L}_K$ has an epistemic stable model is Σ_3^P-complete. The problem to decide whether an epistemic theory $T \subseteq \mathcal{L}_K$ has an epistemic model (epistemic supported model, respectively) is Σ_2^P-complete.*

Proof(Sketch): The hardness part follows from our earlier results concerning epistemic programs. To prove membership, we modify Proposition 1, and show a polynomial time algorithm with a Σ_2^P oracle (NP oracle for the last two problems) that decides, given a propositional theory S and a modal formula $K\varphi$ (with $\varphi \in \mathcal{L}_K$ and not necessarily in $\mathcal{L}$) whether $\mathcal{ST}(S) \models K\varphi$ (respectively, $\mathcal{M}(S) \models K\varphi$, or $\mathcal{SP}(S) \models K\varphi$). □

8 Discussion

In this paper, we proposed a two-valued formalism of epistemic theories — subsets of the language of modal propositional logic. We proposed a uniform way, in which semantics of propositional theories (the classical one as well as nonmonotonic ones: stable and supported) can be extended to the case of epistemic theories. We showed that the semantics of epistemic supported models is closely related to the original semantics of epistemic specifications proposed by Gelfond. Specifically we showed that the original formalism of Gelfond can be expressed in a straightforward way by means of epistemic programs in our sense under the semantics of epistemic supported models. Essentially all that is needed is to use fresh symbols x' to represent strong negation $\neg x$, and use the negation operator of our formalism, $\varphi \rightarrow \bot$ or, in the shorthand, $\neg\varphi$, to model the default negation $not\ \varphi$.

We considered in more detail the three semantics mentioned above. However, other semantics may also yield interesting epistemic counterparts. In particular, it is clear that Definition 6 can be used also with the minimal model semantics or with the Faber-Leone-Pfeifer semantics [14]. Each semantics gives rise to an interesting epistemic formalism that warrants further studies.

In logic programming, eliminating strong negation does not result in any loss of the expressive power but, at least for the semantics of stable models, disjunctions cannot be compiled away in any concise way (unless the polynomial hierarchy collapses). In the setting of epistemic programs, the situation is similar. The strong negation can be compiled away. But the availability of disjunctions in the heads and the availability of epistemic premises in the bodies of rules are essential. Each of these factors separately brings the complexity one level up. Moreover, when used together under the semantics of epistemic stable models they bring the complexity two levels up. This points to the intrinsic importance of having in a knowledge representation language means to represent indefiniteness in terms of disjunctions, and what is known to a program (theory) — in terms of a modal operator K.

Acknowledgments

This work was partially supported by the NSF grant IIS-0913459.

References

1. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9** (1991) 365–385
2. Gelfond, M., Lifschitz, V.: The stable semantics for logic programs. In: Proceedings of the 5th International Conference on Logic Programming (ICLP 1988), MIT Press (1988) 1070–1080
3. Gelfond, M.: Strong introspection. In: Proceedings of AAAI 1991. (1991) 386–391
4. Faber, W., Woltran, S.: Manifold answer-set programs for meta-reasoning. In Erdem, E., Lin, F., Schaub, T., eds.: Logic Programming and Nonmonotonic Reasoning, 10th International Conference, LPNMR 2009. Volume 5753 of Lecture Notes in Computer Science., Springer (2009) 115–128

5. Faber, W., Woltran, S.: Manifold answer-set programs and their applications. In this volume, Springer (2011)
6. Marek, W., Truszczyński, M.: Autoepistemic logic. Journal of the ACM **38** (1991) 588–619
7. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: propositional case. Annals of Mathematics and Artificial Intelligence **15** (1995) 289–323
8. Apt, K., Blair, H., Walker, A.: Towards a theory of declarative knowledge. In Minker, J., ed.: Foundations of deductive databases and logic programming, Morgan Kaufmann (1988) 89–142
9. Baral, C., Gelfond, M.: Logic programming and knowledge representation. Journal of Logic Programming **19/20** (1994) 73–148
10. Brass, S., Dix, J.: Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. Journal of Logic Programming **32(3)** (1997) 207–228
11. Inoue, K., Sakama, C.: Negation as failure in the head. Journal of Logic Programming **35** (1998) 39–78
12. Ferraris, P.: Answer sets for propositional theories. In: Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LPNMR 2005. Volume 3662 of LNAI., Springer (2005) 119–131
13. Truszczynski, M.: Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. Artificial Intelligence (2010) In press, available through Science Direct at http://dx.doi.org/10.1016/j.artint.2010.08.004.
14. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: semantics and complexity. In: Proceedings of the 9th European Conference on Artificial Intelligence (JELIA 2004). Volume 3229 of LNAI., Springer (2004) 200 – 212