Fast Prototyping of a Solver for Reduct-based ELP Semantics*

Stefania Costantini[1,**], Andrea Formisano[2]

[1]DISIM - Università dell'Aquila, via Vetoio, 67100, L'Aquila, Italy
[2]DMIF - Università di Udine, via delle Scienze 206, 33100 Udine, Italy

Abstract

Several semantic approaches have been proposed over time for Epistemic Logic Programs (ELPs), which is an extension to Answer Set Programming (ASP) with epistemic operators. ELP semantics has been defined, in various ways, in terms of *world views*, which are sets whose elements are sets of atoms. Several semantic approaches are *Reduct-based*, i.e., extend to ELPs what done for ASP, in the sense that in order to find the world views of a given program they propose to: start with a candidate world view; build the *reduct* of the program with respect to this candidate world view, according to some specific definition of such reduct; compute the set of stable models of the reduct; check whether the candidate world view is indeed a world view, which is the case if it coincides with the set of stable models of the reduct. Solvers have been developed for some of these approaches, but new semantics/variations have been introduced, and are likely to be introduced in the future, as there is no consensus yet on the "right" semantics. We propose a fast-prototyping approach to obtain a solver for any reduct-based semantics, with the advantage to be able to experiment the approach on small/medium programs, and not only on very small programs as done so far, prior to undertaking the costly process of developing a dedicated solver.

Keywords

Answer Set Programming, Epistemic Logic Programs, ELP semantics

1. Introduction

The first definition of Epistemic Logic Programs (ELPs, in the following just 'programs' if not explicitly stated differently) was presented many years ago in [1, 2]. ELPs were meant to extend in a straightforward way Answer Set Programs (ASP programs), defined under the Answer Set Semantics [3], with *epistemic operators* so as to allow for forms of epistemic reasoning in a practical though principled way, as an extension of ASP solvers to ELPs was envisioned since then. ELPs' semantics resulted however more tricky than expected, because the new epistemic operators are able to introspectively "look inside" a program's own semantics, defined in terms of its "answer sets". In fact $\mathbf{K}A$, $\mathbf{K}$ standing for *knowledge*, is true if the (ground) atom A is true in every answer set of the very program Π where $\mathbf{K}A$ occurs. The derived operator of *possibility* $\mathbf{M}A$ means that A is true in some of the answer sets of Π. The *epistemic negation operator* **not** A

CILC'23: 38th Italian Conference on Computational Logic, June 21–23, 2023, Udine, Italy
* Research partially supported by Action COST CA17124 "DigForASP", by Interdepartmental Project on AI (Strategic Plan UniUD–22-25) and by INdAM-GNCS projects CUP E55F22000270001 and CUP E53C22001930001.
** Corresponding author.
stefania.costantini@univaq.it (S. Costantini); andrea.formisano@uniud.it (A. Formisano)
0000-0002-5686-6124 (S. Costantini); 0000-0002-6755-9314 (A. Formisano)

expresses that A *is not provably true*, meaning that A is false in at least one answer set of Π, i.e., $not\ \mathbf{K} not\ A$ holds, where not is standard ASP default negation.

Semantics of ELPs is provided in terms of some kind of mechanism to characterize *world views*, which are sets of answer sets (instead of a unique set of answer sets like in Answer Set Programming), where each world view consistently satisfies (according to a given semantic approach) the epistemic expressions that appear in a given program. Several semantic approaches for ELPs have been introduced beyond the seminal ones, among which [4, 5, 6, 7, 8, 9, 10].

As seen below, many of these approaches are *Reduct-based*, i.e., they extend to ELPs what done for ASP, in the sense that to find world views of a given program they propose to: start with a candidate world view, which is a set whose elements are sets of atoms; build the *reduct* of the program with respect to this candidate world view, according to some specific definition of such reduct; compute the set of stable models of the reduct; check whether the candidate world view is indeed a world view, which is the case if it coincides with the set of stable models of the reduct. Solvers have been developed for some of these approaches, but new semantics/variations have been introduced, and are likely to be introduced in the future, as there is no consensus yet on the "right" semantics.

We propose a fast-prototyping approach to obtain a solver for any reduct-based semantics, with the advantage to be able to experiment the approach on small/medium programs, and not only on very small programs as done so far. We have devised the method for our new semantics that we present below, but it turned out to be of general applicability for a class of reduct-based semantics. So, the demanding process of implementing a performant dedicated solver can be undertaken only when and if really deemed worthwhile.

The paper is organized as follows. In Section 2 we recall Answer Set Programming. In Section 3 we briefly recall ELPs and their (envisaged) semantic properties. In Section 4 we report the formal definition of the main existing semantics for ELPs, including our own. In Section 5 we introduce and discuss our proposal to the construction of a solver given a reduct-based semantics with no regard to efficiency, but rather to fast prototyping. Finally, in Section 6 we conclude. In the Appendix, for the sake of completeness, in we report tables with the results that well-established semantics return on some tiny programs, and a list of available ELP solvers.

2. Answer Set Programming and Answer Set Semantics

In ASP one can see a program as a set of statements that specify a problem, where each answer set represents a solution compatible with this specification. Whenever an ASP program has no answer sets, it is said to be *inconsistent*, otherwise it is said to be *consistent*. Syntactically, an ASP program Π is a collection of *rules* of the form

$$A_1 \vee \ldots \vee A_g \leftarrow\ L_1, \ldots, L_n.$$

where each A_i, $0 \leq i \leq g$, is an atom, $\vee$ indicates disjunction and the L_is, $0 \leq i \leq n$, are literals (i.e., atoms or negated atoms of the form $not\ A$). The left-hand side and the right-hand side of the rule are called *head* and *body*, resp. A rule with empty body is called a *fact*. Disjunction can occur in rule heads only, so, in facts. A rule with empty head (or, equivalently, with head $\bot$), of the form '$\leftarrow L_1, ..., L_n$.' or '$\bot \leftarrow L_1, ..., L_n$.', is a *constraint*, stating that $L_1, \ldots, L_n$ are not allowed to be simultaneously true in an answer set. The impossibility to fulfill such requirement

is one of the reasons that make a program inconsistent. All extensions of ASP not explicitly mentioned above are not considered in this paper. We implicitly refer to the "ground" version of Π, which is obtained by replacing in all possible ways the variables occurring in Π with constants occurring in Π, and is thus composed of ground atoms, i.e., atoms which contain no variables.

The answer set (or "stable model") semantics, first introduced in [3, 11], can be defined in several ways [12, 13]. However, answer sets of a program Π, if any exists, are the supported minimal classical models of the program interpreted as a first-order theory in the obvious way. The original definition from [3], introduced for programs where rule heads were limited to be single atoms, was in terms of the 'GL-Operator'. Given set of atoms I and program Π, $GL_\Pi(I)$ is defined as the least Herbrand model of the program Π^I, namely, the (so-called) Gelfond-Lifschitz reduct of Π w.r.t. I. Π^I is obtained from Π by: 1. removing all rules which contain a negative literal $not\, A$ for $A \in I$; and 2. removing all negative literals from the remaining rules. The fact that Π^I is a positive program ensures that a least Herbrand model exists and can be computed via the standard immediate consequence operator [14]. Then, I is an answer set whenever $GL_\Pi(I) = I$.

3. Epistemic Logic Programs

Epistemic Logic Programs allow one to express within ASP programs so-called *subjective literals* (in addition to *objective literals*, that are those that can occur in plain ASP programs, plus the truth constants $\top$ and $\bot$). Such new literals are constructed via the *epistemic operator* $\mathbf{K}$ (disregarding, without loss of generality, the other epistemic operators). A positive subjective literal can be of the form $\mathbf{K}A$ or $\mathbf{K}not\, A$ where not is usual ASP default negation, and their negative counterpart can be of the form $not\, \mathbf{K}A$ or $not\, \mathbf{K}not\, A$. For the sake of simplicity, we do not consider double negation either inside or outside a subjective literal. Nesting of epistemic operators is not considered here. The literal $\mathbf{K}L$ means that the (ground) literal L is true in every answer set of a given program Π (i.e., L is a *cautious consequence* of Π).

The syntax of rules of ELP programs is analogous to ASP, save that literals in the body can now be either objective or subjective. An ELP program is called *objective* if no subjective literals occur therein, i.e., it is an ASP program. A constraint involving (also) subjective literals is called a *subjective constraint*, where one involving objective literals only is an *objective constraint*. Let At be the set of atoms occurring (within either objective or subjective literals) in a given program Π, and $Atoms(r)$ be the set of atoms occurring in rule r. Let $Head(r)$ be the head of rule r and $Body_{obj}(r)$ (resp., $Body_{subj}(r)$) be the (possibly empty) set of objective (resp., subjective) literals occurring in the body of r. We often write $Head(r)$ and $Body_{obj}(r)$ in place of $Atoms(Head(r))$ and $Atoms(Body_{obj}(r))$, respectively, when the intended meaning is clear from the context. We call *subjective rules* those rules whose body is made of subjective literals only.

The various semantics proposals for ELPs are based on the notion of *world views*: namely, sets of sets of atoms. A world view provides possible truth values for all literals in a program. Different proposals characterize in a different way how to define world views and how to find them. There are cases where different semantics however agree.

For instance, the program

$$\{a \leftarrow not\, b,\ b \leftarrow not\, a,\ e \leftarrow not\, \mathbf{K}f,\ f \leftarrow not\, \mathbf{K}e\}$$

has, under every semantics, two world views: $[\{a, e\}, \{b, e\}]$, where $\mathbf{K}e$ is true and $\mathbf{K}f$ is false, and $[\{a, f\}, \{b, f\}]$ where $\mathbf{K}f$ is true and $\mathbf{K}e$ is false. Note that, according to a widely-used convention, each world view, which is a set of sets, is denoted by using brackets $[\,]$. The presence of two answer sets in each world view of the above program is due to the cycle on objective atoms, whereas the presence of two world views is due to the cycle on subjective atoms (in general, the existence and number of world views is related to such cycles, cf., [15] for a detailed discussion).

Let a semantics $\mathcal{S}$ be a function mapping each program into sets of 'belief views', i.e., sets of sets of atoms, where $\mathcal{S}$ has the property that, if Π is an objective program, then the unique member of $\mathcal{S}(\Pi)$ is the set of stable models of Π. Given a program Π, each member of $\mathcal{S}(\Pi)$ is called an *$\mathcal{S}$-world view* of Π (we will often write "world view" in place of "$\mathcal{S}$-world view" whenever mentioning the specific semantics is irrelevant). The main existing semantic approaches for ELPs are introduced in Section 4.

As usual, for any world view W and any subjective literal $\mathbf{K}L$, we write $W \models \mathbf{K}L$ iff for all $I \in W$ the literal L is satisfied by I (i.e., if $L \in I$ for L atom, or $A \notin I$ if L is *not* A). W satisfies a rule r if each $I \in W$ satisfies r.

An interesting attempt to establish useful properties that ELP's semantics should obey can be found in [16, 17, 18]. The authors state that properties analogous to the ones which have been defined over time for ASP might prove useful for ELPs as well. In ASP, in fact, the answer sets of a given program can be computed incrementally, starting from the answer sets of the so-called bottom part of the program, which are used to simplify the so-called top part, where the process can be iterated by further splitting the top and/or the bottom [19]. Hence, authors of [16, 17, 18] extend to ELPs this idea by proposing a notion of *Epistemic Splitting*, where top and bottom are defined w.r.t. the occurrence of epistemic operators. They also consider *Subjective Constraint Monotonicity* and *Foundedness*. The property of *Subjective Constraint Monotonicity* states that, for any ELP program Π and any subjective constraint r, W is a world view of $\Pi \cup \{r\}$ iff both W is a world view of Π and W satisfies r. Thus, if this property is fulfilled by a semantic $\mathcal{S}$, a constraint can rule out world views but it cannot rule out some answer set from within a world view. Notice that Epistemic Splitting implies Subjective Constraint Monotonicity. *Foundedness* implies that atoms occurring in sets within a world view cannot have been derived through cyclic positive dependencies, where, to define such dependencies, $\mathbf{K}A$ is seen as the same as A. Finally, they define the class of *Epistemically Stratified Programs* that admit a unique world view (these programs are those where, intuitively, the use of epistemic operators is stratified).

4. Proposals for ELP Semantics

We report below some of the most relevant semantic definitions for ELPs. To compare the behavior of the various semantics on practical tiny examples, the reader may refer to Tables 1 and 2 in the Appendix. At present, there is no consensus about which is the 'right' semantic approach, and which results a semantics should return on controversial examples (e.g., on those reported in Table 2). A debate is quite alive about what should be the 'intuitive' outcome on these examples. This constitutes a strong motivation for our approach, i.e., providing a way of quickly and easily constructing a solver so as to be able to make experiments with new semantic approaches or with variations of existing ones.

We start with the seminal definition of the first ELP semantics, introduced in [2], that we call for short G94. Let Π be an ELP program, and r a rule occurring therein.

Definition 4.1 (G94-world views). *The G94-reduct of* Π *with respect to a non-empty set of interpretations* W *is obtained by: (i) replacing by* $\top$ *every subjective literal* $L \in Body_{subj}(r)$ *such that* L *is of the form* $\mathbf{K}G$ *and* $W \models G$*, and (ii) replacing all other occurrences of subjective literals of the form* $\mathbf{K}G$ *by* $\bot$*. A non-empty set of interpretations* W *is a G94-world view of* Π *iff* W *coincides with the set of all stable models of the G94-reduct of* Π *with respect to* W*.*

This definition was then extended to a new one in [4], that we call for short G11:

Definition 4.2 (G11-world views). *The G11-reduct of* Π *w.r.t. a non-empty set of interpretations* W *is obtained by: (i) replacing by* $\bot$ *every subjective literal* $L \in Body_{subj}(r)$ *such that* $W \not\models L$*, (ii) removing all other occurrences of subjective literals of the form* $not\ \mathbf{K}L$*. (iii) replacing all other occurrences of subjective literals of the form* $\mathbf{K}L$ *by* L*. The set* W *is a G11-world view of* Π *iff* W *coincides with the set of all stable models of the G11-reduct of* Π *w.r.t.* W*.*

In [16], it is noticed that the semantics K15 introduced in [20] and recalled by the following definition, slightly generalizes the semantics proposed in [4]:

Definition 4.3 (K15-world views). *The K15-reduct of* Π *w.r.t. a non-empty set of interpretations* W *is obtained by: (i) replacing by* $\bot$ *every subjective literal* $L \in Body_{subj}(r)$ *such that* $W \not\models L$*, and (ii) replacing all other occurrences of subjective literals of the form* $\mathbf{K}L$ *by* L*. The set* W *is a K15-world view of* Π *iff* W *coincides the set of all stable models of the K15-reduct of* Π *w.r.t.* W*.*

Semantics G11 and K15, that are refinements of the original G94 semantics, have been proposed over time to cope with new examples that were discovered, on which existing semantic approaches produce unwanted or not intuitive world views. K15 can be seen as a basis for the semantics proposed in [7] (called S16 for short). In particular, S16 treats K15 world views as candidate solutions, to be pruned in a second step, where some world views are removed, by applying the principle of keeping those which maximize what is not known. World views in [7] are obtained in particular as follows, where note however that the authors of [7] consider the operator **not**, that can be rephrased as $not\ \mathbf{K}A$ where not is ASP standard 'default negation' (meaning that A must be false in some answer set of a given world view).

Let $EP(\Pi)$ be the set of literals of the form **not** F occurring in given program Π.

Definition 4.4 (S16-world views). *Given* $\Phi \subseteq EP(\Pi)$*, the* Epistemic reduct Π^{Φ} *of* Π *w.r.t.* Φ *is obtained by: (i) replacing every* ***not*** $F \in \Phi$ *with* $\top$*, and (ii) replacing every* ***not*** $F \notin \Phi$ *with* $not\ F$*. Then, the set* $\mathcal{A}$ *of the answer sets of* Π^{Φ} *is a* candidate world view *if every* ***not*** $F \in \Phi$ *is true w.r.t.* $\mathcal{A}$ *(i.e.,* F *is false in some answer set* $J \in \mathcal{A}$*) and every* ***not*** $F \notin \Phi$ *is false (i.e.,* F *is true in every answer set* $J \in \mathcal{A}$*).*

We say that $\mathcal{A}$ *is obtained from* Φ *(or it is corresponding to* Φ*, or that it is a candidate world view w.r.t.* Φ*), where* Φ *is called a* candidate valid guess*. Then,* $\mathcal{A}$ *is an S16* world view *if it is maximal, i.e., if there exists no other candidate world view obtained from guess* Φ' *where* $\Phi \subset \Phi'$ *(so,* Φ *is called a* valid guess*).*

All the above semantics, in order to check whether a belief view $\mathcal{A}$ is indeed a world view, do the following: define some kind of reduct, reminiscent of the one introduced to define to the stable model semantics, and $\mathcal{A}$ is a world view if it is *stable* w.r.t. this reduct, typically if it coincides with the stable models of the reduct. Variations are possible, like, e.g., the minimality criterium adopted by S16. Therefore, we say that all previously introduced approaches, as well as any other similar one, is *reduct-based*. This in contrast with other approaches that have been proposed, where no notion of reduct is involved.

The F15 semantics [6, 21] for instance, is based on very different principles, namely, it is based on a combination of Equilibrium Logic [22, 23] with the modal logic S5. There, an EHT interpretation associates, via a function h, a belief view $\mathcal{A}$ with another belief view $\mathcal{A}'$ composed, for every set $A \in \mathcal{A}$, of sets $A' \subseteq A$. The purpose is to state that an implication is entailed, in any "belief point", i.e., in any interpretation $A \in \mathcal{A}$, by the couple $\langle \mathcal{A}, \mathcal{A}' \rangle$ if it is entailed either by $\mathcal{A}$ or by $\mathcal{A}'$. An EHT interpretation satisfies a theory in the usual way, and is total on a subset $\mathcal{X}$ of $\mathcal{A}$ if h gives back sets in $\mathcal{X}$ unchanged. A *total EHT model* can be an *equilibrium EHT model*, and is defined to be an **F15 world view**, if it is minimal according to two particular minimality conditions (not reported here).

Differently from F15, FAAEL [18] is based on the modal logic KD45. To define FAAEL, a belief view is transformed from a set of interpretations to a set of HT-interpretations, i.e., interpretations in terms of the logic of Here-and-There (HT) [24] which are couples $\langle H, T \rangle$ of 'plain' interpretations. A belief view is *total* if $H = T$ for all composing interpretations, thus reducing to the previous notion of belief view. A total version of any belief view can be formed, taking all the T's as components. A belief interpretation is now a belief view plus an HT interpretation, say $\hat{H}$, possibly not belonging to the belief view. The peculiarity of the entailment relation (defined in terms of HT logic) is in the implication, that must hold (in the usual way) in the belief interpretation, but also in the total version of the belief view therein. For total belief interpretations, the new relation collapses to the modal logic KD45. An epistemic interpretation (i.e., a non-empty set of interpretations) is defined to be a belief model if all its composing HT interpretation as well as $\hat{H}$ entail all formulas of a given theory. It is an epistemic model, if $\hat{H}$ is among the composing interpretations, and it is an *equilibrium belief model* if it satisfies certain minimality conditions. A belief view is a **FAAEL world view** if it is "extracted" from an equilibrium belief model $\mathcal{E}$ by taking all the T components of each $\langle H, T \rangle$ which is found in $\mathcal{E}$. For formal definitions of F15 and FAAEL, that for lack of space we cannot report here, we refer the reader to the aforementioned references. Also, we apologize with the readers and with the authors, because for lack of space, we do not consider other recent semantics, such as [9, 25].

FAAEL satisfies [17] Epistemic Splitting, Subjective Constraint Monotonicity, and Foundedness. G94 satisfies Epistemic Splitting, Subjective Constraint Monotonicity, but not Foundedness. In [18], it is proved that FAAEL world views coincide with *founded* G94 world views, where (roughly) founded world views are those where in every composing interpretation, objective atom G is never derived, directly or indirectly, from $\mathbf{K}G$; c.f. [26] for a discussion on foundedness, very partially reported below. All the other above-mentioned semantic approaches satisfy none of these properties.

Respecting all the three properties implies that ELP programs behave very much like ASP programs; in this way, on the one hand many theoretical results developed for ASP can be extended to ELP, but on the other hand the introspective capabilities of epistemic reasoning that

inspired their introduction are, in our opinion, neglected. Basically in fact, the difference that remains w.r.t. ASP and ELP under. e.g., FAAEL, is that a negative cycle on epistemic literals like the one in the already-seen example leads to two world views, or, better, two for each such cycle; this analogously to how cycles on default negation in ASP that lead to two answer sets. When the two kinds of cycles can co-exist, a world view is composed of several answer sets. This phenomenon is widely discussed in [15], where the number of resulting world views for a given program is also evaluated.

Thus, a discussion about which properties are "really" required to be necessarily respected is in order. Respecting one or more of them could even be a design choice: according to a specific application context of ELPs, one might choose a semantics rather than another one from a 'toolkit' of existing ones.

Foundedness appears to be a reasonable property to fulfill. The problem of unfounded world views was discovered a long time ago with G94. In fact, in the example:

$a \leftarrow \mathbf{K}a.$

one has that $[\{a\}]$ is a G94 world view, in that it entails $\mathbf{K}A$, thus allowing A to be derived. This is due to the fact that, in many approaches, what is known is dictated by the world view, and there is no way to impose that it should be consistently derived from the program as well. Reduct-based semantic approaches subsequent to G94, although devised to try to overcome this problem, still do not satisfy foundedness.

Subjective Constraint Monotonicity is more controversial. According to the definition of this property, adding a constraint might rule out world views, but could not possibly modify existing ones. To see why this point of view may not be fully convincing, consider the following answer set program, which is after all a particular case of an epistemic logic program, where no epistemic literals occur:

$a \vee b.$

$\leftarrow not\, a.$

If it consisted of the first rule only (that constitutes, notice, the program without the constraint), its unique world view would coincide with the set of its answer sets, i.e., $[\{a\}, \{b\}]$. Adding the constraint to the program rules out the answer set $\{b\}$, so to every extent it modifies the world view of the program to be $[\{a\}]$. Thus, apparently, the ASP semantics does not fulfill Subjective Constraint Monotonicity, according to which the constraint should "validate" the world view if fulfilled, or cancel it if violated.

Consider now the very similar epistemic logic program:

$a \vee b.$

$\leftarrow not\, \mathbf{K}a.$

Under many semantics what happens is absolutely analogous (as in our opinion should be), leading to the unique world view $[\{a\}, \{b\}]$ of the rule being modified into $[\{a\}]$ to fulfill the constraint. This is in fact the result returned by K15 and S16, and also by CF22F, seen below. In these semantics therefore, the property of Subjective Constraint Monotonicity is not satisfied.

The CF22F reduct-based semantics has been proposed in a preliminary form and subsequently refined by the authors of the present paper [27, 26], and is reported below for the sake of completeness. In this approach, we consider subjective literals $\mathbf{K}G$ and $\mathbf{K}notG$ as new atoms, called *knowledge atoms*. Negation not in front of knowledge atoms is assumed to be the standard default negation. So, instead of ELPs proper, we here equivalently consider ASP programs (called

below just 'programs') possibly involving knowledge atoms. Let $SM(\Pi)$ be the set of answer sets of such a program Π.

First of all we introduce the concept of internal consistency of a set of atoms.

Definition 4.5. *A set A of atoms, composed of objective atoms and knowledge atoms, is said to be* knowledge-consistent *iff the following conditions both hold:*

(i) if $\mathbf{K}G \in A$ then $G \in A$;
(ii) if $\mathbf{K}notG \in A$ then $G \notin A$.

Notice that a set of objective atoms is knowledge-consistent: as no knowledge atom appears therein, both conditions stated in the definition are trivially satisfied.

Definition 4.6. *Given ASP program Π possibly involving knowledge atoms, let $SMC(\Pi)$ be the set of those answer sets of the program which are knowledge-consistent.*

Property 4.1. *The set $SMC(\Pi)$ coincides with the set of the stable models of the program Π' obtained from Π by adding, for each knowledge atom $\mathbf{K}G$ or $\mathbf{K}notG$ occurring in Π, constraints:*

$$\leftarrow \mathbf{K}G, not\, G$$
$$\leftarrow \mathbf{K}notG, G.$$

An *epistemic interpretation* $\mathcal{W}$ is in our case a possibly empty set of sets of atoms, each such set composed of objective atoms occurring in the head of some (possibly unit) rule in Π. We make a difference with previous approaches where we allow $\mathcal{W}$ to be empty. We make a slight improvement with the subsequent condition, that holds anyway for world views obtained via any semantics.

Definition 4.7. *[CF22F-adaptation] The CF22F-adaptation Π-$\mathcal{W}$ of a program Π with respect to an epistemic interpretation $\mathcal{W}$ is a new program, now obtained by modifying Π as follows:*

(i) whenever $\mathcal{W} \models G$, in all rules with head G substitute head G with $\mathbf{K}G$, and add new rule $G \leftarrow \mathbf{K}G$ and
(ii) whenever $\mathcal{W} \models not\, G$, add new fact $\mathbf{K}notG$.

Let $F_{\Pi\text{-}\mathcal{W}}$ be the set of the newly added rule heads of the form $\mathbf{K}G$.

The CF22F adaptation is none other than our definition of a reduct, though extended with respect to previously-known ones.

Definition 4.8. *Given an epistemic interpretation $\mathcal{W}$ for program Π, we call $SM'(\Pi)$ the set of sets obtained from $SMC(\Pi)$ by canceling knowledge atoms.*

Definition 4.9 (CF22F world view). *An epistemic interpretation $\mathcal{W}$ is called a CF22F world view of a program Π if $\mathcal{W} = SM'(\Pi\text{-}\mathcal{W})$.*

To provide a demonstration of the new semantics, consider the epistemic logic program:

$a \leftarrow not\ \mathbf{K} not\ a.$

Consider the epistemic interpretation $\mathcal{W} = [\emptyset]$. According to Definition 4.7, to obtain Π-$\mathcal{W}$ the only added fact is: $\mathbf{K} not\ a$. We have that $SMC(\Pi\text{-}\mathcal{W}) = [\{\mathbf{K} not\ a\}]$, thus $SM'(\Pi\text{-}\mathcal{W}) = [\emptyset]$, so $\mathcal{W}$ **is** a CF22F world view.

Consider the epistemic interpretation $\mathcal{W} = [\{a\}]$. According to Definition 4.7, the program becomes:

$\mathbf{K}a \leftarrow not\ \mathbf{K} not\ a.$
$a \leftarrow \mathbf{K}a.$

We have that $SM(\Pi\text{-}\mathcal{W}) = SMC(\Pi\text{-}\mathcal{W}) = [\{\mathbf{K}a, a\}]$ (as fact $\mathbf{K} not\ a$ is not present, its negation is true), thus $SM'(\Pi\text{-}\mathcal{W}) = [\{a\}]$, so $\mathcal{W}$ **is** a CF22F world view.

On this example, we notice an agreement among CF22F and G94, G11, FAAEL. Notice that the minimization criterion does not apply here, as the sets of facts and/or rules added according to Definition 4.7 for the two possible epistemic interpretations are not one a subset of the other.

It is easy to prove (proof not reported here) that CF22F enjoys foundedness, yet not Subjective Constraint Monotonicity. Let us consider again program:

$a \leftarrow \mathbf{K}\, a.$

the CF22F-adaptation transforms the program into:

$a \leftarrow \mathbf{K}\, a.$
$\mathbf{K}\, a \leftarrow \mathbf{K}\, a.$

This program has no answer sets, thus $[\{a\}]$ **is not** a CF22F world view.

We may notice that, the S16 semantics is remarkable in the sense that it maximizes what is not known, which is equivalent to minimizing what is known. The proposers of S16 consider each potential world view (that in their approach is associated to a guess about what is not known) as a candidate world view, and discard those for which there exists another one with a larger guess on what is not known (equivalently, a smaller guess on what is known), in terms of set inclusion. Rephrasing their criterion (referred to as S16C) in terms of our approach, we have:

Definition 4.10 (S16 Criterion - CF22F+S16C). *Each world view $\mathcal{W}$ as defined in Definition 4.9 is considered to be a* candidate world view. *A candidate world view $\mathcal{W}$ is indeed a world view under CF22F+S16C if no other candidate world view $\mathcal{W}$' exists, where $F_{\Pi\text{-}\mathcal{W}'} \subset F_{\Pi\text{-}\mathcal{W}}$.*

5. Fast Prototyping of a Solver for Reduct-based Semantic Approaches

As reported in [28], evaluating ELPs is a computationally hard task. The central decision problem, checking whether an ELP has a world view, is Σ_P^3-complete [7, 29], and problems even higher on the polynomial hierarchy exist. As reported in Table 3 in the Appendix, solvers for various semantic approaches have been developed (cf. [30, 31, 20, 32] for discussion and useful references). In order to perform experiments on CF22F, we devised and illustrate below a method which allows one to obtain very quickly a solver for any reduct-based semantics, without implementation effort. Thus, researchers proposing new semantic approaches (as it often happens)

or extending/modifying an existing one can exploit the method in order to make experiments. It must be noticed however that the method provides correctness of the solver w.r.t. a given semantics, but does not cope with efficiency issues. Thus, given the high complexity of evaluating ELPs, the prototypical solver obtained via our method can be only used to experiment with program of small-medium size, which is however better than just considering the tiny ones shown in the Appendix.

Notice that, for checking whether an epistemic interpretation $\mathcal{W}$ is a world view for program Π according to a semantics $\mathcal{S}$ based upon a definition $\mathcal{R}$ of a reduct, one has to apply $\mathcal{R}$ to $\mathcal{W}$, then find the set $\mathcal{W}_{intermediate}$ of answer sets of the reduced program $\Pi_{\mathcal{R}}$, then possibly perform some post-processing $\mathcal{P}$, thus obtaining $\mathcal{W}_{final}$. $\mathcal{W}$ is a world view of Π according to $\mathcal{S}$ iff $\mathcal{W} = \mathcal{W}_{final}$. In the case of CF22F for instance, the post-processing cancels from $\mathcal{W}_{intermediate}$ those sets which are not knowledge consistent, and then cancels knowledge atoms.

In order to find all the world views of Π, one has to devise all the epistemic interpretations. Given the set At_{Π} of the atoms occurring in Π (after the grounding), one has to find all the subsets of such set, and all the sets of such subsets. The sets so found should then be filtered, to select those including atoms occurring as the head of some rule in Π. Clearly, this process as whole has a high complexity, basically it 'absorbs' most of the complexity of the entire method. Then, all the epistemic interpretations must be checked, and if applying the minimality criterion as defined in S16 and adopted in CF22F+S16C, the resulting 'candidate' world views must be filtered, and this, as discussed in [7], adds further complexity.

The method can be automated as illustrated below, resorting to a single call to an answer set solver (although on a large instance).

Definition 5.1. *Let At^i_{Π}, for an integer positive number i, be the* standardized apart i-th version *of At_{Π} obtained by substituting each atom A therein by a fresh atom A^i. The standardized apart i-th version Π^i of program Π is obtained by substituting each atom in Π with its standardized apart counterpart.*

Definition 5.2. *Given an ELP program Π and a semantic $\mathcal{S}$ with associated reduct definition $\mathcal{R}$ and (possibly) post-processing definition $\mathcal{P}$, let us assume to have implemented:*

1. *A module $M_{\mathcal{W}}$ that will return the epistemic interpretations $\mathcal{W}_1, \ldots, \mathcal{W}_k$ of Π.*
2. *A module M_{sa} that, given $\mathcal{W}_1, \ldots, \mathcal{W}_k$, will generate a new program Π_{sa} obtained as the union of the standardized apart i-th versions of Π, $i \leq k$.*
3. *A module M_{red} that applies the reduct $\mathcal{R}$ to Π_{sa}, thus obtaining Π_{sa_R}*
4. *A module M_{int} extracting $\mathcal{W}^i_{intermediate}$, $i \leq k$, by selecting from the answer sets of Π_{sa_R} (obtained via any answer set solver) all the answer sets composed of the i-th standardized apart atoms, collecting them into $\mathcal{W}^i_{intermediate}$, and finally de-standardizing-apart the atoms, i.e., substituting each atom A^i with A.*
5. *A module M_{Post} applying post-processing $\mathcal{P}$ to $\mathcal{W}^i_{intermediate}$, $i \leq k$, thus obtaining $\mathcal{W}^i_{final}$, $i \leq k$.*
6. *A module $M_{compare}$ that selects from the $\mathcal{W}^i_{final}$, $i \leq k$, those which are world views, as they coincide with $\mathcal{W}_i$, and returns the actual set of world views.*
7. *A module M_{filter} that, if a minimality criterion has to be applied, selects among the world views, here intended as 'candidate', those fulfilling the criterion.*

Definition 5.3. *The 'quick solver' for a semantic $\mathcal{S}$ for ELPs with associated reduct definition $\mathcal{R}$ and (possibly) post-processing definition $\mathcal{P}$, which returns all world views of given program* Π *according to $\mathcal{S}$, is obtained by running on* Π *the pipeline of the modules listed in Definition 5.2.*

Correctness of the 'quick solver' w.r.t. a semantic $\mathcal{S}$ depends upon the correct implementation of the various modules, that however should not be difficult to ensure. In fact, each module copes with a single aspect and will thus show enough transparency and a reasonable size.

6. Conclusions

We have revised semantic approach for ELPs, including our new approach called CF22F, not yet presented elsewhere. The experiments of CF22F on several examples taken from the relevant literature, for which the outcome of the other most relevant semantic approaches was well-known (experiments not reported here for lack of space), gave surprising results. In fact, CF22F does not agree uniformly with any of the other semantics, and in some cases it agrees with none of them. CF22F agrees most frequently with S16, often without even applying the S16C Criterion. More investigation and more experiments are required to understand the reasons. In order to make experiments, we devised a method for fast prototyping of solvers for reduct-based semantic approaches. The method can be used for CF22F, but also for any other extension/new proposal. The issue of efficiency is not coped with, yet the method allows one to experiment a semantic on much larger examples than those frequently used in the literature, and listed in Tables 1 and 2 in the Appendix. Thus, we believe that it can be useful to researchers in the area.

References

[1] M. Gelfond, H. Przymusinska, Definitions in epistemic specifications, in: A. Nerode, V. W. Marek, V. S. Subrahmanian (Eds.), Proc. of the 1st Intl. Workshop on Logic Programming and Non-monotonic Reasoning, The MIT Press, 1991, pp. 245–259.

[2] M. Gelfond, Logic programming and reasoning with incomplete information, Ann. Math. Artif. Intell. 12 (1994) 89–116. doi:10.1007/BF01530762.

[3] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, K. Bowen (Eds.), Proc. of the 5th Intl. Conf. and Symp. on Logic Programming, MIT Press, 1988, pp. 1070–1080.

[4] M. Gelfond, New semantics for epistemic specifications, in: J. P. Delgrande, W. Faber (Eds.), Proc. of LPNMR'11, volume 6645 of *LNCS*, Springer, 2011, pp. 260–265.

[5] M. Truszczynski, Revisiting epistemic specifications, in: M. Balduccini, T. C. Son (Eds.), Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning, volume 6565 of *LNCS*, Springer, 2011, pp. 315–333.

[6] L. Fariñas del Cerro, A. Herzig, E. I. Su, Epistemic equilibrium logic, in: Q. Yang, M. J. Wooldridge (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, AAAI Press, 2015, pp. 2964–2970.

[7] Y. Shen, T. Eiter, Evaluating epistemic negation in answer set programming, Artificial Intelligence 237 (2016) 115–135.

[8] P. T. Kahl, A. P. Leclerc, Epistemic logic programs with world view constraints, in: A. D. Palù, P. Tarau, N. Saeedloei, P. Fodor (Eds.), Tech. Comm. of ICLP 2018, volume 64 of *OASIcs*, Schloss Dagstuhl, 2018, pp. 1:1–1:17.
[9] E. I. Su, Epistemic answer set programming, in: F. Calimeri, N. Leone, M. Manna (Eds.), Proc. of JELIA'19, volume 11468 of *LNCS*, Springer, 2019, pp. 608–626.
[10] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, Splitting epistemic logic programs, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), Proc. of LPNMR'19, volume 11481 of *LNCS*, Springer, 2019, pp. 120–133.
[11] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Generation Computing 9 (1991) 365–385.
[12] V. Lifschitz, Thirteen definitions of a stable model, in: A. Blass, N. Dershowitz, W. Reisig (Eds.), Fields of Logic and Computation, volume 6300 of *LNCS*, Springer, 2010, pp. 488–503.
[13] S. Costantini, A. Formisano, Negation as a resource: a novel view on answer set semantics, Fundamenta Informaticae 140 (2015) 279–305.
[14] J. W. Lloyd, Foundations of Logic Programming, Springer-Verlag, 1987.
[15] S. Costantini, About epistemic negation and world views in epistemic logic programs, Theory Pract. Log. Program. 19 (2019) 790–807. doi:10.1017/S147106841900019X.
[16] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, Splitting epistemic logic programs, Theory Pract. Log. Program. 21 (2021) 296–316. doi:10.1017/S1471068420000058.
[17] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, On the splitting property for epistemic logic programs, in: C. Bessiere (Ed.), Proceedings of IJCAI 2020, ijcai.org, 2020, pp. 4721–4725.
[18] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, Autoepistemic answer set programming, Artif. Intell. 289 (2020) 103382.
[19] V. Lifschitz, H. Turner, Splitting a logic program, in: Proc. of ICLP'94, MIT Press, 1994, pp. 23–37.
[20] P. Kahl, R. Watson, E. Balai, M. Gelfond, Y. Zhang, The language of epistemic specifications (refined) including a prototype solver, J. Log. Comput. 30 (2015) 953–989.
[21] E. I. Su, L. Fariñas del Cerro, A. Herzig, Autoepistemic equilibrium logic and epistemic specifications, Artif. Intell. 282 (2020) 103249. doi:10.1016/j.artint.2020.103249.
[22] D. Pearce, A new logical characterization of stable models and answer sets, in: Non-Monotonic Extensions of Logic Programming, number 1216 in LNAI, Springer, 1997, pp. 55–70.
[23] D. Pearce, A. Valverde, Synonymous theories in answer set programming and equilibrium logic, Proc. of ECAI04, 16th Europ. Conf. on Art. Intell. (2004) 388–390.
[24] D. Pearce, Equilibrium logic, Ann. Math. Artif. Intell. 47 (2006) 3–41.
[25] E. I. Su, Refining the semantics of epistemic specifications, in: A. F. et al. (Ed.), Proceedings 37th International Conference on Logic Programming, ICLP Technical Communications 2021, volume 345 of *EPTCS*, 2021, pp. 113–126. doi:10.4204/EPTCS.345.25.
[26] S. Costantini, A. Formisano, Achieving foundedness in reduct-based epistemic logic programs semantics, 2023. Submitted, a preliminary version can be obtained from the authors.
[27] S. Costantini, A. Formisano, Epistemic logic programs: a novel perspective and some extensions, in: J. Arias, R. Calegari, L. Dickens, W. Faber, J. Fandinno, G. Gupta, M. Hecher, D. Inclezan, E. LeBlanc, M. Morak, E. Salazar, J. Zangari (Eds.), Proc. of the ICLP'22

Table 1
On the left, examples where G94, G11, K15, F15, S16, and FAEEL agree. On the right, examples where G94/G11/FAEEL differ from K15/F15/S16. (Taken from [18].)

Program	World views
$a \vee b$	$[\{a\}, \{b\}]$
$a \vee b$ $a \leftarrow \mathbf{K}b$	$[\{a\}, \{b\}]$
$a \vee b$ $a \leftarrow not\ \mathbf{K}b$	$[\{a\}]$
$a \vee b$ $c \leftarrow not\ \mathbf{K}b$	$[\{a, c\}, \{b, c\}]$
$a \leftarrow not\ \mathbf{K}b$ $b \leftarrow not\ \mathbf{K}a$	$[\{a\}],\ [\{b\}]$
$a \leftarrow not\ \mathbf{K}not\ a$ $a \leftarrow not\ \mathbf{K}a$	$[\{a\}]$

Program	G94/G11/FAEEL	K15/F15/S16
$a \leftarrow not\ \mathbf{K}not\ a$	$[\emptyset],\ [\{a\}]$	$[\{a\}]$
$a \vee b$ $a \leftarrow not\ \mathbf{K}not\ b$	none	$[\{a\}]$
$a \vee b$ $a \leftarrow \mathbf{K}not\ b$	$[\{a\}],\ [\{a\}, \{b\}]$	$[\{a\}, \{b\}]$
$a \leftarrow b$ $b \leftarrow not\ \mathbf{K}not\ a$	$[\emptyset],\ [\{a, b\}]$	$[\{a, b\}]$
$a \leftarrow not\ \mathbf{K}not\ b$ $b \leftarrow not\ \mathbf{K}not\ a$	$[\emptyset],\ [\{a\}, \{b\}]$	$[\{a\}, \{b\}]$

Workshops, volume 3193 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022.

[28] M. Hecher, M. Morak, S. Woltran, Structural decompositions of epistemic logic programs, in: Proc. of AAAI'20, IAAI'20, EAAI'20, AAAI Press, 2020, pp. 2830–2837.

[29] M. Morak, Epistemic logic programs: A different world view, in: B. Bogaerts, E. Erdem, P. Fodor, A. Formisano, G. Ianni, D. Inclezan, G. Vidal, A. Villanueva, M. D. Vos, F. Yang (Eds.), Proc. of ICLP'19 (Technical Communications), volume 306 of *EPTCS*, 2019, pp. 52–64. doi:10.4204/EPTCS.306.11.

[30] A. P. Leclerc, P. T. Kahl, A survey of advances in epistemic logic program solvers, CoRR abs/1809.07141 (2018). URL: http://arxiv.org/abs/1809.07141. arXiv:1809.07141.

[31] P. T. Kahl, A. P. Leclerc, T. C. Son, A parallel memory-efficient epistemic logic program solver: harder, better, faster, Ann. Math. Artif. Intell. 86 (2019) 61–85. doi:10.1007/s10472-019-09621-1.

[32] P. Cabalar, J. Fandinno, J. Garea, J. Romero, T. Schaub, eclingo : A solver for epistemic logic programs, Theory Pract. Log. Program. 20 (2020) 834–847.

A. Semantic Results for Interesting ELP Programs

In Tables 1 and 2 a summary is reported, taken from [18], of how the previously-existing semantics presented in this paper behave on some examples which are considered to be significant of situations that can be found in practical programming.

B. Available ELP Solvers

Table 3 shows, to the best of our knowledge, a list of available solvers for the semantics reported in previous sections.

Table 2
Examples showing differences among several semantics. (Taken from [18].)

Program	G94	G11/FAEEL	K15	F15/S16
$a \leftarrow not\ \mathbf{K} not\ b \wedge not\ b$ $b \leftarrow not\ \mathbf{K} not\ a \wedge not\ a$	$[\emptyset], [\{a\}, \{b\}]$			$[\{a\}, \{b\}]$
$a \leftarrow \mathbf{K} a$	$[\emptyset],\ [\{a\}]$	$[\emptyset]$		
$a \leftarrow \mathbf{K} a$ $a \leftarrow not\ \mathbf{K} a$	$[\{a\}]$	none		

Table 3
List of solvers available for the semantics summarized in Section 4

Solver	Year	Semantics	Underlying ASP-solver	Impl. language	Available form
ELMO	1994	G94	dlv	Prolog	n/a
sismodels	1994	G94	claspD	C++	n/a
Wviews	2007	G94	clingo	C++	Windows binary
Esmodels	2013	G11	clingo	(unknown)	Windows binary
ELPS	2014	K15	clingo	Java	source+binary
GISolver	2015	K15	clingo	(unknown)	Windows binary
ELPsolve	2016	K15/S16	clingo	C++	binary only
Wviews2	2017	G94	clingo	Python	Windows binary
EP-ASP	2017	K15/S16	clingo	Python+ASP	Windows binary
PelpSolver	2017	S16	clingo	Java	Windows binary
ELPsolve2	2017	S16	clingo	C++	currently not public release
EHEX	2018	S16	clingo	Python	source
selp	2018	S16	clingo	Python	source
eclingo	2020	G94	clingo	Python	source