*TPLP* **19** (5–6): 790–807, 2019. © Cambridge University Press 2019
doi:10.1017/S147106841900019X


# *About Epistemic Negation and World Views in Epistemic Logic Programs*

STEFANIA COSTANTINI
*DISIM - Università dell'Aquila, Italy*
(*e-mail:* stefania.costantini@univaq.it)

*submitted 25 July 2019; accepted 31 July 2019*

**Abstract**

In this paper we consider Epistemic Logic Programs, which extend Answer Set Programming (ASP) with "epistemic operators" and "epistemic negation", and a recent approach to the semantics of such programs in terms of *World Views*. We propose some observations on the existence and number of world views. We show how to exploit an extended ASP semantics in order to: (i) provide a characterization of world views, different from existing ones; (ii) query world views and query the whole set of world views.

*KEYWORDS*: Answer Set Programming, Epistemic Logic Programs, Epistemic Negation

## 1 Introduction

Answer Set Programming (ASP) is a successful logic programming paradigm under the answer set semantics (AS) (Gelfond and Lifschitz 1988; 1991). ASP has been widely applied, e.g., to information integration, constraint satisfaction, routing, planning, diagnosis, configuration, computer-aided verification, biology/biomedicine, knowledge management, etc. (cf. (Brewka et al. 2016) and the references therein). The ASP approach to problem-solving consists basically in the following: (i) encoding of the given problem via an ASP program; (ii) computing the "answer sets" of such a program via an inference engine, or "ASP solver"; (iii) extracting the problem solutions by examining such answer sets; in fact, answer set contents can be in general reformulated in order to present the solution in terms of the given problem.

Epistemic Logic programs (ELPs), first introduced in (Gelfond and Przymusinska 1991), extend ASP with *epistemic operators* **K** and **M**: **K***A* means that (ground) atom *A* is true in every answer set of given program Π, whereas **M***A* means that *A* is true in some of the answer sets of Π. Thus, such operators introduce a form of reasoning over multiple answer sets, where the collection of all the answer sets of Π is called a *world view* and, if **M** and **K** occur in Π, more than one world view may exist. The *epistemic negation operator* **not** expresses that *A is not provably true*, meaning that *A* is false in at least one answer set of Π. By means of **not** one can define both **K** and **M**: in fact, **K***A* and **M***A* can be rephrased as *not* **not** *A* and **not** *not A*, respectively, where *not* is ASP standard 'default negation'. In fact: concerning **K**, if it is not true that *A* is false in some answer set then *A* must be true in all of them; concerning **M**, if *not A*

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

is false in some answer set then $A$ must be true in that answer set. Consequently, many approaches to ELPs, e.g., (Shen and Eiter 2016), consider explicitly only the operator **not**. The semantics of ELPs is provided in terms of *World Views*: instead of a unique set of answer sets like in ASP, there is now a set of such sets. Each one, called "world view", consistently satisfies the epistemic expressions that appear in a given program. An epistemic logic program may have several world views. ELP solving systems have been defined and implemented (Son et al. 2017) on top of state-of-the-art ASP solvers, that are invoked (more than once) to generate and check potential world views.

In this paper we refer to the work of (Shen and Eiter 2016) concerning the "general epistemic semantics" of ELPs. Many other semantic approaches/characterizations have been introduced, among which (Gelfond 2011; Truszczynski; del Cerro et al. 2015; Su 2017; Kahl and Leclerc 2018; Su 2019; Cabalar et al. 2019a; Cabalar et al. 2019b), essentially with the aim to avoid unintended world views, but also to propose extensions of the basic paradigm. We consider the semantics introduced in (Shen and Eiter 2016) because: it takes properly into account previous relevant work, and copes in a satisfactory way with most of the controversial examples; moreover, a "plus" of this approach is that it provides a very general characterization of world views, which is applicable to every variant of the AS semantics. Concerning practical applicability, the work of (Bichler et al. 2018) introduces an effective method for the practical characterization and computation of world views under the semantics of (Shen and Eiter 2016).

The traditional example proposed in the literature to illustrate and motivate the use of **not** is a re-elaboration of a famous example by Bowen and Kowalski (Bowen and Kowalski 1983):

$$innocent(X) \leftarrow suspect(X), not\ demo(guilty(X)).$$

stating that one is innocent if (s)he cannot be *proved* to be guilty (clearly, $X$ will be instantiated to some constant defined elsewhere in the program, say *john*). This example was formulated to support the introduction of prolog metainterpreters to implement the *demo* predicate. However, it can be reasonable to assume the adoption of ASP to encode available knowledge about the underlying investigative case: in fact, this knowledge (collected by investigators) can often be incomplete, uncertain, etc. If using ASP however, a formulation such as

$$innocent(X) \leftarrow suspect(X), not\ guilty(X).$$

does not suffice, as there can be answer sets where the suspect is deemed innocent and others where (s)he is deemed guilty, thus not allowing a unique conclusion to be drawn. By exploiting epistemic negation the formulation becomes the following:

$$innocent(X) \leftarrow suspect(X), \mathbf{not}\ guilty(X).$$

In particular, it states that one is innocent if not provably guilty: i.e., (s)he innocent if there exists some answer set where (s)he is not guilty. In fact, the existence of such an answer set introduces a *reasonable doubt*. In Section 7 we provide and discuss a simple though plausible underlying description of the case at hand where, due to uncertain knowledge (that in practical investigations occurs very often) the resulting ELP program has more than one world view.

**Our Contributions:**

1. We propose some observations about programs with epistemic negation, and in particular on the existence and number of world views.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

2. We introduce *epistemic scenarios* that are easy to obtain, and among which are the *valid guesses*, i.e., the hypotheses about the truth value of epistemic literals that determine world views. We establish, for the first time in the literature, an upper bound on the number of valid guesses (and thus of world views). This is of theoretical interest, as no estimation of the number of world views had ever been provided before. Also, the notion of epistemic scenarios reduces (in many cases very significantly) the number of guesses to be checked for validity in order to be able to compute world views.
3. We define a new method to check validity of a guess, which is particularly convenient when the given program contains few epistemic literals.
4. We provide for the first time in the literature a device for top-down (prolog-style) query-answering concerning either a single world view, or also the whole set of world views, which does not require to compute the world views beforehand.

Querying world views may be useful in reference to the above example, as an investigator/lawyer/judge may wish to pose a query about *john* without computing a whole world view. Moreover, in case several world views exist, the involved parties may wish to query whether *john* is innocent/guilty in some/every world view. We in fact provide several operators to query world views of given ELP program. $\mathbf{not}^W A$ means that **not** $A$ holds in some world view of $\Pi$, $\mathbf{K}^W A$ means that $A$ is true in every world view (where, according to (Shen and Eiter 2016), $A$ is true in a world view if $A$ belongs to all the answer sets that compose the world view). $\mathbf{M}^W A$ means that $A$ is possible in at least one world view (i.e., $A$ belongs to at least one of the composing answer sets). An enhanced $\mathbf{M}^W$ can check whether $A$ is possible in *every* world view. $\mathbf{NOT}\, A$ is a shorthand for $\neg\mathbf{M}A$, meaning that $A$ is false in every answer set of a world view and $\mathbf{NOT}^W\, A$ means falsity of $A$ in every answer set of every world view. The usefulness of these operators will be illustrated in Section 7 where we further elaborate on the above example.

In general terms, the possibility of querying world views may have significant practical applications, for instance in Intelligent Agents[1], recommender systems, decision support systems: such systems might employ answer set modules for reasoning tasks (Costantini 2011), and might prefer to receive answers to queries rather that having to inspect all the answer sets/world views. **Notice that:** *query-answering might be simulated by suitable APIs defined on top of a full world-view computation. However, this would not be resilient to modifications of a program's knowledge base, as each modification would require to recompute the world views. Our query-answering facility does not need to compute full world views, and new queries automatically consider the updated knowledge. This is important in evolving systems such as, e.g., agents.*

We resort to the RAS semantics (Resource-based Answer Set Semantics, cf. (Costantini and Formisano 2015)) for formalizing the proposed new characterization of world views. RAS provides answer sets to all programs, including those which are inconsistent under AS. For consistent programs, which are those for which epistemic negation makes sense as some world view may exist, RAS returns the same answer sets as AS (under very simple conditions, seen below) and therefore the same world views. RAS however features, differently from AS, a prolog-like top-down query answering facility (Costantini and Formisano 2016).

[1] cf. (Bordini et al. 2006) for a survey about logical approaches to the definition of intelligent agents.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

The paper is organized as follows. In Sections 2–3 we recall Answer Set Programming, and Epistemic Logic Programs under the semantics of (Shen and Eiter 2016) (we assume a basic knowledge on logic programming and its declarative and procedural semantics as illustrated in standard textbooks, e.g., (Lloyd 1987)). Then, in Section 4 we propose some observations on ELPs that will lead to an alternative approach to the computation of world view via *epistemic scenarios*. We introduce Resource-Based Answer set semantics in Section 5, and then we show in Section 6 how by means of its associated query-answering device, RAS-XSB-resolution (Costantini and Formisano 2016), we are able to query ELPs so as to check the validity of guesses, and to query world views and sets of world views. In Section 7 we expand the case study introduced above, and finally we conclude. In this paper we discuss semantics and applications, but we treat implementation issues only in general terms. We intend in fact to provide here novel contributions that can be of theoretical interest by themselves, and can also in perspective be useful to developers and users of ELP programs and related applications. Coping with efficiency issues and performing experiments is deferred to a future implementation effort.

## 2 Answer Set semantics (AS) and Answer Set Programming (ASP)

"Answer Set Programming" (ASP) is a successful programming paradigm based on the Answer Set Semantics (Gelfond and Lifschitz 1988; Gelfond and Lifschitz 1991). In ASP, one can see an answer set program (for short, just "program") as a set of statements that specify a problem, where each answer set represents a solution compatible with this specification. Whenever a program has no answer sets (no solution can be found), it is said to be *inconsistent*, otherwise it is said to be *consistent*. Several well-developed freely available *answer set solvers* exist (Brewka et al. 2016), that compute the answer sets of a given program. Syntactically, an ASP program $\Pi$ is a collection of *rules* of the form '$H \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_{m+n}$.' where $H$ is an atom, $m, n \geqslant 0$, and each $A_i$, $i \leq m+n$, is an atom. Atoms and their negations are called *literals*. The left-hand side and the right-hand side of the clause are called *head* and *body*, respectively. A rule with empty body is called a *fact*. A rule with empty head is a *constraint*, where a constraint of the form '$\leftarrow L_1, ..., L_n$.' states that literals $L_1, \ldots, L_n$ cannot be simultaneously true in any answer set. Constraints are often rephrased as '$f \leftarrow not\ f, L_1, ..., L_n$.' where $f$ is a fresh atom. To avoid the contradiction over $f$, some of the $L_i$'s must be false thus forcing $f$ to be false, and this, if achieved, fulfills the constraint.

There are other features that for the sake of simplicity we do not consider in this paper, namely: (i) explicit disjunction; (ii) so-called classical negation $\neg A$, that can be however easily compiled away (Gelfond and Lifschitz 1991); (iii) advanced programming features such as aggregates; (iv) function symbols. In the rest of the paper, as it is customary in the ASP literature and as is done in (Shen and Eiter 2016), we will implicitly refer to the "ground" version of $\Pi$, which is obtained by replacing in all possible ways the variables occurring in $\Pi$ with the constants occurring in $\Pi$ itself, and is thus composed of ground atoms, i.e., atoms which contain no variables.

The answer set (or "stable model") semantics (AS) can be defined in several ways. However, answer sets of a program $\Pi$, if any exists, are supported minimal classical models of the program interpreted as a first-order theory in the obvious way. The original definition by (Gelfond and Lifschitz 1988) was in terms of the 'GL-Operator' $\Gamma$ where,

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

given set of atoms $I$ and program $\Pi$, $\Gamma_\Pi(I)$ is the least Herbrand model of $\Pi^I$, where $\Pi^I$, called the (Gelfond-Lifschitz) 'reduct' of $\Pi$ w.r.t. $I$, is a positive program (and so its least Herbrand model can be computed via the immediate consequence operator[2] $T_P$ applicable to any positive program $P$) obtained from $\Pi$ by: 1. removing from $\Pi$ all rules which contain a negative literal $not\, A$ such that $A \in I$; and 2. removing all negative literals from the remaining rules. $I$ is an answer set whenever $\Gamma_\Pi(I) = I$.

## 3 Epistemic Negation: Semantics

As discussed in the Introduction, we can understand Epistemic Logic Programs as answer set programs augmented with the epistemic negation operator **not**. In this paper, for the semantics of Epistemic Logic Programs we especially refer to the approach of (Shen and Eiter 2016), based on the notion of *World Views*: for a given program, instead of a set of answer sets like in ASP, there is now a set of such sets. Each one, called "world view", consistently satisfies the epistemic negations occurring in the given program (as well as the other modal expressions possibly defined in terms of **not**). An epistemic logic program may admit none or several world views. World views in (Shen and Eiter 2016) are obtained as follows. Let $\Pi$ be a (ground) epistemic program and let $EP(\Pi)$ be the set of literals of the form **not** $F$ occurring in $\Pi$. Given $\Phi \subseteq EP(\Pi)$, the *Epistemic reduct* $\Pi^\Phi$ of $\Pi$ w.r.t. $\Phi$ is obtained by: (i) replacing every **not** $F \in \Phi$ with true, and (ii) replacing every **not** $F \notin \Phi$ with $not\, F$. Then, the set $\mathcal{A}$ of the answer sets of $\Pi^\Phi$ 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 $\Phi$, or is corresponding to $\Phi$, or that it is a candidate world view w.r.t. $\Phi$, where $\Phi$ is called a *candidate valid guess*. $\mathcal{A}$ is a *world view* if it is maximal, i.e., there exists no other candidate world view obtained from guess $\Phi'$ where $\Phi \subset \Phi'$, and we call $\Phi$ a *valid guess*. This maximality condition, as discussed in (Shen and Eiter 2016), is essential for avoiding unintended world views. A literal $F$ is said to be true in $\Pi$ under the general epistemic semantics if $\Pi$ admits a world view $\mathcal{A}$ such that $F$ is true in every answer set in $\mathcal{A}$. The answer sets can be computed by any variant of the answer set semantics while retaining all formal properties of the approach.

The work (Shen and Eiter 2016) thoroughly studies the complexity of the proposed semantics, taking as a case study the FLP semantics of (Faber et al. 2011) for answer set programming, and establishes that deciding whether a program has a world view is at the third level of the polynomial hierarchy. They observe that complexity falls at the second level of the polynomial hierarchy if one refers to classical AS semantics, i.e., if one excludes explicit disjunction and additional programming constructs such as aggregates.

For the semantics of (Shen and Eiter 2016), the work (Bichler et al. 2018) proposes an ingenious method to compute and enumerate all the world views via an ASP solver. Precisely, they define a metaprogram which has first a stage of guessing a truth assignment of epistemic literals, and then a phase of checking. The method is effective, needs just one call to an ASP solver, and it is also optimal from a complexity-theoretic point of view. As this encoding makes use of large non-ground rules, specialized grounding methods must be adopted.

[2] As already mentioned, for general terminology about logic programming a reader may refer to (Lloyd 1987)

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

## 4 Observations

In this section we propose some observations concerning programs with epistemic disjunction under the semantics of (Shen and Eiter 2016), applied however for the sake of simplicity to plain AS semantics. Thus, the kind of programs that we consider are ASP programs where: (i) there is no explicit disjunction in the head or body of rules; (ii) both default negation (that we indicate with *not*) and epistemic negation **not** may appear (only) in the body of rules; (iii) concerning nested negations, we accept *not* **not** $A$ ($\mathbf{K}A$), **not** *not* $A$ ($\mathbf{M}A$), and *not* **not** *not* $A$ ($\mathbf{K}$ *not* $A$, i.e., *not* $\mathbf{M}A$), where $A$ is an atom; we do not consider *not not* $A$ for lack of space; (iv) like (Shen and Eiter 2016), we do not explicitly consider *classical negation* $\neg A$.

Consider for instance the following programs, aimed to study how world views are formed in similar though different significant cases.

$\Pi_1$

$a \leftarrow not\ b.$ $\quad b \leftarrow not\ d.$ $\quad d \leftarrow not\ b.$

$\Pi_2$

$a \leftarrow \mathbf{not}\ b.$ $\quad b \leftarrow not\ d.$ $\quad d \leftarrow not\ b.$

$\Pi_3$

$a \leftarrow \mathbf{not}\ b, not\ b.$ $\quad b \leftarrow not\ d.$ $\quad d \leftarrow not\ b.$

$\Pi_1$ (involving no epistemic literals) has the unique world view $\{\{b\},\{a,d\}\}$ coinciding with its set of answer sets. It can be easily verified that $\Pi_2$ (involving the epistemic literal **not** $b$) has the unique world view $\{\{a,b\},\{a,d\}\}$ as in fact there exists an answer set of the even cycle involving $b$ and $c$ where $b$ is false: this makes **not** $b$ true, and therefore $a$ can always be derived. Instead, $\Pi_3$ has the unique world view $\{\{b\},\{a,d\}\}$, coinciding with the one of $\Pi_1$. This because the presence of *not* $b$ in the body of the first rule of $\Pi_3$ makes literal **not** $b$ completely irrelevant: in fact, if we suppose that **not** $b$ is true then the truth value of the head is determined by *not* $b$; if **not** $b$ is false then so is *not* $b$, as there can be no answer sets where $b$ is false. Therefore, for the sake of simplicity and without loss of generality we assume that the body of each rule may include either **not** $A$ or *not* $A$, where $A$ is an atom, but not both. We assume also that $\mathbf{K}$ and $\mathbf{M}$ are applied to atoms occurring as the head of some rule in given program. Otherwise in fact, $\mathbf{K}A$ and $\mathbf{M}A$ are trivially false while their negations are trivially true.

It is important to notice that a given program may admit multiple world views only in presence of conflicting epistemic assumptions, like, e.g., in the example below:

$a \leftarrow \mathbf{not}\ b.$ $\quad b \leftarrow \mathbf{not}\ a.$

where in fact there are the two world views $\{\{a\}\}$ and $\{\{b\}\}$. In the variant

$a \leftarrow c.$ $\quad c \leftarrow \mathbf{not}\ b.$ $\quad b \leftarrow d.$ $\quad d \leftarrow \mathbf{not}\ a.$

there are the two world views $\{\{a,c\}\}$ and $\{\{b,d\}\}$ where however the conflicting assumptions are still **not** $a$ and **not** $b$, as the positive dependencies are to all effects irrelevant.

In order to assess which are the guesses that can be sensibly made to find world views, we propose the following method: (a) simplifying the given program so as to put into evidence cycles on epistemic literals; (b) finding the answer sets of such simplified version, and (c) extracting potentially valid guesses from these answer sets. As we will show, if valid guesses exist they correspond to some of these answer sets, whose number thus establishes an upper bound on the number of valid guesses and consequently of world views.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

*Definition 4.1* (*Epistemic simplified version of a program*)
Given program Π including epistemic literals[3], the *epistemic simplified version of* Π, that we call $\Pi^s$, is obtained via the following steps.

1. Preliminary step, for **M** and **K** elimination: replace each literal **not** *not A* with **not** $A'$, where $A'$ is a fresh atom; cancel each literal *not* **not** $A$.
2. Simplification, composed of the following steps:
   (i) Fold all positive dependencies internal to cycles, following the criteria provided in (Costantini and Provetti 2005); folding can be performed efficiently, for instance by exploiting the techniques discussed in (Brass et al. 2001)[4].
   (ii) Eliminate all rules with no epistemic literals in the body.
   (iii) For each remaining rule $\rho$, substitute the conjunction of non-epistemic literals in the body of the rule (if present) by a fresh atom $a_\rho$; for each such fresh atom, add the fresh even cycle $\{a_\rho \leftarrow not\, noa_\rho.\ \ noa_\rho \leftarrow not\, a_\rho.\}$ where also $noa_\rho$ is a fresh atom, to signify that the non-epistemic body of each rule can be either true or false.
   (iv) For each epistemic literal **not** $C$ such that $C$ does not occur in the head of any of the rules of $\Pi^s$ as obtained in previous steps (i.e., **not** $C$ is not involved in cyclic dependencies with other epistemic literals; notice that this case includes the **not** $A'$ since $A'$ cannot occur in rule heads) substitute **not** $C$ with fresh atom $NC$ and add the fresh even cycle $\{C \leftarrow not\, NC.\ \ NC \leftarrow not\, C.\}$ to signify that **not** $C$ can be assumed or not.
   (v) Substitute **not** with *not*.

After computing $\Pi^s$ (point (a) of the proposed method), we compute (point (b)) the answer sets $M_1, \ldots, M_n$ of $\Pi^s$, and then (point (c)) we determine the potentially valid guesses related to these answer sets. Since in building $\Pi^s$ the epistemic literals occurring in the original program Π have been transformed so as to admit only *not* as negation, in this step we restore such literals to their original form, and we get rid of the fresh atoms introduced in $\Pi^s$.

*Definition 4.2* (*Epistemic scenarios*)
Given program Π and its epistemic simplified version $\Pi^s$, for each answer sets $M_i$ of $\Pi^s$, $i \leq n$,

$$\begin{aligned} S_i = \ & \{\textbf{not}\ A \mid A \notin M_i\} \ \cup\ \{\textbf{not}\ A \mid NA \in M_i\} \ \cup\ \{\textbf{not}\ not\, A \mid NA' \in M_i\} \cup \\ & \{not\ \textbf{not}\ A \mid A \in M_i \text{ and } not\ \textbf{not}\ A \text{ occurs in } \Pi\} \cup \\ & \{not\ \textbf{not}\ not\, A \mid A' \in M_i \text{ and } not\ \textbf{not}\ not\, A \text{ occurs in } \Pi\} \end{aligned}$$

We call the $S_i$s the *epistemic scenarios* for Π, and $\mathcal{S}$ the set of such sets. We call $\mathcal{S}_M$ the set of maximal scenarios, i.e., $\mathcal{S}_M = \{S_i \in \mathcal{S} : \nexists S_j \in \mathcal{S}, S_i \subset S_j\}$.

We can prove that:

*Theorem 4.1*
For every world view $\mathcal{A}$ of a given program Π with set of epistemic scenarios $\mathcal{S}$, if $\mathcal{A}$ is obtained from guess Φ then $\Phi \in \mathcal{S}$.

[3] If given program does not contain epistemic literals then it admits a unique world view.
[4] Folding techniques may lead to exponential growth of program size but this does not happen if they are applied only inside cycles (cf. (Costantini and Provetti 2005)). We do not provide details about how such techniques are applied, because the step of folding is not really necessary for the proposed method to work: it is performed here only for the sake of clarity, in order to get rid of irrelevant non-epistemic literals and thus make Definition 4.2 (seen below) simpler.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

*Proof*
(sketch) As it is well known, a program may have several answer sets only if the program includes cycles on negation. In fact, in a cycle such as $a \leftarrow not\, b.\ b \leftarrow not\, a$, only one of $not\, a$ or $not\, b$ can be deemed to be true in each answer set (in absence of other rules supporting $a$ or $b$ or both). The semantics of (Shen and Eiter 2016) computes world views by means of the epistemic reduct, which is very similar to the (Gelfond-Lifschitz) reduct for computing answer sets, and thus presents the same idiosyncrasies. In fact, consider, e.g., a cycle $a \leftarrow \mathbf{not}\ b, a_\rho \quad b \leftarrow \mathbf{not}\ a, b_\rho$ where the fresh atoms $a_\rho$ and $b_\rho$ are assumed to be true. If making a guess including both **not** $a$ and **not** $b$ (i.e., assuming both of them to be true) then both $a$ and $b$ would belong to any answer set, thus determining a contradiction since both epistemic literals should then be false. So in this case, each valid guess may involve either **not** $A$ or **not** $B$, but not both. Thus, guesses which can possibly be valid (and thus generate a world view) must include sets of epistemic literals not conflicting with each other in cycles. Epistemic literals not involved in cycles are independent of the others, and may thus be freely assumed as true/false in any guess. Assumptions on nested negations follow in a straightforward way from assumptions on **not**. In fact, the proposed method determines the set $\mathcal{S}$ of potentially valid guesses on a simplified version of the program, obtained as specified in Definition 4.1. This simplified version is focused on epistemic literals though faithful to the program intended meaning, and thus makes it possible to compute via plain answer set computation all those guesses potentially corresponding to world views. □

*Corollary 4.1*
Let $\mathbf{n}$ be the number of atoms $A$ occurring in epistemic negations **not** $A$ and $n$ the number of rule heads in the given program $\Pi$, and let $\hat{n}$ be the number of rule heads occurring in its epistemic simplified version $\Pi^s$. The maximum number of valid guesses leading to world views of $\Pi$ is $\Theta(3^{\hat{n}/3})$, where we have $\Theta(3^{\hat{n}/3}) \simeq \Theta(3^{\mathbf{n}/3}) \simeq \Theta(3^{n/3})$.

This follows from the famous result by (Cholewinski and Truszczynski 1999) on the maximum number of answer sets of a given program that, they show, depends on cycles over negation. In fact, the valid guesses are among the epistemic scenarios $\mathcal{S}$, which are extracted from the answer sets of $\Pi^s$ whose number is $\Theta(3^{\hat{n}/3})$. The cycles over negation in $\Pi^s$ correspond (by Definition 4.1) to cycles over epistemic negations in $\Pi$, except for the addition of few fresh even cycles that do not significantly increase the size of $\Pi^s$ w.r.t. the size of $\Pi$.

**Our Contribution:** Corollary 4.1 *provides for the first time in the literature an upper bound on the number of valid guesses.*

Not all epistemic scenarios will in general correspond to valid guesses, as validity of a guess will depend upon effective true/falsity of non-epistemic literals occurring in rules. Validity can be verified as stated in (Shen and Eiter 2016) and mentioned in Section 3, or via the new method introduced below in Section 6. Theorem 4.1 however allows one to reduce the number of guesses to be considered. Though this cannot reduce complexity, it can nonetheless be of practical usefulness. Finding epistemic scenarios may provide in the average case (when not all guesses are valid) an advantage if applied, e.g., as a pre-processing stage for the method of (Bichler et al. 2018). In fact, this method might thus consider, represent and check fewer guesses, where each guess is represented by a

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

long rule in the metaprogram that they adopt, and is therefore computationally heavy. Since however computing epistemic scenarios has a cost, the effective usefulness will have to be evaluated by experiments that assess "how small" **n** should be in order to obtain an advantage. Or, one might allot a predefined threshold proportional to **n** to the running time of the pre-processing stage, that would be stopped if exceeding this time: so, pre-processing would bring either a substantial advantage or a negligible disadvantage.

## 5 Resource-based Answer Set semantics (RAS)

Resource-based Answer Set semantics (RAS, presented in (Costantini and Formisano 2015)) has the property to provide answer sets to every program (so, under RAS there are no inconsistent programs). The principles underlying the RAS semantics are the following: (i) atoms belonging to a RAS answer set are either definitely true (i.e., a RAS answer set includes all atoms that are true w.r.t. the well-founded semantics[5]), or have been rationally (though defeasibly) assumed to hold; (ii) atoms not included in a RAS answer set are either definitely false (i.e., false w.r.t. the well-founded semantics), or have been rationally assumed not to hold in order to draw some conclusion, or no judgment about them has been devised because any such assessment would lead to contradiction. For instance, program $\{p \leftarrow not\ p.\}$ has an empty RAS answer set because, in our view, a rational agent cannot believe both $not\ p$ and $p$, so no rational judgment on truth value of $p$ can be given; differently from AS, this is not seen as a reason not to provide a semantics. Therefore, a RAS answer set: includes atoms either proved or assumed to be true; does not contain atoms either proved or assumed to be false, or whose truth value cannot be assessed.

To formally explain the difference between AS and RAS we may resort to a modal logic formulation. For AS, as discussed in (Marek and Truszczyński 1993) an answer set programming rule can be transposed, to express its logical meaning, into its "modal image", where $L\,A$ is intended as "$A$ is believed" under any modal logic contained in **S5**:

$$L\,A_1 \wedge \cdots \wedge L\,A_n \wedge L \neg L\,B_1 \wedge \cdots \wedge L \neg L\,B_m \supset L\,A \qquad (Ae0)$$

In RAS, as discussed in (Costantini and Formisano 2015), the logical meaning of each rule $\rho$ is expressed by the following couple of modal rules, that form its modal image:

$$L\,A_1 \wedge \cdots \wedge L\,A_n \wedge L \neg L\,B_1 \wedge \cdots \wedge L \neg L\,B_m \supset L\,\dot{A} \qquad (Ae1)$$
$$L\,\dot{A} \wedge \neg L \neg L\,A \supset L\,A \qquad (Ae2)$$

$(Ae1)$ modifies $(Ae0)$ in the sense that, based on the same premises, one concludes $L\dot{A}$, which means that one believes *to be enabled* to prove $A$. $(Ae2)$ states that $L\,A$ is derived only if $L\,\dot{A}$ holds, and one does not believe not to believe $A$. Thus, in the case for instance of the unary odd cycle $p \leftarrow not\ p$ which makes an ASP program inconsistent, by $(Ae1)$ one from $not\ p$, that in terms of the modal image is expressed as $L \neg L\,p$, can derive $L\,\dot{p}$, i.e., to be enabled to prove $p$. However, one cannot do so, as $L \neg L\,p$ precisely accounts to believing not to believe $p$. Consequently $(Ae2)$ cannot be applied, and $p$ therefore is false without raising inconsistencies.

[5] The well-founded semantics (Van Gelder et al. 1991) provides to every program a unique three-valued model $\langle W^+, W^- \rangle$, where atoms in $W^+$ are *true*, those in $W^-$ are *false*, and all the others are *undefined*. All atoms in $W^+$ are true in every AS answer set, and all atoms in $W^-$ are false; the answer set semantics (AS) in fact assigns, for consistent programs, truth values to the undefined atoms, and so does RAS, though in a slightly different way.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

Each AS answer set is also a RAS answer set, but a RAS answer set is not necessarily a classical model of the program. RAS answer sets are in fact all the "Maximal Consistently Supported" (MCS) sets of atoms that a given program $\Pi$ admits. I.e., for each RAS answer set $M$ of $\Pi$: every atom $A$ in $M$ is *consistently supported* by a set $S$ of rules of $\Pi$, where each rule in $S$ is supported in $M$, $A$ is the head of exactly one rule in $S$, and each atom (positive literal) in the body of rules in $S$ is different from $A$, and is in turn consistently supported; moreover, $M$ is maximal, meaning that there exists no $M'$ with the same properties such that $M \subset M'$. E.g., for program $\{a \leftarrow not\, b.\ \ b \leftarrow not\, c.\ \ c \leftarrow not\, a.\}$, the sets of atoms $\{a\}$, $\{b\}$ and $\{c\}$ are the only MCSs and thus they are the RAS answer sets of this program. In fact, each of the composing atoms $a, b, c$'s is consistently supported by set $S$ consisting of the single rule of which that atom is the head and the three sets are maximal, as all their supersets (among which are the classical models) are not consistently supported. We made the examples of unary and ternary odd cycles because what makes programs inconsistent under AS are indeed (direct or indirect) odd loops, i.e., when an atom depends upon its own negation through an odd number of negative dependencies.

RAS answer sets of program $\Pi$ can be computed via an operator $\hat{\Gamma}_\Pi$ which is analogous to $\Gamma_\Pi$ though it takes as input those interpretations $I$ such that for every $A \in I$ there exists rule $\rho$ in $\Pi$ with head $A$, and is based upon: a modified reduct (that, with respect to the 'traditional' reduct recalled in Section 2, does not perform step 2), and a modified $T_\Pi$ operator, that computes consistently supported sets of atoms by discarding those atoms for which all possible derivations depend on their own negation. In fact, as discussed before such atoms are excluded from any RAS answer set because their truth value cannot be assessed. $M = \hat{\Gamma}_\Pi(I)$ is a RAS answer set iff $M \subseteq I$ and $M$ is maximal, i.e., there is no proper subset $I_1$ of $I$ that determines $M' \supset M$.

On programs which are consistent under AS, 'traditional' answer sets are among RAS answer sets, that can be more numerous. E.g., program $\{a \leftarrow not\, b.\ b \leftarrow not\, a.\ p \leftarrow not\, p,\ a.\}$ has unique answer set $\{b\}$ under AS and answer sets $\{a\}$ and $\{b\}$ under RAS. For AS, falsity of $a$ is required in order to bypass the odd cycle by falsifying $p$. For RAS this is no longer required, as $p$ is excluded from any answer set because it depends on its own negation. Odd cycles are in fact the only possible source of difference between RAS and AS on programs which are consistent under AS. For programs that are either 'call-consistent' (i.e., they do not involve odd cycles) or that fulfill straightforward syntactic sufficient conditions concerning odd cycles, the answer sets returned by the two semantics are exactly the same. So, RAS can be seen as a variant of AS. Sufficient conditions are based on the fact that, as discussed in (Costantini 2006), a program including odd cycles can be consistent under AS only if for each odd cycle one of the following conditions hold: (i) some rule involved in the cycle has conjuncts that, if false, force falsity of the rule head or (ii) some atom involved in the cycle is the head of a rule outside the cycle where conjuncts in its body, if true, force truth of the head atom; in both cases, the cycle is 'broken' and becomes harmless, where these conjuncts are called "handles" of the cycle. Thus, a first sufficient condition requires that atoms appearing in the handles of some odd cycle do not occur, either directly or indirectly through dependencies, in other cycles or in their handles: this in fact ensures that different cycles do not interact. A second sufficient condition, which is more restrictive but easier to verify, requires that atoms occurring in handles of odd cycles do not occur elsewhere in the program.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

In perspective, as future work it is possible to study and possibly devise other, more fine-grained, sufficient conditions.

Differently from AS, RAS enjoys the properties that, ideally, every non-monotonic formalism should enjoy (Dix 1995): (i) cumulativity, i.e., the possibility of asserting lemmas while keeping the same answer sets; (ii) relevance, i.e., the fact that the truth value of each atom is determined by the subprogram consisting of the *relevant rules*, which are those upon which the atom depends (directly or indirectly, positively or negatively). AS does not enjoy such properties because some atoms may be forced to assume certain truth values in order to prevent inconsistencies; this is no longer the case for RAS. An advantage of the property of relevance is to make top-down query-answering possible (Costantini and Formisano 2016) in prolog-style, i.e., without computing the answer sets in advance but rather via an enhanced resolution procedure. This because in RAS only the relevant rules have a role in proving/disproving any atom $A$. Instead, under AS a query $?A$ might *locally* succeed, but still, for the lack of relevance, the overall program may not have answer sets including $A$. Under RAS, a query $?A$ w.r.t. ground program $\Pi$ asks whether $A$ is true in (belongs to) some answer set of $\Pi$. Query *not* $A$ asks whether *not* $A$ is true in some answer set of $\Pi$, which implies that there exists some answer set to which $A$ *does not* belong. Series of queries can be, upon user's choice: (i) *contextual*, i.e., query $?A, B$ asks whether $A$ is true (belongs to) some answer set, and $B$ is true in (belongs to) some of those; (ii) independent.

RAS query-answering is performed via RAS-XSB-resolution, described in (Costantini and Formisano 2016). It is implemented on top of XSB-resolution (Swift and Warren 2012; Chen and Warren 1993), which is an efficient, fully described and implemented procedure, correct and complete w.r.t. the well-founded semantics. Features of XSB-resolution that are crucial for the implementation of RAS-XSB-resolution are negative cycles detection and the tabling mechanism, that associates to program $\Pi$ a table $\mathcal{T}(\Pi)$, which is initialized prior to posing queries. Such table contains information about true and false atoms useful for both the present and the subsequent queries (if they are in conjunction). The principle of functioning for table $\mathcal{T}(\Pi)$ under RAS-XSB-resolution are the following. During a proof: (a) the negation of any atom which is not a program fact is available unless this atom has been proved; (b) the negation of an atom which has been proved becomes unavailable, and the atom is asserted as true; so, (c) the negation of an atom which cannot be proved remains always available. Since RAS-XSB-resolution is a top-down proof procedure, modifications to the table might be undone and redone differently upon backtracking. The present (prototype) implementation exploits XSB (or, more precisely, its basic version XOLDTNF), as a "plugin" for definite success and failure, where new cases are added to manage atoms with truth value *undefined* under XSB. RAS-XSB-resolution is correct and complete w.r.t. resource-based answer set semantics, in the sense that, given a program $\Pi$, a query $?A$ succeeds under RAS-XSB-resolution with an initialized $\mathcal{T}(\Pi)$ iff there exists some resource-based answer set $M$ for $\Pi$ where $A \in M$. The result extends to sequences of queries.

A consequence of rendering odd cycles consistent is that under RAS constraints must be defined as such, since they can no longer be rephrased in terms of unary odd cycles. So, constraints must be associated to the program as an additional layer. However, as RAS answer sets computation is a variant of answer set computation, constraints can still be checked during computation as solvers usually do. In what follows, when clear from the context, we will call RAS answer sets simply 'answer sets'.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

## 6 Epistemic Negation in RAS

It can be easily seen that the approach of (Shen and Eiter 2016) is applicable to RAS (in fact, the approach has been devised in order to be applicable to *any* variant of the AS semantics). Indeed, as seen in Section 3, after performing the epistemic reduct $\Pi^{\Phi}$ of given program $\Pi$ w.r.t. a guess $\Phi$, the set $\mathcal{A}$ of answer sets of $\Pi^{\Phi}$ is computed, which is a candidate world view to be checked for validity and maximality. If, instead of computing the answer sets, one computes RAS answer sets, or even if one adopts some other semantics which produces sets of sets as result, all the rest remains unchanged. In fact, in Section 4 we have shown that epistemic scenarios can be found by considering epistemic literals only.

In this Section we provide the following **contributions**:

1. We define a method to check validity of guesses without computing the world views, via RAS query-answering.
2. We introduce more generally the possibility to **query world views, and to query the whole set of world views**, and we define useful operators.

So, a reader can either suppose to adopt RAS instead of AS, or (s)he can assume that for any given program $\Pi$ the AS and RAS answer sets coincide (for instance, because $\Pi$ fulfills one of the above-mentioned sufficient conditions).

Consider first a program $\Pi$ which does not contain epistemic negation. Its unique world view will thus coincide with the set of its answer sets. Via RAS-XSB-resolution, we are able to implement epistemic queries, among which the following (where $A$ is an atom):

- Query ? **not** $A$ asks whether $A$ is false w.r.t. some answer set of $\Pi$, and therefore succeeds if *not* $A$ is true in some of them. This can be implemented via RAS query ?*not* $A$.
- Query ? **not** *not* $A$ asks whether *not* $A$ is false in some answer set, and therefore succeeds if $A$ is true in some of them, which corresponds to query ? $\mathbf{M}A$. This can be implemented via RAS query ?$A$.
- Query ? *not* **not** $A$ asks whether it is not true that $A$ is false w.r.t. some answer set of $\Pi$, i.e., that $A$ is true in all of them, which corresponds to ? $\mathbf{K}A$. This can be implemented via RAS query ?$A$, *not* $A(fail)$, where this overall query succeeds if ? $A$ succeeds, i.e., $A$ belongs to some answer set of $\Pi$, whereas ? *not* $A$ fails, so $A$ is not false in any of them.
- Query ? *not* **not** *not* $A$ asks whether $A$ is false in every answer set, meaning $\mathbf{K}$ *not* $A$, i.e., *not* $\mathbf{M}A$. This can be implemented via RAS query ?$A(fail)$ that succeeds if $A$ fails, i.e., exactly whenever there is no answer set where $A$ is true. We introduce a new operator **NOT** as a shorthand for *not* **not** *not* $A$.

We will now proceed to consider programs including epistemic negation. We have seen in Section 4 how to identify plausible guesses for given program $\Pi$. Then one has to verify, for each guess $\Phi$, that it indeed corresponds to a world view. To check a guess under RAS, one can exploit the following method, alternative to computing all the answer sets of $\Pi^{\Phi}$.

*Theorem 6.1* (*RASCGK test: RAS Candidate Guess check*)
Potential validity of a guess $\Phi$ w.r.t. program $\Pi$ (i.e., the fact that $\Phi$ corresponds to a candidate world view) can be checked as follows.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

1. Derive $\Pi^{\Phi}$ as said in Section 3, i.e., (i) delete from $\Pi$ all the epistemic literals belonging to $\Phi$ and, (ii) for all the epistemic literals which occur in $\Pi$ but are not in $\Phi$, substitute each of them with a fresh atom (a shortcoming for what done in (Shen and Eiter 2016), where in such literals they substitute **not** with *not*, which accounts to considering them false).
2. Given $\Pi^{\Phi}$: for every epistemic literal **not** $A$ occurring in $\Pi$ pose the query $?notA$; for every epistemic literal **not** *not* $A$ occurring in $\Pi$, pose the query $?A$; for every epistemic literal *not* **not** *not* $A$ occurring in $\Pi$, pose the query $?A(fail)$; and, for every epistemic literal *not* **not** $A$ pose the query $?A, not\ A(fail)$. Guess $\Phi$ is a *candidate valid guess* if all queries concerning epistemic literals $\mathcal{L} \in \Phi$ succeed, while instead all the others fail.

*Proof*
Straightforward given the above observations about the meaning of queries. □

In order to establish validity of a guess $\Phi \in \mathcal{S}$, one has to establish that $\Phi$ passes the RASCGK test whereas no superset $\Phi' \supset \Phi$ with $\Phi' \in \mathcal{S}$ does. In order to find *all* valid guesses, one can do the following: (i) check epistemic scenarios in $\mathcal{S}_M$ (which are the largest ones); those which pass the RASCGK test are valid guesses; (ii) for every $\bar{\Phi} \in \mathcal{S}_M$ which does not pass the test, check via the RASCGK test every $\bar{\Phi}' \in \mathcal{S}$ such that $\bar{\Phi}' \subset \bar{\Phi}$ starting from the sets with greater cardinality. Notice that the empty set may be a valid guess.

To implement world views querying, we exploit the tabling mechanism of XSB- and RAS-XSB-resolution, that as mentioned associates to program $\Pi$ a table $\mathcal{T}(\Pi)$, initialized prior to program execution. In order to query the program under a certain valid guess $\Phi$, the initialization is customized accordingly, by setting to true all and only the epistemic literals occurring in the valid guess under consideration.

*Definition 6.1* (*Guess-tailored table initialization*)
Given program $\Pi$ and a valid guess $\Phi$, *guess-tailored table initialization* will be performed in addition to normal initialization, in the following way: each epistemic literal occurring in $\Phi$ will be set to true, and all the other epistemic literals will be set to false.

*Definition 6.2* (*G-RAS-XSB-Resolution*)
The variant of RAS-XSB-resolution where, given program $\Pi$ and valid guess $\Phi$, guess-tailored table initialization is applied is called G-RAS-XSB-resolution (tailored to $\Phi$).

*Theorem 6.2*
Given program $\Pi$ and a valid guess $\Phi$, G-RAS-XSB-resolution tailored to $\Phi$ is correct and complete w.r.t. the world view $\mathcal{A}$ obtained from $\Phi$.

*Proof*
(sketch) RAS-XSB-resolution is correct and complete w.r.t. the answer sets of given program $\Pi$. Thus, the RASCGK test is able to correctly establish whether a guess $\Phi$ is valid. Since a world view consists of the (set of) answer sets of given program where however epistemic literals belonging to valid guess $\Phi$ are deemed to be true and the other epistemic literals are deemed to be false, and since guess-tailored table initialization does

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

exactly so, then G-RAS-XSB-resolution is correct w.r.t. the world view resulting from $\Pi$ given $\Phi$. □

For instance, to check whether $F$ is true in $\Pi$ under the general epistemic semantics of (Shen and Eiter 2016), i.e., to check whether $F$ belongs to every answer set of a the world view, it will suffice to issue the query $?\,\mathbf{K}F$ to this world view.

Moreover, via a further extension one can be able to query the whole set of world views.

*Definition 6.3* (*Extended multi-view program*)
Let $\Pi$ be a program, and $\Phi_1, \ldots, \Phi_k$ be the valid guesses for $\Pi$. Let the *extended multi-view program* $\Pi^E$ be an extended program obtained as the union of $k$ copies $\Pi^1, \ldots, \Pi^k$ of $\Pi$, where such copies have been however previously *standardized apart*, i.e., atoms occurring therein have been suitably renamed. Let as assume for instance that each atom $A$ in $\Pi$ becomes, in $\Pi^1, \ldots, \Pi^k$, respectively $A^1, \ldots, A^k$.

*Definition 6.4* (*Multi-guess table initialization*)
Given program $\Pi$, given its associated valid guesses $\Phi_1, \ldots, \Phi_k$, and given the *extended multi-view program* $\Pi^E$, *multi-guess table initialization* will be performed by: performing guess-tailored table initialization w.r.t. each $\Phi_1, \ldots, \Phi_k$, where however these guesses have been previously *standardized apart* correspondingly to what done for $\Pi^E$; i.e., each atom $A$ occurring in some literal of $\Phi_i$ $(i \leq k)$ becomes $A^i$.

*Definition 6.5* (*W-G-RAS-XSB-Resolution*)
The variant of G-RAS-XSB-resolution where, given program $\Pi$ and valid guesses $\Phi_1, \ldots, \Phi_k$, multi-guess table initialization is applied to $\Pi^E$, is called W-G-RAS-XSB-resolution (tailored to $\Phi$).

Now, by W-G-RAS-XSB-resolution it is possible to issue the following queries on given program $\Pi$:

- Query $?\,\mathbf{K}^W A$ asks whether $A$ is true in all world views of $\Pi$. This query is translated into query $?\,\mathbf{K}A^1, \ldots, \mathbf{K}A^k$ to be executed on $\Pi^E$ (where comma stands for conjunction, i.e., for this query to succeed all conjuncts must succeed).
- Query $?\,\mathbf{M}^W A$ asks whether $A$ is possible in some world view of $\Pi$. This query is translated into query $?\,\mathbf{M}A^1; \ldots; \mathbf{M}A^k$ to be executed on $\Pi^E$ (where semicolon stands for disjunction, i.e., for this query to succeed at least one disjunct must succeed). We might of course modify $\mathbf{M}^W$ so as to check whether $A$ is possible in *every* world view, by substituting ';' with ','.
- Query $?\,\mathbf{not}^W A$ asks whether $A$ is false w.r.t. some world view of $\Pi$, and therefore if *not* $A$ is true in some of them. So, this corresponds to query $?\mathit{not}\, A^1; \ldots; \mathit{not}\, A^k$ to be executed on $\Pi^E$ (where semicolon again stands for disjunction).
- Query $?\,\mathbf{NOT}^W A$ asks whether $A$ is false in every answer set of every world view, translated under W-G-RAS-XSB-resolution as $?\,\mathbf{NOT}\, A^1, \ldots, \mathbf{NOT}\, A^k$.

This amounts to introducing new epistemic operators, that can be useful in practice as shown by the example presented in the next section.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

## 7 Case Study

In this section we will reconsider the example presented in the Introduction. Let us elaborate on this example, stating that a person is guilty if a witness recognizes that person (say, when perpetrating the crime):

$guilty(X) \leftarrow suspect(X), witness_recognizes(X).$
$witness_recognizes(X) \leftarrow suspect(X), witness1_recognizes(X).$
$witness_recognizes(X) \leftarrow suspect(X), witness2_recognizes(X).$

Assume to have the fact $suspect(john)$.

Now, if we have also a fact, e.g., $witness1_recognizes(john)$, then this will be true in every answer set and thus $guilty(john)$ will be provable. Assume instead that witness number one is not sure, which can be represented as:

$witness1_recognizes(john) \leftarrow not\ witness1_not_recognizes(john)$
$witness1_not_recognizes(john) \leftarrow not\ witness1_recognizes(john).$

In this case, if we have no information about evidence provided by the other witness, then $innocent(john)$ will be provable, as there is an answer set where both $witness_recognizes(john)$ and thus $guilty(john)$ are false.

Assume now that we have $witness1_recognizes(john)$. Assume that instead we have been informed that $witness2_recognizes(john)$ is false. So, we can state that the two disagree via a fact $disagree_w1_w2(john)$. If we do not want to give priority to any of the witnesses, in absence of other information we may state:

$guilty(X) \leftarrow suspect(X), witness_recognizes(X).$
$witness_recognizes(X) \leftarrow witness1_recognizes(X), reliable(witness1, X).$
$witness_recognizes(X) \leftarrow witness2_recognizes(X), reliable(witness2, X).$

Whenever two witnesses propose contrary evidence on someone/something, then they cannot be both reliable (concerning the situation at hand). This can be expressed as follows:

$reliable(witness1, X) \leftarrow disagree_w1_w2(X), \mathbf{NOT}\ reliable(witness2, X).$
$reliable(witness2, X) \leftarrow disagree_w1_w2(X), \mathbf{NOT}\ reliable(witness1, X).$

We use the new proposed operator **NOT** because **not** is too weak: it is not sufficient, to deem a witness to be reliable, to state that the other one *may be* unreliable. In such case as this one should seek certainty, i.e., **NOT** checks falsity in *every* answer set of a world view. Notice that the above formulation results in a cycle on the operator **NOT**, namely, in abstract terms $\{a \leftarrow \mathbf{NOT}\, b.\ b \leftarrow \mathbf{NOT}\, a.\}$ which gives rise to alternative world views for exactly the same reasons stated in Theorem 4.1 for **not**. So, at the present stage we have two world views:

$\{\{suspect(john), witness1_recognizes(john), disagree_w1_w2(john),$
$reliable(witness1, john), witness_recognizes(john), guilty(john)\}\}$
and
$\{\{suspect(john), witness1_recognizes(john), disagree_w1_w2(john),$
$reliable(witness2, john), innocent(john)\}\}$

We can see that in the first world view *john* is guilty, while in the second one he is not. Here we can appreciate the usefulness of the queries over the whole set of world views:

? $\mathbf{K}^W\, guilty(john)$ asks whether *john* is guilty in every world view, and the answer is negative;

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

? $\mathbf{M}^W$ *guilty*(*john*) asks whether *john* is possibly guilty in some world view, and the answer is positive.

We can rephrase the original example as follows:
*innocent*$(X) \leftarrow$ *suspect*$(X)$, *provably_innocent*$(X)$.
*innocent*$(X) \leftarrow$ *suspect*$(X)$, *presumed_innocent*$(X)$.
*provably_innocent*$(X) \leftarrow \mathbf{NOT}^W$ *guilty*$(X)$.
*presumed_innocent*$(X) \leftarrow \mathbf{not}\, ^W$ *guilty*$(X)$.

In this formulation, *john* is *presumed_innocent* because there exists indeed a world view where he is not guilty. However, he is not provably innocent because there will exist world views where he is guilty. A court would deem *john* innocent in both cases. However, prior to court, the law enforcement might decide for supplementary investigations (for instance, concerning reliability of witnesses) in order to ascertain a conclusion 'beyond any reasonable doubt'.

## 8 Conclusions

We have proposed alternative characterizations of ELPs (Epistemic Logic Programs), and we have shown the usefulness of our approach in formalizing significant examples. Future directions include the extension of the approach so as to make the syntax of the programs that we consider more general. An implementation is also in order, to replace the embryonic prototype on which we have run the proposed examples. Then, experiments will be performed on suitable benchmarks in order to assess the effective practical value of the approach.

## References

Bichler, M., Morak, M., and Woltran, S. 2018. Single-shot epistemic logic program solving. In *Proc. of the Twenty-Seventh Intl. Joint Conf. on Artificial Intelligence, IJCAI 2018*, J. Lang, Ed. ijcai.org, 1714–1720.

Bordini, R. H., Braubach, L., Dastani, M., Fallah-Seghrouchni, A. E., Gómez-Sanz, J. J., Leite, J., O'Hare, G. M. P., Pokahr, A., and Ricci, A. 2006. A survey of programming languages and platforms for multi-agent systems. *Informatica (Slovenia) 30,* 1, 33–44.

Bowen, K. and Kowalski, R. A. 1983. Amalgamating language and metalanguage in logic programming. In *Logic Programming*, K. Clark and S. A. Tarnlund, Eds. Academic Press.

Brass, S., Dix, J., Freitag, B., and Zukowski, U. 2001. Transformation-based bottom-up computation of the well-founded model. *Theory and Practice of Logic Programming 1,* 5, 497–538.

Brewka, G., Eiter, T., and Truszczyński, M. (eds.) 2016. Answer set programming: Special issue. *AI Magazine 37,* 3: 5-6.

Cabalar, P., Fandinno, J., and del Cerro, L. F. 2019a. Founded world views with autoepistemic equilibrium logic. In *Logic Programming and Nonmonotonic Reasoning - 15th Intl. Conf., LPNMR 2019, Proceedings*, M. Balduccini, Y. Lierler, and S. Woltran, Eds. Lecture Notes in Computer Science, vol. 11481. 134–147.

Cabalar, P., Fandinno, J., and del Cerro, L. F. 2019b. Splitting epistemic logic programs. In *Logic Programming and Nonmonotonic Reasoning - 15th Intl. Conf., LPNMR 2019, Proceedings*, M. Balduccini, Y. Lierler, and S. Woltran, Eds. Lecture Notes in Computer Science, vol. 11481. 120–133.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

Chen, W. and Warren, D. S. 1993. A goal-oriented approach to computing the well-founded semantics. *Journal of Logic Programming 17,* 2/3&4, 279–300.

Cholewinski, P. and Truszczynski, M. 1999. Extremal problems in logic programming and stable model computation. *Journal of Logic Programming 38,* 2, 219–242.

Costantini, S. 2006. On the existence of stable models of non-stratified logic programs. *Theory and Practice of Logic Programming 6,* 1-2.

Costantini, S. 2011. Answer set modules for logical agents. In *Datalog Reloaded - First Intl. Workshop, Datalog 2010, Revised Selected Papers*, O. de Moor, G. Gottlob, T. Furche, and A. J. Sellers, Eds. Number 6702 in Lecture Notes in Computer Science. Springer, 37–58.

Costantini, S. and Formisano, A. 2015. Negation as a resource: a novel view on answer set semantics. *Fundamenta Informaticae 140,* 3-4, 279–305.

Costantini, S. and Formisano, A. 2016. Query answering in resource-based answer set semantics. *Theory and Practice of Logic Programming 16,* 5-6, 619–635.

Costantini, S. and Provetti, A. 2005. Normal forms for answer sets programming. *Theory and Practice of Logic Programming 5,* 6, 747–760.

del Cerro, L. F., Herzig, A., and Su, E. I. 2015. Epistemic equilibrium logic. In *Proceedings of the Twenty-Fourth Intl. Joint Conf. on Artificial Intelligence, IJCAI 2015*, Q. Yang and M. Wooldridge, Eds. AAAI Press, 2964–270.

Dix, J. 1995. A classification theory of semantics of normal logic programs I-II. *Fundamenta Informaticae 22,* 3, 227–255 and 257–288.

Faber, W., Pfeifer, G., and Leone, N. 2011. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence 175,* 1, 278–298.

Gelfond, M. 2011. New semantics for epistemic specifications. In *Logic Programming and Nonmonotonic Reasoning - 11th Intl. Conf., LPNMR 2011, Proceedings*, J. P. Delgrande and W. Faber, Eds. Lecture Notes in Computer Science, vol. 6645. Springer, 260–265.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of the 5th Intl. Conf. and Symposium on Logic Programming*, R. Kowalski and K. Bowen, Eds. MIT Press, 1070–1080.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9*, 365–385.

Gelfond, M. and Przymusinska, H. 1991. Definitions in epistemic specifications. In *Logic Programming and Non-monotonic Reasoning, Proceedings of the First Intl. Workshop*, A. Nerode, V. W. Marek, and V. S. Subrahmanian, Eds. The MIT Press, 245–259.

Kahl, P. T. and Leclerc, A. P. 2018. Epistemic logic programs with world view constraints. In *Technical Communications of the 34th Intl. Conf. on Logic Programming, ICLP 2018*, A. D. Palù, P. Tarau, N. Saeedloei, and P. Fodor, Eds. OASIcs, vol. 64. Schloss Dagstuhl, 1:1–1:17.

Lloyd, J. W. 1987. *Foundations of Logic Programming*. Springer-Verlag.

Marek, V. W. and Truszczyński, M. 1993. Reflective autoepistemic logic and logic programming. In *LPNMR, Logic Programming and Non-monotonic Reasoning, Proc. of the Second Intl. Workshop*, A. Nerode and L. M. Pereira, Eds. The MIT Press, 115–131.

Shen, Y. and Eiter, T. 2016. Evaluating epistemic negation in answer set programming. *Artificial Intelligence 237*, 115–135.

Son, T. C., Le, T., Kahl, P. T., and Leclerc, A. P. 2017. On computing world views of epistemic logic programs. In *Proc. of the Twenty-Sixth Intl. Joint Conf. on Artificial Intelligence, IJCAI 2017*, C. Sierra, Ed. ijcai.org, 1269–1275.

Su, E. I. 2017. A monotonic view on reflexive autoepistemic reasoning. In *Logic Programming and Nonmonotonic Reasoning - 14th Intl. Conf., LPNMR 2017, Proceedings*, M. Balduccini and T. Janhunen, Eds. Lecture Notes in Computer Science, vol. 10377. Springer, 85–100.

Su, E. I. 2019. Epistemic answer set programming. In *Logics in Artificial Intelligence - 16th European Conf., JELIA 2019, Proceedings*, F. Calimeri, N. Leone, and M. Manna, Eds. Lecture Notes in Computer Science, vol. 11468. Springer, 608–626.

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press

Swift, T. and Warren, D. S. 2012. XSB: Extending prolog with tabled logic programming. *Theory and Practice of Logic Programming 12,* 1-2, 157–187.

Truszczynski, M. Revisiting epistemic specifications. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, M. Balduccini and T. C. Son, Eds.

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

https://doi.org/10.1017/S147106841900019X Published online by Cambridge University Press