

Top-down Splitting Property for Epistemic Logic Programs

Stefania Costantini¹

¹*DISIM - Università dell'Aquila, Italy*

Abstract

In this paper we consider Epistemic Logic Programs (ELPs), which extend Answer Set Programming (ASP) with “epistemic operators”. There are several approaches to the semantics of such programs in terms of *World Views*, which are sets of belief sets. Recent work has proposed an analysis of the structure of ELPs in terms of a concept of “splitting”, in order to be able to modularly compute their semantics in a bottom-up fashion, analogously to ‘traditional’ ASP. The proposal is brilliant but the problem is, that few of the semantics that have been proposed so far enjoy this new “Epistemic Splitting Property”. Thus, the notion of modular computation of world views does not work for most of the cases. We analyse the possibility to change the perspective about how to exploit a splitting, shifting from a bottom-up to a top-down approach. Our new definition: (i) copes with concerns regarding, e.g. “unfoundedness” of world views and “subjective constraint monotonicity”; (ii) is applicable to many of the existing semantics; (iii) coincides with the bottom-up notion of splitting on a significant class of programs.

Keywords

Answer Set Programming, Epistemic Logic Programs, Epistemic Splitting

1. Introduction

In this paper we discuss Epistemic Logic Programs (first introduced in [1, 2]), that extend Answer Set Programs with introspective capabilities, where Answer Set Programming (ASP) [3] is a successful logic programming paradigm under the answer set semantics (AS) [4, 5]. As it can be found in the wide corpus of existing literature concerning ASP (cf., among many, [6, 7, 8, 9] and the references therein), this programming paradigm has encountered a remarkable success, and has been applied in many fields, e.g., information integration, constraint satisfaction, routing, planning, diagnosis, configuration, computer-aided verification, biology/biomedicine, knowledge management, etc. The approach to problem-solving proper of ASP consists in the following conceptually distinct steps: (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” (many solvers are freely available, cf. [10]); (iii) extracting the problem solutions from the answer sets.

Epistemic Logic programs (ELPs, in the following just ‘programs’ if not explicitly stated differently), extend ASP with *epistemic operators* that are able to introspectively “look inside” a program’s own semantics, which is defined in terms of its answer sets. In fact, **KA** means


14th Workshop on Answer Set Programming and Other Computing Paradigms

✉ stefania.costantini@univaq.it (S. Costantini)

🆔 0000-0002-5686-6124 (S. Costantini)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

that (ground) atom A is true in every answer set of the very program Π where $\mathbf{K}A$ occurs, whereas $\mathbf{M}A$ means that A is true in some of the answer sets of Π . The *epistemic negation operator* \mathbf{not} A expresses that A is *not provably true*, meaning that A is false in at least one answer set of Π . Epistemic operators, that form *subjective literals*, are interchangeable, in fact $\mathbf{K}A$ and $\mathbf{M}A$ can be rephrased as *not not* A and *not not* A , respectively (where *not* is ASP standard ‘default negation’) while \mathbf{not} A can be rephrased as *not* $\mathbf{K}A$. Consequently, in discussing the semantics of ELPs, many approaches consider explicitly only the operator \mathbf{K} (as it is done here), while some approaches, primarily [11], consider explicitly only the operator \mathbf{not} . 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 world view consistently satisfies (according to a given semantics) the epistemic expressions that appear in a given program. Many semantic approaches/characterizations for ELPs have been introduced beyond the seminal one of [1, 2], among which [12, 13, 14, 11, 15, 16, 17, 18, 19]. Their aim is essentially to avoid world views which are ‘unintended’ with respect to a given program, and to compute the world views that appear ‘intuitively adequate’. Some approaches also propose extensions to the basic paradigm. ELP solving systems (for some specific semantics) have been defined and implemented [20, 21, 22, 23, 24] on top of state-of-the-art ASP solvers, that are invoked (more than once) to generate and check potential world views.

Recent work presented in [18, 19, 25, 26, 27] has been aimed to extend to Epistemic Logic Programming notions which have been previously defined for ASP. Primarily, they consider *splitting* (introduced for ASP by [28]), which allows a program to be divided into parts in a principled way, so that the answer sets can be computed incrementally, starting from the answer sets of the bottom part, used to simplify the top part w.r.t. them, and then compute the answer sets of the simplified top part (such procedure can be iterated for as many levels as the program has been divided into, i.e., the top and the bottom could be split recursively). So, these works try first of all to extend the concept of splitting and the method of incremental calculation of the semantics (here, it is the world views that must be calculated). They thus define a notion of *Epistemic Splitting*, where top and bottom are defined w.r.t. the occurrence of subjective literals, and a corresponding property, which is respected by a semantics if it allows the world views to be computed bottom up (a precise definition is seen below). They then adapt to ELPs other properties of ASP, namely the fact that adding constraints leads to reduce the number of answer sets (*Constraint Monotonicity*, implied by the Epistemic Splitting Property), and *Foundedness*, meaning that atoms composing answer sets cannot have been derived through cyclic positive dependencies. They also define the class of *Epistemically Stratified Programs*, that admit a unique world view. The aim of this line of work is to extend to ELPs such properties because they have turned out to be convenient for ASP, where many useful results have stemmed from them. So, the authors believe that they might prove useful in ELPs as well. The problem is, virtually none of the semantics existing in the literature obeys these properties, except the original one of [2], and the authors’ own one, first presented in [25], called FAEEL, which has however been developed with these properties in mind.

We believe that the approach of [18, 19, 25, 26, 27] is the correct one to examine ELPs: rather than defining ‘yet-another-semantics’, they establish properties that a semantics should fulfil, and then they compare the existing semantics with respect to these properties (and the

same can be done for possible future new semantics). The introduction of a notion of epistemic splitting is brilliant, and highly useful. So, **we have no intention whatsoever to argue against the work of Cabalar et al.**, which we found inspiring. Only, in our opinion it cannot be said that, apart from those in [2, 25], all the other semantics presented in the literature deliver unreasonable results. Without denying the potential practical utility of the properties proposed therein, **we only tried to take a different stance, on the basis of the notion of epistemic splitting.**

Our Contributions:

1. We considered the original splitting property in ASP, defined by Lifschitz and Turner in [28] (say, L&T splitting); it is not by its very nature an operational notion, yet it has been procedurally adopted for iterative answer sets computation (cf., e.g., [29, 30, 31, 32]). We noticed that L&T splitting can be exploited to this aim both bottom-up (as commonly done) but also in a top-down way: in fact, a splitting set U for program Π defines a ‘bottom’ program, composed solely of atoms of U , and a ‘top’ program, where atoms from U can occur only on the body of rules. (Note that, the top part can be in turn split, over a number of layers). In a bottom-up fashion, basically, any layer can compute its answer sets given the answer sets of the lower layer(s), that allow truth values to be assigned to literals referring the that layer (literals involving “lower atoms”). In a top-down fashion, every layer could make all possible hypotheses about truth values of the set of lower atoms; it would calculate its own answer sets according to each hypothesis, and then discard answer sets deriving from those hypotheses not matching the actual answer sets of the lower layer, when they become known. This method is clearly somehow impractical, yet feasible; it is potentially useful in forms of modular programming, where a top part defining general (e.g., ontological) useful properties might be combined ‘on demand’ with any bottom part. It is easy to see that the bottom-up and top-down processes provide identical results.
2. We observed that every occurrence of a subjective literal is an act of introspection aimed to inspect, but also potentially to influence, the set of consequences which are derivable, and we considered that such influence, in order to be “global”, should spread bottom-up.
3. We take as granted the notion of Epistemic Splitting to subdivide the program into layers, but we then define an alternative process of incremental computation of world views, which operates top-down instead of bottom-up.
4. **Our results:** we prove that the proposed method of *Top-Down Epistemic Splitting* returns the world views that are computed by some of the most significant semantics proposed in the literature, and makes the problem of unfoundedness immaterial. The method is orthogonal to bottom-up Epistemic Splitting, and provides coincident results on the class of *Epistemically Stratified Programs*.
5. **Our conclusions:** we show that **not only FAEEL but also other semantics produce their results in a principled way, and obey useful properties**; both methods (bottom-up and top-down) for exploiting the splitting of ELPs may have their merits, that will have to be evaluated on practical applications. Such evaluation will be the subject of future work.

The paper is organized as follows. In Sections 2–3 we recall Answer Set Programming and Epistemic Logic Programs (we assume a basic knowledge on logic programming and its declarative and procedural semantics as illustrated in standard textbooks, e.g., [33]). In Section 4

we provide background on the proposal by [18, 19, 25, 26, 27]. In Section 5 we introduce some observations on ELPs that will lead to formulate our proposal, that we discuss in Section 6. Finally, in Section 7 we conclude.

2. Answer Set Programming (ASP) and Answer Set semantics

In ASP, one can see an answer set program (for short ‘ASP 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 (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 [6], that compute the answer sets of a given program.

Syntactically, an ASP program Π is a collection of *rules* of the form

$$A_1 | \dots | A_g \leftarrow L_1, \dots, L_m, \text{not } A_{m+1}, \dots, \text{not } A_n.$$

where each A_i , $i \leq g$, is an atom and $|$ indicates disjunction (that can be alternatively indicated as \vee), and the L_i s, $i \geq 1$, $m, n \geq 0$, are literals (i.e., atoms or negated atoms). The left-hand side and the right-hand side of the rule are called *head* and *body*, respectively. A rule with empty body is called a *fact*. Notation $A | B$ indicates disjunction, usable only in rule heads and, so, in facts. A rule with empty head (or, equivalently, with head \perp), of the form ‘ $\leftarrow L_1, \dots, L_n$.’ or ‘ $\perp \leftarrow L_1, \dots, L_n$.’, is a *constraint*, stating that literals L_1, \dots, L_n are not allowed to be simultaneously true in any answer set; the impossibility to fulfil such requirement is one of the reasons that make a program inconsistent.

All features of ASP not explicitly mentioned above are, for the sake of simplicity, not considered in this paper. As it is customary in the ASP literature, we implicitly refer to the ‘ground’ version of Π , which is obtained by replacing in all possible ways the variables occurring in Π with the constants occurring in Π 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 (see, for instance, [34] or also [35]). 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 by [4], introduced for programs where rule heads were limited to be single atoms, was in terms of the ‘GL-Operator’ Γ . Given set of atoms I and program Π , $\Gamma_\Pi(I)$ is defined as the least Herbrand model of Π^I , called the (Gelfond-Lifschitz) ‘reduct’ of Π w.r.t. I ; Π^I is positive program, so, its least Herbrand model can be computed via the standard immediate consequence operator (cf. [33]). Π^I is obtained from Π by: 1. removing all rules which contain a negative literal *not* A such that $A \in I$; and 2. removing all negative literals from the remaining rules. Then, I is an answer set whenever $\Gamma_\Pi(I) = I$.

3. Epistemic Logic Programs

Epistemic Logic Programs (ELPs), introduced in [1], 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 \perp). Such new literals are constructed via the *epistemic operator* \mathbf{K} (disregarding without loss of generality the other epistemic operators): $\mathbf{K}A$ means that (ground) atom A is true in every answer set of given program Π (it is a so-called *cautious consequence* of Π). The syntax of rules is analogous to ASP, save that literals can now be either objective or subjective, where subjective literals are allowed to appear (only) in the body of rules. Nesting of subjective literals is not considered here.

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 in Π . Let $Head(r)$ be the head of rule r , $Body_obj(r)$ be the (possibly empty) set of objective literals occurring in the body of rule r , and $Body_subj(r)$ be the (possibly empty) set of subjective literals occurring in the body of rule r . We call *subjective rules* those rules whose body is composed of subjective literals only.

The semantics of ELPs is 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 all subjective literals occurring in the program. Take for instance program $\{a \leftarrow not\ b, b \leftarrow not\ a, e \leftarrow not\ \mathbf{K}f, f \leftarrow not\ \mathbf{K}e\}$. Under every semantics, there are two world views: $\{\{a, e\}, \{b, e\}\}$, where $\mathbf{K}e$ is true and $\mathbf{K}f$ false, and $\{\{a, f\}, \{b, f\}\}$ where $\mathbf{K}f$ is true and $\mathbf{K}e$ false. Notice that the presence of two answer sets in each world view is due to the cycle on objective atoms, whereas the presence of two world views is due instead to the cycle on subjective atoms (c.f. [36] for a discussion).

4. Epistemic Logic Programs: useful Properties

As argued in [25] and [26], it would be useful if ELPs would enjoy (‘mutatis mutandis’) properties similar to those of ASP programs. So, in these works such useful properties are outlined and adapted, as we report (almost literally) below. To begin with, since several semantics for ELPs have been proposed, it is useful to abstract away from the specific semantic definition.

Definition 4.1. [*Slightly modified version of Definition 1 in [26] (Abstract semantics)*] An (abstract) semantics \mathcal{S} is a function mapping each program into sets of ‘belief views’, i.e., sets of sets of objective literals, where if Π is an objective program, then $\mathcal{S}(\Pi)$ is the set of stable models of Π . Given a program Π , each belief view in $\mathcal{S}(\Pi)$ is called a \mathcal{S} -world view of Π .

Drawing inspiration from the *Splitting Theorem* introduced in [28], which defines a subdivision of ASP programs into layers so as to be able compute the answer sets incrementally, an analogous properties is defined for ELPs.

Definition 4.2. [*Reported from Definition 2 in [26] (Epistemic splitting set)*] A set of atoms $U \subseteq At$ is said to be an *epistemic splitting set* of given program Π if for any rule r in Π one of the following conditions hold:

- (i) $Atoms(r) \subseteq U$,

(ii) $(\text{Body_obj}(r) \cup \text{Head}(r)) \cap U = \emptyset$.

An epistemic splitting of Π is a pair $\langle B_U(\Pi), T_U(\Pi) \rangle$ satisfying $B_U(\Pi) \cap T_U(\Pi) = \emptyset$ (meaning that they have no rules in common), $B_U(\Pi) \cup T_U(\Pi) = \Pi$, and also that all rules in $B_U(\Pi)$ satisfy (i) and all rules in $T_U(\Pi)$ satisfy (ii).

Intuitively, the second condition means that the top program may refer to atoms U which occur as heads of rules in the bottom, only through epistemic operators.

Epistemic splitting can be used, similarly to ‘traditional’ L&T splitting, for iterative computation of world views. In the case of ELPs, [25] proposes to compute first the world views of the bottom program $B_U(\Pi)$ and, for each one of them, simplify the corresponding subjective literals in the top part. Given an epistemic splitting set U for a program Π and a set of interpretations W , they define the subjective reduct of the top with respect to W and signature U , called $E_U(\Pi, W)$. This operator, according to [25], considers all subjective literals L occurring in $T_U(\Pi)$, such that the atoms occurring in them belong to $B_U(\Pi)$. In particular, L will be substituted by \top in $E_U(\Pi, W)$ if $W \models L$, and by \perp otherwise. So, $E_U(\Pi, W)$ is a version of $T_U(\Pi)$ where some subjective literal, namely those referring to the bottom part of the program, have been simplified as illustrated.

Definition 4.3 (Reported from Definition 3 in [26]). Given a semantics \mathcal{S} , a pair $\langle W_b, W_t \rangle$ is said to be an \mathcal{S} -solution of Π with respect to an epistemic splitting set U if W_b is a \mathcal{S} -world view of $B_U(\Pi)$ and W_t is a \mathcal{S} -world view of $E_U(\Pi, W_b)$.

The definition is parametric w.r.t. \mathcal{S} , as each different semantics \mathcal{S} will define in its own way the \mathcal{S} -solutions for a given U and Π , .

So, world views of the entire program will be obtainable by suitably combining some world view of the bottom with some world view of the top, i.e., the world views of the entire program should be obtained as (where I_b and I_t are answer sets occurring respectively in W_b and W_t):

$$W_b \sqcup W_t = \{I_b \cup I_t \mid I_b \in W_b \wedge I_t \in W_t\}$$

Therefore,

Property 4.1. [Property 1 in [26] (Epistemic Splitting Property)] A semantics \mathcal{S} satisfies epistemic splitting if, for any epistemic splitting set U of any given program Π : W is an \mathcal{S} -world view of Π iff there is an \mathcal{S} -solution $\langle W_b, W_t \rangle$ of Π with respect to U such that $W = W_b \sqcup W_t$.

As discussed at length in [25], some semantics satisfy epistemic splitting, and some others do not. Actually, most semantics do not satisfy this property, which is satisfied only in: the very first semantics of ELPs, proposed in [1] (and in some of its generalizations), and in Founded Autoepistemic Equilibrium Logic (FAEEL), introduced in [18].

The property of epistemic splitting implies *subjective constraint monotonicity* [19, 26]. I.e., if a semantics \mathcal{S} satisfies epistemic splitting then for any epistemic 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 .

Another property considered in [27] is *foundedness*, again extended from objective programs. A set X of atoms is *unfounded* w.r.t. program Π and interpretation I , if for every $A \in X$ there is

no rule of r by which A might be derived, without incurring in positive circularities and without forcing the derivation of more than one atom from the head of a disjunctive rule. For ELPs, [27] adds the condition that positive dependencies are also those on positive subjective literals, like, e.g., in program $A \leftarrow \mathbf{K}A$. Then, a world view for ELP Π will be *founded* if there is no composing interpretations \hat{I} which contains an unfounded set w.r.t. Π and \hat{I} . It turns out that, among existing semantics, only FAEEL satisfy foundedness.

5. Our Observations and Proposal

The subdivision of an ELP into layers as defined in [25, 26, 27] suggests that, in the upper layer, epistemic literals referring to the lower layer may be aimed to perform some kind of meta-reasoning about that layer. In the aforementioned approach however, meta-level reasoning is in practice prevented, as it is the lower layer that decides the truth value of the subjective literals that connect the two layers. In fact, according to the epistemic splitting property as defined in [26], through the simplification w.r.t. the answer sets of the lower layer, the upper layer is strongly (maybe sometimes too strongly) constrained. We can see that for instance, for program Π_0

$$a \vee b, c \leftarrow \mathbf{K}a, \leftarrow \text{not } c,$$

that once one has computed the unique world view of the lower level $a \vee b$ considered as a program ‘per se’, i.e., $\{\{a\}, \{b\}\}$, then the overall program has no world views: in fact, under this world view $\mathbf{K}a$ does not hold and so c is false, violating the constraint. The world view $\{\{a, c\}\}$, returned instead by semantics such as [12, 11], does not fulfil the epistemic splitting property. This world view may however be seen as corresponding to an approach where the upper layer, in order to retain consistency, ‘requires’ the lower layer to entail a , which is absolutely feasible by choosing a over b in the disjunction.

We follow (since a long time) the line, amply represented in the literature, in which meta-reasoning is aimed in general not only at ‘observing’ the lower layers, but also at trying to influence them (cf., e.g., [37] for a survey on the subject). So, we tried to look at the matter from another point of view, to understand whether the concept of splitting might be applied top-down, and how the existing semantics would behave in the new perspective. In our approach, the notion of splitting set remains the same, save for one detail. As noticed in [25], subjective constraints, and rules without objective literals in their body, according to the definition might be placed at either level. For convenience concerning definitions that will be introduced later, we impose the additional condition:

Definition 5.1 (Subjective rules and Constraints in Epistemic Splitting). *Given an epistemic splitting $\langle B_U(\Pi), T_U(\Pi) \rangle$ of a given program Π , subjective rules satisfying condition (ii) of the definition above, and subjective constraints, are put in $T_U(\Pi)$.*

Let us proceed step by step to the new definition of *Top-down Epistemic Splitting Property* (TDESP for short). As in the aforementioned previous work, we consider only the epistemic operator \mathbf{K} .

Let us consider an epistemic splitting of given program Π as a pair $\langle B_U(\Pi), T_U(\Pi) \rangle$ according to Definition 4.2. Let us, also, consider a semantics \mathcal{S} as given.

Definition 5.2 (Epistemic top-down Constraint set and Requirement set). *The Epistemic top-down subjective constraint set and the Requirement set concerning the world views of $T_U(\Pi)$ (obtained according to \mathcal{S}) are obtained as follows.*

- *Build from $T_U(\Pi)$ the new program $T'_U(\Pi)$ in the following way:*
 - *take the subjective literals $\mathbf{KL}_1, \dots, \mathbf{KL}_z$ occurring in $T_U(\Pi)$ but referring to $B_U(\Pi)$, in the sense that atoms involved therein occur in $B_U(\Pi)$ but not in $T_U(\Pi)$;*
 - *transform such literals into fresh atoms, kl_1, \dots, kl_z (keeping track of the correspondence);*
 - *for each of the kl_i s, add new fact $kl_i \mid \text{not } kl_i$ to $T_U(\Pi)$.*
- *Given the world views W'_1, \dots, W'_k of $T'_U(\Pi)$, for each such world view W' , identify the sets*

$$S_1 = \{kl_1, \dots, kl_r\}, \quad S_2 = \{kl_{r+1}, \dots, kl_s\}$$
where: all elements of both S_1 and S_2 are true in all sets composing the world view (i.e., they are cautious consequences), and all elements of S_1 are also directly or indirectly involved in a constraint in $T'_U(\Pi)$ (cf. [38] for a formal definition of direct and indirect dependencies).
- *Given the world views W'_1, \dots, W'_k of $T'_U(\Pi)$, cancel the kl_i 's and $\text{not } kl_j$'s from their composing sets, thus obtaining world views W_1, \dots, W_k for $T_U(\Pi)$ (after removing any empty set that might result, except if it is the only set composing the world view).*
- *Given each of the world views W_1, \dots, W_k of $T_U(\Pi)$, say W_i , the (possibly empty) set*

$$\{\mathbf{KL}_1, \dots, \mathbf{KL}_r\}$$
i.e., $\{\mathbf{KL}_i \mid 1 \leq i \leq r \wedge kl_i \in S_1\}$
is called Epistemic top-down Constraint set, indicated as $EC_{T_U(\Pi)}(W_i)$.
- *Given each of the world views W_1, \dots, W_k of $T_U(\Pi)$, say W_i , the (possibly empty) set*

$$\{\mathbf{KL}_{r+1}, \dots, \mathbf{KL}_s\}$$
i.e., $\{\mathbf{KL}_j \mid r+1 \leq j \leq s \wedge kl_j \in S_2\}$
is called Requirement set, indicated as $RQ_{T_U(\Pi)}(W_i)$.

The overall set $ECRQ_{T_U(\Pi)}(W_i) = EC_{T_U(\Pi)}(W_i) \cup RQ_{T_U(\Pi)}(W_i)$ is called Requisite Set, as it expresses prerequisites, about which epistemic literals must be entailed in some world view of $B_U(\Pi)$, so that such world view can be merged with W_i in order to obtain a world view of the overall program Π . We keep the two sets separate because, from a knowledge engineering point of view, it can be useful to distinguish literals in $EC_{T_U(\Pi)}(W_i)$, that if not entailed lead to a constraint violation and so to non-existence of a world view of Π containing W_i , from literals in $RQ_{T_U(\Pi)}(W_i)$, that instead correspond to mere assumptions.

In case, given world view W of $T_U(\Pi)$, literals belonging to $ECRQ_{T_U(\Pi)}(W)$ occur in the bodies of rules in $B_U(\Pi)$, our approach enforces the ‘required’ truth value of such literals by means of the following simplification.

Definition 5.3 (Top-down Influence). *Given world view W of $T_U(\Pi)$, and its corresponding requisite set $ECRQ_{T_U(\Pi)}(W)$, the W -tailored version $B_U^W(\Pi)$ of $B_U(\Pi)$ is obtained by substituting in $B_U(\Pi)$ all literals $\mathbf{KA} \in ECRQ_{T_U(\Pi)}(W)$ by A .*

World views of given program Π will be obtained, similarly to what done for the bottom-up approach, from world views of the top and the bottom, but with two important differences (i) Top-down Influence; (ii) a subset of a world view of the bottom (i.e., some of the answer sets occurring therein) can be cut out, so as to be combined with a ‘compatible’ world view of the top.

Definition 5.4 (Candidate World Views (CWWs)). A Candidate World View (CWW) W for given program Π (w.r.t. a semantics \mathcal{S}) is obtained as follows. Take a world view W_T of $T_U(\Pi)$ and a subset W_B of a world view of $B_U^{W_T}(\Pi)$ such that $\forall \mathbf{KL} \in EC_{RQ_{T_U(\Pi)}}(W_T), W_B \models \mathbf{KL}$. Then, from W_T and W_B , we have:

$$W = W_B \sqcup W_T = \{I_b \cup I_t \mid I_b \in W_B \wedge I_t \in W_T\}$$

Notice that, CWWs are computed after applying Top-down Influence. There can be the case that no subset of any world view of the bottom complies with the conditions posed by world views of the top; in this situation, Π has no candidate world views. Notice that the process can be iterated, in the sense that both $B_U(\Pi)$ and $T_U(\Pi)$ can in turn be split into a top and a bottom.

Definition 5.5 (Top-down Epistemic Splitting Property (TDESP)). A semantics \mathcal{S} satisfies Top-down Epistemic Splitting if any candidate world view of Π according to Definition 5.4 is indeed a world view of Π under \mathcal{S} .

Let us experiment this methodology on some of the examples proposed in recent literature. Consider program Π_1 , reported in [39].

$$\begin{array}{ll} p \mid q & (r1) \\ \perp \leftarrow \text{not } \mathbf{K}p & (C) \end{array}$$

Here, $B_U(\Pi_1)$ consists of rule (r1), and $T_U(\Pi_1)$ of constraint (C). So, $T'_U(\Pi_1)$ is (where kp is a fresh atom):

$$\begin{array}{ll} kp \mid \text{not } kp & \\ \perp \leftarrow \text{not } kp & (C) \end{array}$$

whose unique world view is $\{\{kp\}\}$. After cancelling kp , we obtain world view $W_T = \{\emptyset\}$ for $T_U(\Pi_1)$, where $EC_{T_U(\Pi_1)}(W_T) = \{\mathbf{K}p\}$ and $RQ_{T_U(\Pi_1)}(W_T) = \emptyset$. Regardless of \mathcal{S} , as no subjective literals occur therein, the unique world view of $B_U(\Pi_1)$ is $\hat{W} = \{\{p\}, \{q\}\}$. Since $W_B = \{\{p\}\}$ is only subset of \hat{W} fulfilling $EC_{T_U(\Pi_1)}(W_T)$, then it is the one selected by our method. It is also a world view for the overall program, as the unique world view of the top part is empty. This world view violates subjective constraint monotonicity, still it is the one delivered by the semantics of [11] and, as noticed in [39], by several other semantics among which [16, 17].

Consider now the following program Π_2 .

$$\begin{array}{ll} p \mid q & (r1) \\ p \leftarrow \mathbf{K}q & (r2) \\ q \leftarrow \mathbf{K}p & (r3) \\ \perp \leftarrow \text{not } \mathbf{K}p & (C) \end{array}$$

Here, $B_U(\Pi_2)$ consists of rules (r1-r3), and $T_U(\Pi_2)$ of constraint (C). So, $T'_U(\Pi_2)$ is (where kp is a fresh atom):

$$\begin{aligned} & kp \mid \text{not } kp \\ & \perp \leftarrow \text{not } kp \quad (C) \end{aligned}$$

whose unique world view is $\{\{kp\}\}$. After cancelling kp , we obtain world view $W_T = \{\emptyset\}$ for $T_U(\Pi_2)$ where $EC_{T_U(\Pi_2)}(W_T) = \{\mathbf{K}p\}$ and set RQ is empty. Regardless of \mathcal{S} , the potential world views of $B_U(\Pi_2)$ are $W_1 = \{\{p\}\}$, $W_2 = \{\{q\}\}$, $W_3 = \{\{p\}, \{q\}\}$, $W_4 = \{\{p, q\}\}$. W_4 is the only one fulfilling $EC_{T_U(\Pi_2)}(W_T)$; W_1 has the problem that, having p and fulfilling $\mathbf{K}p$, (r3) might be applied thus getting q . W_4 is in fact the world view returned by semantics such as [24, 11]. However, it is easy to see that W_4 violates foundedness as defined in [25]. Notice that, in our approach q is not derived via the positive cycle (extended to subjective literals), but from the $\mathbf{K}p$ “forced” by the upper layer via Top-down Influence, which substitutes $\mathbf{K}p$ with p in rule (r3) of $B_U(\Pi_2)$. This actually guarantees foundedness. Given that the unique world view of the top is empty, then the unique world view of the overall program is indeed, according to our method, $W = W_4 = \{\{p, q\}\}$. Notice that, there is still the problem of unfoundedness of world view $\{\{p, q\}\}$ for the program consisting of rules (r1-r3) only. The example suggests that adding an upper-level constraint involving negated subjective literal(s) might be an empirical method to solve this problem.

Let us go back to program Π_0 that was the first one that we mentioned before:

$$\begin{aligned} & a \mid b \quad (r1) \\ & c \leftarrow \mathbf{K}a \quad (r2) \\ & \leftarrow \text{not } c \quad (C) \end{aligned}$$

Here, $B_U(\Pi_0)$ consists of rule (r1), and $T_U(\Pi_0)$ of rule (r2) and constraint (C). So, $T'_U(\Pi_0)$ is (where ka is a fresh atom):

$$\begin{aligned} & ka \mid \text{not } ka \\ & c \leftarrow ka \quad (r2') \\ & \leftarrow \text{not } c \quad (C) \end{aligned}$$

whose unique world view is $\{\{ka, c\}\}$. After cancelling ka , we obtain world view $W_T = \{\{c\}\}$ for $T_U(\Pi_0)$. And, $EC_{T_U(\Pi_0)}(W_T) = \{\mathbf{K}a\}$ with empty RQ . So, given the unique world view $\{\{a\}, \{b\}\}$ of $B_U(\Pi_0)$, its subset $\{\{a\}\}$ fulfils the condition in Definition 5.4 so it will be selected to form the overall candidate world view $W = \{\{a, c\}\}$. As said, W does not satisfy the epistemic splitting property, but in our opinion it captures the ‘intended meaning’ of the program, where the top layer “asks” the bottom layer to support, if possible, $\mathbf{K}A$ (in order not to make the overall program inconsistent).

Let us now consider Π_3 to be the seminal example introduced in [1], that motivated the introduction of ELPs and, later, the introduction of the notion of epistemic splitting. The specific formulation (variations have appeared over time) is the one seen in [26].

$$\begin{aligned}
eligible(X) &\leftarrow high(X) & (r1) \\
eligible(X) &\leftarrow minority(X), fair(X) & (r2) \\
noeligible(X) &\leftarrow not\ fair(X), not\ high(X) & (r3) \\
fair(mike) &| high(mike) & (f1) \\
\\
interview(X) &\leftarrow not\ \mathbf{K}\ eligible(X), not\ \mathbf{K}\ noeligible(X) & (r4) \\
\\
appointment(X) &\leftarrow \mathbf{K}\ interview(X) & (r5)
\end{aligned}$$

Since in this version of the program we have only *mike* as an individual, we may obtain the following ground abbreviated version:

$$\begin{aligned}
e &\leftarrow h & (r1) \\
e &\leftarrow m, f & (r2) \\
ne &\leftarrow not\ f, not\ h & (r3) \\
f &| h & (f1) \\
\\
in &\leftarrow not\ \mathbf{K}e, not\ \mathbf{K}ne & (r4) \\
\\
a &\leftarrow \mathbf{K}in & (r5)
\end{aligned}$$

Here, we consider (r5) as the top $T_U(\Pi_3)$, and (r1-r4) plus (f1) as the bottom, which can be however in turn divided into the top $T_{1U}(\Pi_3)$ including (r4), and the bottom $B_U(\Pi_3)$, composed of (r1-r3) plus (f1). So, $T'_U(\Pi_3)$ is (where *kin* is a fresh atom):

$$\begin{aligned}
kin &| not\ kin \\
a &\leftarrow kin & (r5')
\end{aligned}$$

with world view $\{\{a, kin\}, \emptyset\}$. After cancelling *kin* and the empty set, we obtain for $T_U(\Pi_3)$ world view $W_{11} = \{\{a\}\}$ with $RQ_{T_U(\Pi_3)}(W_{11}) = \{\mathbf{K}in\}$. Set *EC* is empty.

Then, $T'_{1U}(\Pi_3)$ is (where *ke* and *kne* are fresh atoms),

$$\begin{aligned}
ke &| not\ ke \\
kne &| not\ kne \\
in &\leftarrow not\ ke, not\ kne & (r4')
\end{aligned}$$

So, $T_{1U}(\Pi_3)$ has the unique world view (after cancelling fresh atoms and empty sets) $W_2 = \{\{in\}\}$. Here, both sets *EC* and *RQ* are empty. Finally, $B_U(\Pi_3)$ is

$$\begin{aligned}
e &\leftarrow h & (r1) \\
e &\leftarrow m, f & (r2) \\
ne &\leftarrow not\ f, not\ h & (r3) \\
f &| h & (f1)
\end{aligned}$$

with world view $W_3 = \{\{h, e\}, \{f\}\}$. Since no constraints or requirements are given, we can obtain a Candidate World View $W^{\Pi_3} = \{\{h, e, in, a\}, \{f, in, a\}\}$ for the part of the program

including (r1-r4) plus (f1) by performing the union $\{h, e\} \cup W_2 = \{h, e, in\}$ and $\{f\} \cup W_2 = \{f, in\}$, since both sets are compliant with $RQ_{TV(\Pi_3)}(W_{11}) = \{\mathbf{K}in\}$. It is easily seen that W^{Π_3} is the unique the world view of the overall program.

Notice that the above program is *Epistemically Stratified* in the sense of [25, 26], according to which a program is epistemically stratified if there exists a mapping of atoms to levels, where: all the objective atoms occurring in a rule are at the same level; instead, atoms occurring in subjective literals in the body of rules are at a strictly lower level. They prove that, for any semantics obeying epistemic splitting, an epistemically stratified program has a unique world view. Actually, according to the result stated in [36], epistemically stratified programs have a unique world view under any semantics, as it is shown there that multiple world views can arise only in consequence of negative cycles involving epistemic literals: these cycles are obviously impossible for epistemically stratified programs. In [25, 26] it is shown how to compute the unique world view bottom-up. We have just seen how to compute it top-down. So, on this class of programs the two methods coincide.

6. Discussion

It is at this point interesting to try to assess formally which semantics (if any) satisfy top-down epistemic splitting.

Observation. We can see that the “generate and test” style of programming which is commonly used in traditional ASP, where the bottom part of the program generates a search space and constraints in the top prune it, does not immediately generalize to epistemic logic programming. Subjective constraints in fact (or, more generally, constraints involving directly or indirectly some subjective literal) do indeed reduce the number of world views of the overall program, w.r.t. the number of world views of the bottom part. But, they concur to determine the contents of the remaining world views. So, we might in perspective define a notion of *Epistemic Subjective Constraint Monotonicity*, different however from the one of [18].

For testing compliance with the Top-down Epistemic Splitting Property TDESP, we examine the case of the semantics introduced in [24], that, following [27], we call K15 for short.

Definition 6.1 (K15-world views, as reported in [27]). *Given a logic program Π , its K15-reduct with respect to a non-empty set of interpretations W is obtained by:*

1. replacing by \perp every subjective literal $L \in \text{Body}_{sub}(r)$ such that $W \not\models L$, and
2. replacing all other occurrences of subjective literals of the form $\mathbf{K}A$ by A .

A non-empty set of interpretations W is a K15-world view of Π iff W is the set of all stable models of the K15-reduct of Π with respect to W .

In [27] it is in fact noticed that K15 slightly generalizes the semantics proposed in [12] (called G11 for short) and can be seen as a basis for the semantics of [11] (called S16 for short). In particular, S16 treats K15 world views as candidate solutions, to be pruned in a second step, to allow some unwanted world view to be removed; this because S16 considers the operator **not** A which means *not* KA , so they need to maximize what is not known. Thus, should K15 satisfy top-down Epistemic Splitting, also G11 and S16 would do as well.

Theorem 6.1 (K15 TDESP). *The K15 semantics satisfies the Top-down Epistemic Splitting Property. I.e., given an ELP Π , and set of sets W , where each set is composed of atoms occurring in Π , W is a K15 world view for Π if and only if it is a Candidate world view for Π according to Definition 5.4.*

Proof

Assume that there are two layers, top $T_U(\Pi)$ and bottom $B_U(\Pi)$. The reasoning below can be iterated over a subdivision into an arbitrary number of levels. Notice that, given a K15 world view W , since each atom A that occurs in the sets composing W is derived in the part of the program including rules with head A , then W can be divided into two parts, W_T which is a world view of the top $T_U(\Pi)$ and W_B which is a world view of the bottom $B_U(\Pi)$, each one composed of stable models of the K15-reduct of that part of the program.

If part. Given a K15 world view W , let Sl^T be the subjective literals occurring in $T_U(\Pi)$ for which $W_B \models KA$, i.e., which are entailed by the bottom. So, the subset of Sl^T that consists of literals involved in constraints in $T_U(\Pi)$ will form set $EC_{T_U(\Pi)}(W_T)$, and the remaining ones will form set $RQ_{T_U(\Pi)}(W_T)$. Therefore, we can conclude that W , which is a K15 world view, is indeed a Candidate World View according to Definition 5.4.

Only if part. Consider a Candidate World View W w.r.t. the K15 semantics, obtained by combining a subset W_B of a K15 world view of $B_U(\Pi)$ with a K15 world view W_T of $T_U(\Pi)$ (see below for Top-down Influence). According to Definition 5.4, the combination is possible only if for each epistemic literal $\mathbf{KA} \in ECRQ_{T_U(\Pi)}(W_T)$, $W_B \models \mathbf{KA}$. If $\mathbf{KA} \in EC_{T_U(\Pi)}(W_T)$, if this is not the case then there would be a constraint violation in $T_U(\Pi)$, so there would be no world views for $T_U(\Pi)$, and for the overall program Π . Considering $\mathbf{KA} \in RQ_{T_U(\Pi)}(W_T)$, if it were not that $W_B \models \mathbf{KA}$, then by definition of K15 \mathbf{KA} would have been substituted by \perp instead of by A , so W_T would have been a different set. The Top-down Influence step can be disregarded, since it performs in advance on elements of $ECRQ_{T_U(\Pi)}(W_T)$, that are required to be entailed by W_B anyway, the same transformation performed by K15, step 2. Then, a Candidate World view W obtained according to Definition 5.4 is indeed a K15 world view.

7. Conclusions

In this paper, we have discussed properties of semantics of ELPs. We explored a similar though complementary approach w.r.t. the work of Cabalar et al., starting from the concept, that they propose, of epistemic splitting of an ELP. In particular, we defined the Top-down Epistemic Splitting Property. We proved that the K15 semantics satisfies this property, and in consequence so do G11 and S16. An investigation of which other semantics might satisfy this property is a subject of future work. A question that may arise concerns efficiency of computing world views in a top-down fashion. We believe that, if the subjective literals connecting adjacent layers are in small number (as it seems reasonable), then efficiency might not be a concern.

In [25] Section 6, it is argued that, with epistemic splitting and answer sets computation in the bottom-up fashion, a problem of conformant planning can be expressed in a way which is ‘more natural’ than under other semantics. It remains to be seen in which kinds of applications the different approaches (top-down and bottom-up) might be profitably exploited.

References

- [1] M. Gelfond, H. Przymusińska, Definitions in epistemic specifications, in: A. Nerode, V. W. Marek, V. S. Subrahmanian (Eds.), *Logic Programming and Non-monotonic Reasoning, Proceedings of the First Intl. Workshop*, 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] V. W. Marek, M. Truszczyński, *Stable logic programming - an alternative logic programming paradigm*, Springer, 1999, pp. 375–398.
- [4] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, K. Bowen (Eds.), *Proceedings of the 5th Intl. Conf. and Symposium on Logic Programming*, MIT Press, 1988, pp. 1070–1080.
- [5] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Computing* 9 (1991) 365–385.
- [6] G. Brewka, T. Eiter, M. T. (eds.), *Answer set programming: Special issue*, *AI Magazine* 37 (2016).
- [7] C. Baral, *Knowledge representation, reasoning and declarative problem solving*, Cambridge University Press, 2003.
- [8] M. Truszczyński, Logic programming for knowledge representation, in: V. Dahl, I. Niemelä (Eds.), *Logic Programming, 23rd Intl. Conf., ICLP 2007*, 2007, pp. 76–88.
- [9] M. Gelfond, Answer sets, in: *Handbook of Knowledge Representation. Chapter 7*, Elsevier, 2007.
- [10] M. Gebser, N. Leone, M. Maratea, S. Perri, F. Ricca, T. Schaub, Evaluation techniques and systems for answer set programming: a survey, in: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, International Joint Conferences on Artificial Intelligence Organization, 2018*, pp. 5450–5456. URL: <https://doi.org/10.24963/ijcai.2018/769>. doi:10.24963/ijcai.2018/769.
- [11] Y. Shen, T. Eiter, Evaluating epistemic negation in answer set programming, *Artificial Intelligence* 237 (2016) 115–135.
- [12] M. Gelfond, New semantics for epistemic specifications, in: J. P. Delgrande, W. Faber (Eds.), *Logic Programming and Nonmonotonic Reasoning - 11th Intl. Conf., LPNMR 2011, Proceedings*, volume 6645 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 260–265.
- [13] M. Truszczyński, Revisiting epistemic specifications, in: M. Balduccini, T. C. Son (Eds.), *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 315–333.
- [14] L. Fariñas del Cerro, A. Herzig, E. I. Su, Epistemic equilibrium logic, in: Q. Yang, M. Wooldridge (Eds.), *Proceedings of the Twenty-Fourth Intl. Joint Conf. on Artificial Intelligence, IJCAI 2015, AAAI Press, 2015*, pp. 2964–270.
- [15] E. I. Su, A monotonic view on reflexive autoepistemic reasoning, in: M. Balduccini, T. Janhunen (Eds.), *Logic Programming and Nonmonotonic Reasoning - 14th Intl. Conf., LPNMR 2017, Proceedings*, volume 10377 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 85–100.

- [16] P. T. Kahl, A. P. Leclerc, Epistemic logic programs with world view constraints, in: A. D. Palù, P. Tarau, N. Saeedloei, P. Fodor (Eds.), Technical Communications of the 34th Intl. Conf. on Logic Programming, ICLP 2018, volume 64 of *OASICS*, Schloss Dagstuhl, 2018, pp. 1:1–1:17.
- [17] E. I. Su, Epistemic answer set programming, in: F. Calimeri, N. Leone, M. Manna (Eds.), Logics in Artificial Intelligence - 16th European Conf., JELIA 2019, Proceedings, volume 11468 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 608–626.
- [18] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, Founded world views with autoepistemic equilibrium logic, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), Logic Programming and Nonmonotonic Reasoning - 15th Intl. Conf., LPNMR 2019, Proceedings, volume 11481 of *Lecture Notes in Computer Science*, 2019, pp. 134–147.
- [19] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, Splitting epistemic logic programs, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), Logic Programming and Nonmonotonic Reasoning - 15th Intl. Conf., LPNMR 2019, Proceedings, volume 11481 of *Lecture Notes in Computer Science*, 2019, pp. 120–133.
- [20] T. C. Son, T. Le, P. T. Kahl, A. P. Leclerc, On computing world views of epistemic logic programs, in: C. Sierra (Ed.), Proc. of the Twenty-Sixth Intl. Joint Conf. on Artificial Intelligence, IJCAI 2017, ijcai.org, 2017, pp. 1269–1275.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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. doi:10.1093/logcom/exv065.
- [25] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, Splitting epistemic logic programs, in: M. Balduccini, Y. Lierler, S. Woltran (Eds.), Logic Programming and Nonmonotonic Reasoning - 15th International Conference, LPNMR 2019, Philadelphia, PA, USA, June 3-7, 2019, Proceedings, volume 11481 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 120–133.
- [26] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, On the splitting property for epistemic logic programs (extended abstract), in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, ijcai.org, 2020, pp. 4721–4725.
- [27] P. Cabalar, J. Fandinno, L. Fariñas del Cerro, Autoepistemic answer set programming, *Artif. Intell.* 289 (2020) 103382.
- [28] V. Lifschitz, H. Turner, Splitting a logic program, in: Proc. of ICLP’94, Intl. Conf. on Logic Programming, 1994, pp. 23–37.
- [29] K. Marple, G. Gupta, Dynamic consistency checking in goal-directed answer set programming, *TPLP* 14 (2014) 415–427.

- [30] S. Costantini, A. Formisano, Query answering in resource-based answer set semantics, *Theory and Practice of Logic Programming* 16 (2016) 619–635.
- [31] Y. Lierler, M. Truszczynski, On abstract modular inference systems and solvers, *Artif. Intell.* 236 (2016) 65–89. doi:10.1016/j.artint.2016.03.004.
- [32] B. Bogaerts, A. Weinzierl, Exploiting justifications for lazy grounding of answer set programs, in: J. Lang (Ed.), *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, ijcai.org, 2018, pp. 1737–1745. doi:10.24963/ijcai.2018/240.
- [33] J. W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1987.
- [34] V. Lifschitz, Thirteen definitions of a stable model, in: A. Blass, N. Dershowitz, W. Reisig (Eds.), *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*, volume 6300 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 488–503.
- [35] S. Costantini, A. Formisano, Negation as a resource: a novel view on answer set semantics, *Fundamenta Informaticae* 140 (2015) 279–305.
- [36] S. Costantini, About epistemic negation and world views in epistemic logic programs, *Theory Pract. Log. Program.* 19 (2019) 790–807. doi:10.1017/S147106841900019X.
- [37] S. Costantini, Meta-reasoning: A survey, in: A. C. Kakas, F. Sadri (Eds.), *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 253–288.
- [38] J. Dix, A classification theory of semantics of normal logic programs I-II., *Fundamenta Informaticae* 22 (1995) 227–255 and 257–288.
- [39] Y. Shen, T. Eiter, Constraint monotonicity, epistemic splitting and foundedness are too strong in answer set programming, *CoRR* abs/2010.00191 (2020). URL: <https://arxiv.org/abs/2010.00191>.