Open Archive Toulouse Archive Ouverte

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible

This is an author's version published in:
http://oatao.univ-toulouse.fr/24815

Official URL

DOI : https://doi.org/10.1007/978-3-030-20528-7_10

To cite this version: Calabar, Pedro and Fandinno, Jorge and Fariñas del Cerro, Luis *Splitting Epistemic Logic Programs.* (2019) In: International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2019), 3 June 2019 - 7 June 2019 (Philadelphia, PA, United States).

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

Splitting Epistemic Logic Programs

Pedro Cabalar[1(✉)], Jorge Fandinno[2], and Luis Fariñas del Cerro[2]

[1] University of Corunna, Corunna, Spain
cabalar@udc.es
[2] IRIT, University of Toulouse, CNRS, Toulouse, France
{jorge.fandinno,farinas}@irit.fr

Abstract. Epistemic logic programs constitute an extension of the stable models semantics to deal with new constructs called *subjective literals*. Informally speaking, a subjective literal allows checking whether some regular literal is true in all or some stable models. As it can be imagined, the associated semantics has proved to be non-trivial, as the truth of subjective literals may interfere with the set of stable models it is supposed to query. As a consequence, no clear agreement has been reached and different semantic proposals have been made in the literature. Unfortunately, comparison among these proposals has been limited to a study of their effect on individual examples, rather than identifying general properties to be checked. In this paper, we propose an extension of the well-known splitting property for logic programs to the epistemic case. We formally define when an arbitrary semantics satisfies the *epistemic splitting property* and examine some of the consequences that can be derived from that, including its relation to conformant planning and to epistemic constraints. Interestingly, we prove (through counterexamples) that most of the existing proposals fail to fulfill the epistemic splitting property, except the original semantics proposed by Gelfond in 1991.

1 Introduction

The language of *epistemic specifications*, proposed by Gelfond in 1991 [3], constituted an extension of disjunctive logic programming that introduced modal operators to quantify over the set of stable models [5] of a program. These new constructs were later incorporated as an extension of the Answer Set Programming (ASP) paradigm in different implemented solvers (see [8] for a recent survey). The new constructs, *subjective literals*, have the form $\mathbf{K}\,l$ and $\mathbf{M}\,l$ and allow respectively checking whether regular literal l is true in every stable model (cautious consequence) or in some stable model (brave consequence). In many cases, these subjective literals can be seen as simple queries, but what makes

A preliminary version of this work was presented at [1]. Partially supported by MINECO, Spain, grant TIC2017-84453-P, Xunta de Galicia, Spain (GPC ED431B 2016/035 and 2016-2019 ED431G/01, CITIC). The second author is funded by the Centre International de Mathématiques et d'Informatique de Toulouse (CIMI) through contract ANR-11-LABEX-0040-CIMI within the program ANR-11-IDEX-0002-02.

https://doi.org/10.1007/978-3-030-20528-7_10

them really interesting is their use in rule bodies, which may obviously affect the set of stable models they are meant to quantify. This feature makes them suitable for modelling introspection but, at the same time, easily involves cyclic specifications whose intuitive behaviour is not always easy to define. For instance, the semantics of an epistemic logic program may yield alternative sets of stable models, each set being called a *world view*. Deciding the intuitive world views of a cyclic specification has motivated a wide debate in the literature. In fact, in Gelfond's original semantics (G91) [3] or in its extension [12], some cyclic examples manifested self-supportedness, so in [4] Gelfond himself and, later on, other authors [2,6,10,11] proposed different variants trying to avoid unintended results. Unfortunately, comparison among these variants was limited to studying their effect on a set of "test" examples, leading to a lack of confidence as any proposal is always subject to the appearance of new counterintuitive examples. A next methodological step would consist in defining formal properties to be established and that would cover complete families of examples and, hopefully, could help to reach an agreement on some language fragments. For instance, one would expect that, at least, the existing approaches agreed on their interpretation of acyclic specifications. Regretfully, as we will see, this is not the case.

In this paper we propose a candidate property, we call *epistemic splitting*, that not only defines an intuitive behaviour for stratified epistemic specifications but also goes further, extending the splitting theorem [9], well-known for standard logic programs, to the epistemic case. Informally speaking, we say that an epistemic logic program can be split if a part of the program (the *top*) only refers to the atoms of the other part (the *bottom*) through subjective literals. A given semantics satisfies epistemic splitting if, given any split program, it is possible to get its world views by first obtaining the world views of the bottom and then using the subjective literals in the top as "queries" on the bottom part previously obtained. If epistemic splitting holds, the semantics immediately satisfies other properties. For instance, if the use of epistemic operators is stratified, the program has a unique world view at most. Similarly, epistemic constraints (those only consisting of subjective literals) can be guaranteed to only rule out candidate world views. As we will see, however, only the G91 semantics satisfies epistemic splitting among the previously cited approaches. So, somehow, the recent attempts to fix the behaviour of cycles has neglected the attention on the effects produced on acyclic specifications. In fact, a different property of epistemic splitting was already proved in [13] as a method to compute world views for this semantics. However, this definition is based on a "safety" condition that needs to be checked for all possible world views and is specific for G91 semantics, so it is harder to justify as a general property required for other approaches.

The rest of the paper is organised as follows. First, we motivate the main idea through a well-known example. After that, we recall basic definitions of (non-epistemic) ASP and splitting, introduce the language of epistemic specifications and define the G91 semantics. In the next section, we proceed to define the property of epistemic splitting and study some of its consequences. Then, we formally prove that G91 satisfies this property while we provide counterexamples for the other approaches, concluding the paper after that.

2 Motivation

To illustrate the intuition behind our proposal, let us consider the well-known standard example introduced in [3].

Example 1. A given college uses the following set of rules to decide whether a student X is eligible for a scholarship:

$$eligible(X) \leftarrow high(X) \tag{1}$$

$$eligible(X) \leftarrow minority(X), fair(X) \tag{2}$$

$$\sim eligible(X) \leftarrow \sim fair(X), \sim high(X) \tag{3}$$

Here, '$\sim$' stands for strong negation and $high(X)$ and $fair(X)$ refer to the grades of student X. We want to encode the additional college criterion "*The students whose eligibility is not determined by the college rules should be interviewed by the scholarship committee*" as another rule in the program. □

The problem here is that, for deciding whether $eligible(X)$ "*can be determined*," we need to check if it holds in all the answer sets of the program, that is, if it is one of the cautious consequences of the latter. For instance, if the only available information for some student *mike* is the disjunction

$$fair(mike) \vee high(mike) \tag{4}$$

we get that program {(1)–(4)} has two stable models, {$high(mike)$, $eligible(mike)$} and {$fair(mike)$} so $eligible(mike)$ cannot be determined and an interview should follow. Of course, if we just want to query cautious and brave consequences of the program, we can do it inside ASP. For instance, the addition of constraint:

$$\bot \leftarrow eligible(mike)$$

allows us to decide if $eligible(mike)$ is a cautious consequence by just checking that the resulting program has no answer sets. The difficulty comes from the need to *derive* new information from a cautious consequence. This is where subjective literals come into play. Rule

$$interview(X) \leftarrow \texttt{not}\, \mathbf{K}\, eligible(X), \texttt{not}\, \mathbf{K} \sim eligible(X) \tag{5}$$

allows us to prove that $interview(X)$ holds whenever neither $eligible(X)$ nor $\sim eligible(X)$ are cautious consequences of {(1)–(4)}. Recall that $\mathbf{K}l$ holds when the literal l is true in all stable models of the program. The novel feature here is that (5) is also part of the program, and so, it affects the answer sets queried by $\mathbf{K}$ too, which would actually be:

$$\{fair(mike), interview(mike)\} \tag{6}$$

$$\{high(mike), eligible(mike), interview(mike)\} \tag{7}$$

So, there is a kind of cyclic reasoning: operators **K** and **M** are used to query a set of stable models that, in their turn, may depend on the application of that query. In the general case, this kind of cyclic reasoning is solved by resorting to multiple world views, but in our particular example, however, this does not seem to be needed. One would expect that separating the queried part {(1)–(4)} and the rule that makes the query (5) should be correct, since the first four rules do not depend on (5) and the latter exclusively consults them without interacting with their results. This same reasoning could be applied if we added one more level such as, for instance, by including the rule:

$$appointment(X) \leftarrow \mathbf{K}\, interview(X) \tag{8}$$

The two answer sets of program {(1)–(5)} contain $interview(mike)$ and so $appointment(mike)$ can be added to both answer sets incrementally. This method of analysing a program by division into independent parts shows a strong resemblance to the *splitting theorem* [9], well-known in standard ASP. Splitting is applicable when the program can be divided into two parts, the *bottom* and the *top*, in such a way that the bottom never refers to head atoms in the top. When this happens, we can first compute the stable models of the bottom and then, for each one, simplify the top accordingly, getting new stable models that complete the information. We could think about different ways of extending this method for the case of epistemic logic programs, depending on how restrictive we want to be on the programs where it will be applicable. However, we will choose a very conservative case, looking for a wider agreement on the proposed behaviour. The condition we will impose is that our top program can only refer to atoms in the bottom through epistemic operators. In this way, the top is seen as a set of rules that derive facts from epistemic queries on the bottom. Thus, each world view W of the bottom will be used to replace the subjective literals in the top by their truth value with respect to W. For the sake of completeness, we recall next the basic definitions of ASP and splitting, to proceed with a formalization of epistemic splitting afterwards.

3 Background of ASP and Epistemic Specifications

Given a set of atoms At, a *regular literal* is either an atom or a truth constant[1], that is $a \in At \cup \{\top, \bot\}$, or its default negation, `not` a. A *rule* r is an implication of the form:

$$a_1 \vee \cdots \vee a_n \leftarrow L_1, \ldots, L_m \tag{9}$$

with $n \geq 0$ and $m \geq 0$, where each $a_i \in At$ is an atom and each L_j a regular literal. The left hand disjunction of (9) is called the rule *head* and abbreviated as $Head(r)$. When $n = 0$, it corresponds to $\bot$ and r is called a *constraint*. The right hand side of (9) is called the rule *body* and abbreviated as $Body(r)$. When

[1] For a simpler description of program transformations, we allow truth constants with their usual meaning.

$m = 0$, the body corresponds to $\top$ and r is called a *fact* (in this case, the body and the arrow symbol are usually omitted). A *(regular) program* Π is a (possibly infinite) set of rules. We write $Atoms(F)$ to represent the set of atoms occurring in any syntactic construct F (a literal, head, body, rule or program). A propositional interpretation I is a set of atoms. We assume that strong negation '$\sim a$' is just another atom in At and that the constraint $\bot \leftarrow a, \sim a$ is implicitly included in the program. We allow the use of variables, but understood as abbreviations of their possible ground instances. Given any syntactic construct F, we write $I \models F$ to stand for "I satisfies F" in classical propositional logic, where the commas correspond to conjunctions, '`not`' corresponds (under this interpretation) to classical negation and '$\leftarrow$' is just a reversed material implication. An interpretation I is a *(classical) model* of a program Π if it satisfies all its rules. The *reduct* of a program Π with respect to some propositional interpretation I, in symbols Π^I, is obtained by replacing in Π every negative literal `not` a by $\top$ if $I \models$ `not` a or by $\bot$ otherwise. A propositional interpretation I is a *stable model* of a program Π iff it is a $\subseteq$-minimal model of Π^I. By SM$[\Pi]$, we denote the set of all stable models of Π. The following is a well-known property in ASP.

Property 1 (Supraclassicality). Any stable model of a (regular) program Π is also a classical model of Π.

We extend now the syntax of ASP to the language of epistemic specifications. Given a set of atoms At, we call *subjective literal* to any expression of the form $\mathbf{K}\, l$, $\mathbf{M}\, l$, `not` $\mathbf{K}\, l$ or `not` $\mathbf{M}\, l$, for any regular literal l. We keep the same syntax for rules as in (9) excepting that body literals L_j can also be subjective literals now. Given rule r we define the sets $Body^{reg}(r)$ and $Body^{sub}(r)$ respectively containing the regular and the subjective literals in $Body(r)$. Rules or programs are *regular* if they do not contain subjective literals. We say that a rule is a *subjective constraint* if it is a constraint, $Head(r) = \bot$, and its body exclusively consists of subjective literals, that is $Body(r) = Body^{sub}(r)$.

We can define the concept of *model* of a program, in a similar way as we did for classical models in regular ASP. A *modal interpretation* $\mathcal{M} = \langle W, I\rangle$ is pair where I is a propositional interpretation and $W \subseteq 2^{At}$ is a non-empty set of propositional interpretations. A modal interpretation $\mathcal{M} = \langle W, I\rangle$ *satisfies* a literal L, written $\langle W, I\rangle \models L$, if

1. $\langle W, I\rangle \models \top$,
2. $\langle W, I\rangle \not\models \bot$,
3. $\langle W, I\rangle \models a$ if $a \in I$, for any atom $a \in At$,
4. $\langle W, I\rangle \models \mathbf{K}\, l$ if $\langle W, I'\rangle \models l$ for all $I' \in W$,
5. $\langle W, I\rangle \models \mathbf{M}\, l$ if $\langle W, I'\rangle \models l$ for some $I' \in W$, and
6. $\langle W, I\rangle \models$ `not` L if $\langle W, I\rangle \not\models L$.

Since for a subjective literal L, $\langle W, I\rangle \models L$ does not depend on I, we sometimes write $W \models L$. For a rule r of the form (9), we write $\langle W, I\rangle \models r$ iff either $\langle W, I\rangle \models a_i$ for some $1 \leq i \leq n$ or $\langle W, I\rangle \not\models L_j$ for some $1 \leq j \leq m$. We say that $\langle W, I\rangle$ is a *model* of a program Π, written $\langle W, I\rangle \models \Pi$, if it satisfies all

its rules. Among the possible models of an epistemic logic program, all semantic approaches agree on selecting some preferred models called *world views*, each one being characterized by the W component. These world views satisfy a similar property to that of supraclassicality (Property 1) in non-epistemic ASP. In this case, however, rather than talking about classical models, we resort to modal logic S5, so all world views of a program are also S5 models of the program. This property can be formally stated as follows:

Property 2 (Supra-S5). A semantics satisfies *supra-S5* when for every world view W of an epistemic program Π and for every $I \in W$, $\langle W, I\rangle \models \Pi$. □

To the best of our knowledge, all existing semantics satisfy supra-S5. Another property that is shared by all semantics is that, when Π is a regular ASP program (it has no modal epistemic operators) then it has a unique world view containing all the stable models of Π. We will formalize this property in the following way.

Property 3 (Supra-ASP). A semantics satisfies *supra-ASP* if for any regular program Π either Π has a unique world view $W = \mathrm{SM}[\Pi] \neq \emptyset$ or $\mathrm{SM}[\Pi] = \emptyset$ and Π has no world view at all. □

Originally, some semantics like [3] or [12], allowed empty world views $W = \emptyset$ when the program has no stable models, rather than leaving the program without world views. Since this feature is not really essential, we exclusively refer to non-empty world views in this paper.

We define next a useful transformation extending the idea of the reduct to epistemic specifications, and generalized for a given signature.

Definition 1 (Subjective reduct). *The* subjective reduct *of a program Π with respect to a set of propositional interpretations W and a signature $U \subseteq$ At, also written Π_U^W, is obtained by replacing each subjective literal L with $Atoms(L) \subseteq U$ by; $\top$ if $W \models L$ or by $\bot$ otherwise. When $U =$ At we just write Π^W.* □

We use the same notation Π^W as for the standard reduct, but ambiguity is removed by the type of W (a set of interpretations now). This subjective reduct can be used to define [3] (G91) semantics in the following way.

Definition 2 (G91-world view). *A non-empty set of interpretations W is a* G91-world view *of an epistemic program Π if $W = \mathrm{SM}[\Pi^W]$.* □

We will not provide the formal definitions of the rest of semantics compared in this paper, since none of them satisfies our goal property of epistemic splitting. In those cases, it will suffice with providing counterexamples and the reader can check their behaviour by resorting to the corresponding original definition.

4 Epistemic Splitting

We proceed now to introduce our definition of the epistemic splitting property. To do so, we begin extending the idea of splitting set from [9]. For space reasons, we refer the reader to [9] for the formal definition of splitting set.

Definition 3 (Epistemic splitting set). *A set of atoms* $U \subseteq$ At *is said to be an* epistemic splitting set *of a program* Π *if for any rule* r *in* Π *one of the following conditions hold*

(i) $Atoms(r) \subseteq U$,
(ii) $(Atoms(Body^{reg}(r) \cup Head(r))) \cap U = \emptyset$

We define an splitting *of* Π *as a pair* $\langle B_U(\Pi), T_U(\Pi)\rangle$ *satisfying* $B_U(\Pi) \cap T_U(\Pi) = \emptyset$, $B_U(\Pi) \cup T_U(\Pi) = \Pi$, *all rules in* $B_U(\Pi)$ *satisfy (i) and all rules in* $T_U(\Pi)$ *satisfy (ii).* □

With respect to the original definition of splitting set, we have replaced the condition for the top program, $Atoms(Head(r)) \cap U = \emptyset$, by the new condition (ii), which in other words means that the top program may only refer to atoms U in the bottom through epistemic operators. Note that this introduces a new kind of "dependence," so that, as happens with head atoms, regular literals in the body also depend on atoms in subjective literals. For instance, if $U = \{p, q\}$, the program $\Pi_1 = \{p \vee q \ , \ s \leftarrow p, \mathbf{K}\, q\}$ would not be splittable due to the second rule, since $s \notin U$ and we would also need the regular literal $p \notin U$. The reason for this restriction is to avoid imposing (to a potential semantics) a fixed way of evaluating p with respect to the world view $[\{p\}, \{q\}]$ for the bottom.

Another observation is that we have kept the definition of $B_U(\Pi)$ and $T_U(\Pi)$ non-deterministic, in the sense that some rules can be arbitrarily included in one set or the other. These rules correspond to subjective constraints on atoms in U, since these are the only cases that may satisfy conditions (i) and (ii) simultaneously.

If we retake our example program $\Pi_2 = \{(1)\text{–}(5)\}$, we can see that the set U consisting of atoms $high(mike), fair(mike)$, $eligible(mike), minority(mike)$ and their corresponding strong negations is an epistemic splitting set that divides the program into the bottom $B_U(\Pi_2) = \{(1)\text{–}(4)\}$ and the top $T_U(\Pi_2) = \{(5)\}$. As in regular splitting, the idea is computing first the world views of the bottom program $B_U(\Pi)$ and for each one, W, simplifying the corresponding subjective literals in the top program. Given an epistemic splitting set U for a program Π and set of interpretations W, we define $E_U(\Pi, W) \stackrel{\text{def}}{=} T_U(\Pi)^W_U$, that is, we make the subjective reduct of the top with respect to W and signature U. A pair $\langle W_b, W_t\rangle$ is said to be a *solution* of Π with respect to an epistemic splitting set U if W_b is a world view of $B_U(\Pi)$ and W_t is a world view of $E_U(\Pi, W_b)$. Notice that this definition is semantics-dependent in the sense that each alternative semantics for epistemic specifications will define its own solutions for a given U and Π, since it defines the selected world views for a program in a different way. Back to our example, notice that $B_U(\Pi_2)$ is a regular program without epistemic operators. Thus, any semantics satisfying supra-ASP will provide $W_b = [\{fair(mike)\}, \{high(mike), eligible(mike)\}]$ as the unique world view for the bottom. The corresponding simplification of the top would be $E_U(\Pi_2, W_b)$ containing (after grounding) the single rule $interview(mike) \leftarrow \texttt{not}\ \bot, \texttt{not}\ \bot$ Again, this program is regular and its unique world view would be $W_t = [\{interview(mike)\}]$. Now, in the general case, to reconstruct the world views for the global program we define the operation:

$$W_b \sqcup W_t \quad = \quad \{\ I_b \cup I_t \mid I_b \in W_b \text{ and } I_t \in W_t\ \}$$

(remember that both the bottom and the top may produce multiple world views, depending on the program and the semantics we choose). In our example, $W_b \sqcup W_t$ would exactly contain the two stable models (6) and (7) we saw in the introduction.

Property 4 (Epistemic splitting). A semantics satisfies *epistemic splitting* if for any epistemic splitting set U of any program Π: W is a world view of Π iff there is a solution $\langle W_b, W_t\rangle$ of Π with respect to U such that $W = W_b \sqcup W_t$. □

In the example, this means that the world view we obtained in two steps is indeed the unique world view of the whole program, under any semantics satisfying epistemic splitting. Uniqueness of world view was obtained in this case because both the bottom program $B_U(\Pi_2)$ and the top, after simplification, $E_U(\Pi_2, W_b)$ were regular programs and we assumed supra-ASP. In fact, as we see next, we can still get a unique world view (at most) when there are no cyclic dependences among subjective literals. This mimics the well-known result for *stratified negation* in logic programming. Let us define a modal dependence relation among atoms in a program Π so that $dep(a, b)$ is true iff there is a rule $r \in \Pi$ such that $a \in Atoms(Head(r) \cup Body^{reg}(r))$ and $b \in Atoms(Body^{sub}(r))$.

Definition 4. *We say that an epistemic program Π is* epistemically stratified *if we can assign an integer mapping $\lambda : \mathrm{At} \to \mathbb{N}$ to each atom such that $\lambda(a) > \lambda(b)$ for any pair of atoms a, b satisfying $dep(a, b)$.* □

Take, for instance, the extended program $\Pi_3 = \{(1)\text{–}(5), (8)\}$. We can assign atoms $high(mike)$, $fair(mike)$, $minority(mike)$ and $eligible(mike)$ layer 0. Then $interview(mike)$ could be assigned layer 1 and, finally, $appointment(mike)$ can be located at layer 2. So, Π_3 is epistemically stratified.

Theorem 1. *Let Π be a finite, epistemically stratified program. Then, any semantics satisfying supra-ASP and epistemic splitting assigns, at most, a unique world view to Π.* □

The proof of the theorem just relies on multiple applications of splitting to each layer backwards and the fact that each simplification $E_U(\Pi, W_b)$ will be a regular program. This is very easy to see in the extended example Π_3. We can split the program using as U all atoms but $appointment(mike)$ to get a bottom Π_2 and a top $\{(8)\}$. Program Π_2 can be split in its turn as we saw before, producing the unique world view [(6), (7)]. Then $E_U(\Pi_3, \{(6), (7)\})$ contains the single rule $appointment(mike) \leftarrow \top$ that is a regular program whose unique world view is $[\{appointment(mike)\}]$ and, finally, the combination of these two world views yields again a unique world view $[(6) \cup\{appointment(mike)\}, (7) \cup\{appointment(mike)\}]$.

Another consequence of epistemic splitting is that subjective constraints will have a monotonic behaviour. Note first that, for a subjective constraint r, we can abbreviate $\langle W, I\rangle \models r$ as $W \models r$ since the I component is irrelevant. Additionally, $W \models r$ means that $Body(r) = Body^{sub}(r)$ is falsified, since $Head(r) = \bot$.

Property 5 (subjective constraint monotonicity). A semantics satisfies *subjective constraint monotonicity* if, 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 $W \models r$. □

Theorem 2. *Epistemic splitting implies subjective constraint monotonicity.* □

To conclude the exploration of consequences of epistemic splitting, let us consider a possible application to conformant planning. To this aim, consider the following simple example.

Example 2. To turn on the light in a room, we can toggle one of two lamps l_1 or l_2. In the initial state, lamp l_1 is plugged but we ignore the state of l_2. Our goal is finding a plan that guarantees we get light in the room in one step.

A logic program that encodes this scenario for a single transition[2] could be Π_4:

$$\begin{array}{cc} plugged(l_1) & light \leftarrow toggle(L), plugged(L) \\ plugged(l_2) \vee {\sim}plugged(l_2) & \bot \leftarrow toggle(l_1), toggle(l_2) \end{array}$$

for $L \in \{l_1, l_2\}$. As we can see, $toggle(l_1)$ would constitute a conformant plan, since we obtain *light* regardless of the initial state, while this does not happen with plan $toggle(l_2)$. In order to check whether a given sequence of actions $A_0, \ldots, A_n$ is a valid conformant plan one would expect that, if we added those facts to the program, a subjective constraint should be sufficient to check that the goal holds in all the possible outcomes. In our example, we would just use:

$$\bot \leftarrow \texttt{not}\ \mathbf{K}\, light \tag{10}$$

and check that the program $\Pi_4 \cup \{toggle(L)\} \cup \{(10)\}$ has some world view, varying $L \in \{l_1, l_2\}$. Subjective constraint monotonicity guarantees that the addition of this "straighforward" formalisation has the expected meaning.

This method would only allow testing if the sequence of actions constitutes a conformant plan, but does not allow generating those actions. A desirable feature would be the possibility of applying the well-known ASP methodology of separating the program into three sections: generate, define and test. In our case, the "define" and the "test" sections would respectively be Π_4 and (10), but we still miss a "generate" part, capable of considering different alternative conformant plans. The problem in this case is that we cannot use a simple choice:

$$toggle(L) \vee {\sim}toggle(L)$$

because this would allow a same action to be executed in some of the stable models and not executed in others, all inside a *same* world view. Let us assume that our epistemic semantics has some way to non-deterministically generate

[2] For simplicity, we omit time arguments or inertia, as they are not essential for the discussion.

a world view in which either $\mathbf{K}\, a$ or $\mathbf{K}\, \texttt{not}\ a$ holds using a given set of rules[3] $Choice(a)$. Then, take the program Π_5 consisting of rules

$$Choice(toggle(L)) \tag{11}$$

with $L \in \{l_1, l_2\}$ plus Π_4 and (10). If our semantics satisfies epistemic splitting, it is safe to obtain the world views in three steps: generate first the alternative world views for $toggle(l_1)$ and $toggle(l_2)$ using (11), apply Π_4 and rule out those world views not satisfying the goal *light* in all situations using (10). To fulfill the preconditions for applying splitting, we would actually need to replace regular literal $toggle(L)$ by $\mathbf{K}\, toggle(L)$ in all the bodies of Π_4, but this is safe in the current context. Now, we take the bottom program to obtain 4 possible world views $W_0 = [\{toggle(l_1)\}]$, $W_1 = [\{toggle(l_2)\}]$, $W_2 = [\{toggle(l_1), toggle(l_2)\}]$ and $W_3 = [\emptyset]$. When we combine them with the top Π_4 we obtain W_0' consisting of two stable models:

$$\{toggle(l_1), plugged(l_2), light, \dots\} \qquad \{toggle(l_1), {\sim}plugged(l_2), light, \dots\}$$

and W_1' consisting of other two stable models:

$$\{toggle(l_2), plugged(l_2), light, \dots\} \qquad \{toggle(l_2), {\sim}plugged(l_2), \dots\}$$

where the latter does not contain *light*. Finally, constraint (10) would rule out W_1'.

To sum up, epistemic splitting provides a natural way of formulating conformant planning problems by a separation into three sections: a generation part, the usual encoding of the actions scenario and a test part consisting of a subjective constraint to guarantee that the goal is always reached.

5 Splitting in Some Existing Semantics

In this section we study the property of epistemic splitting for the approaches mentioned in the introduction. We will begin by stating that G91 actually satisfies this property. The proof of the following theorm can be found in the appendix.

Main Theorem. *Semantics G91 satisfies epistemic splitting.* □

A similar proof can be developed to show that [12], that generalises[4] [3] from subjective literals to subjective formulas, also satisfies epistemic splitting.

To illustrate the behaviour of other semantics with respect to splitting, we will use several examples. Let us take the program Π_6 consisting of rules:

$$a \vee b \tag{12}$$

$$c \vee d \leftarrow \texttt{not}\ \mathbf{K}\, a \tag{13}$$

[3] For instance, in the G91-semantics, this could be just the rule $a \leftarrow \texttt{not}\ \mathbf{K}\, \texttt{not}\ a$. Other semantics may have alternative ways of expressing this intended behaviour.

[4] In fact, [12] defines several semantics but, among them, we refer here to the epistemic stable model semantics.

The set $U = \{a, b\}$ splits the program into the bottom (12) and the top (13). The bottom has a unique world view $W_b = [\{a\}, \{b\}]$ so $\mathbf{K}\, a$ does not hold and the top is simplified as $E_U(\Pi_6, W_b)$ containing the unique rule

$$c \vee d \leftarrow \texttt{not}\ \bot \tag{14}$$

This program has a unique world view $W_t = [\{c\}, \{d\}]$ that, combined with W_b yields $[\{a, c\}, \{b, c\}, \{a, d\}, \{b, d\}]$ as the unique solution for Π_6, for any semantics satisfying epistemic splitting (and so, also for G91). Let us elaborate the example a little bit further. Suppose we add now the constraint:

$$\bot \leftarrow c \tag{15}$$

The top must also include this rule and has now a unique stable model, yielding the world view $W_t = [\{d\}]$, so the world view for the complete program would be $[\{a, d\}, \{b, d\}]$. Finally, let us forbid the inclusion of atom d too:

$$\bot \leftarrow d \tag{16}$$

so we consider $\Pi_7 = \{(12), (13), (15), (16)\}$. This last constraint leaves the simplified top program $E_U(\Pi_6, W_b) = \{(14), (15), (16)\}$ without stable models, so epistemic splitting would yield that program Π_7 has no world view at all. This is the result we obtain, indeed, in [3,4][5] and in [12]. Surprisingly, recent approaches like [2,6,10,11] yield world view $[\{a\}]$, violating the epistemic splitting property. For instance, in the case of [6], the reduct of Π_7 with respect to $[\{a\}]$ is the program

$$\begin{array}{lll} a \leftarrow \texttt{not}\ b & c \vee d \leftarrow \texttt{not}\ a & \bot \leftarrow c \\ b \leftarrow \texttt{not}\ a & & \bot \leftarrow d \end{array}$$

which has a unique stable model $\{a\}$. As a second example, take the program Π_8 consisting of the same bottom program (12) and the rule:

$$c \leftarrow \mathbf{K}\, a \tag{17}$$

As expected, all approaches agree that Π_8 has a unique world view $W_b = [\{a\}, \{b\}]$ because $\mathbf{K}\, a$ is not satisfied and rule (17) is not applicable. Under epistemic splitting, we get that $E_U(\Pi_8, W_b)$ is the rule:

$$c \leftarrow \bot \tag{18}$$

whose unique world view is $[\emptyset]$, so that $W_b \sqcup [\emptyset] = W_b$. But let us further elaborate the example taking Π_8' containing Π_8 plus:

$$\bot \leftarrow \texttt{not}\ c \tag{19}$$

[5] These two semantics actually produce empty world views, but as we said before, we disregard them, as they just point out that the program has no solution.

Under epistemic splitting, the new top $E_U(\Pi'_8, W_b)$ contains now (19) and (18) which have no stable models. As a result, no world view can be combined with W_b and we obtain that Π'_8 has no world views at all. This is the result we obtain under [3,12], which agree that the program is inconsistent. However, [4] joins [2,6,10,11] in the group of approaches that provide the world view $[\{a, c\}]$. That is, in all these approaches, adding a constraint intended to remove all belief sets that do not satisfy c, may surprisingly lead to justify c. Note that, according to [3,12], the reduct of Π'_8 with respect to $[\{a, c\}]$ is $\{a \leftarrow \texttt{not}\ b,\ b \leftarrow \texttt{not}\ a,\}$ $c \leftarrow \top,\ \bot \leftarrow \texttt{not}\ c$ which has two stable models, $\{a, c\}$ and $\{b, c\}$, so $[\{a, c\}]$ is not a world view. In contrast, the reduct with respect to [4] and [6] is $\{a \leftarrow \texttt{not}\ b,\}$ $b \leftarrow \texttt{not}\ a,\ c \leftarrow a,\ \bot \leftarrow \texttt{not}\ c$ which has a unique stable model $\{a, c\}$, so $[\{a, c\}]$ is a world view.

6 Conclusions

We have introduced a formal property for semantics of epistemic specifications. This property that we call *epistemic splitting* has a strong resemblance to the splitting theorem well-known for regular ASP programs. Epistemic splitting can be applied when we can divide an epistemic logic program into a bottom part for a subset U of atoms and a top part, that only refers to atoms in U through subjective literals (those using modal epistemic operators). When this happens, the property of splitting states that we should be able to compute the world views of the program in two steps: first, computing the world views of the bottom and, second, using each bottom world view W to replace subjective literals for atoms in U in the top by their truth value with respect to W.

We have studied several consequences of epistemic splitting: for instance, if the program is stratified with respect to subjective literals then it will have a unique world view, at most. Another consequence is that constraints only consisting of subjective literals will have a monotonic behaviour, ruling out world views that satisfy the constraint body.[6] We have also explored how epistemic splitting may facilitate the simple application of the generate-define-test methodology, well-known in ASP, to the formalisation of conformant planning. The application of epistemic specifications to conformat planning was first discussed in [6], though with a more complex formulation due to the lack of epistemic splitting.

Our study of the main semantics in the literature has shown that only the original semantics [3] (G91), and its generalisation [12], satisfy epistemic splitting while the rest of approaches we considered do not, as we showed with counterexamples. As said in the introduction, a different kind of epistemic splitting had also been proved for G91 in [13], reinforcing the idea that this semantics can be interpreted in a modular way. Notice that the sets of programs that can be split under these two definitions is incomparable. We do not mean with this, however, that G91 is always intuitive. As it is well-known, G91 suffers from self-supportedness: for instance, the program consisting of the single rule $p \leftarrow \mathbf{K}\, p$

[6] The lack of monotonicity suffered by epistemic constraints in some semantics has been recently discussed by [7].

yields two world views $[\emptyset]$ and $[\{p\}]$ but the latter justifies p by the mere assumption of $\mathbf{K}\,p$ without further evidence, something that seems counterintuitive. What we claim instead is that G91 has a reasonable behaviour when subjective literals are stratified. Unfortunately, later attempts to solve self-supportedness on cyclic epistemic specifications have somehow spoiled that feature.

Acknowledgements. We are thankful to Michael Gelfond, Richard Watson and Patrick T. Kahl for their helpful comments on early versions of this work. We are also thankful to the anonymous reviewers for their valuable feedback, which helped to improve the paper.

References

1. Cabalar, P., Fandinno, J., Fariñas del Cerro, L.: Splitting epistemic logic programs. In: Proceedings of the 17th International Workshop on Non-monotonic Reasoning (NMR 2018), pp. 81–89 (2018)
2. Fariñas del Cerro, L., Herzig, A., Su, E.I.: Epistemic equilibrium logic. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2015), pp. 2964–2970. AAAI Press (2015)
3. Gelfond, M.: Strong introspection. In: Dean, T.L., McKeown, K. (eds.) Proceedings of the AAAI Conference, vol. 1, pp. 386–391. AAAI Press/The MIT Press (1991)
4. Gelfond, M.: New semantics for epistemic specifications. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS (LNAI), vol. 6645, pp. 260–265. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20895-9_29
5. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the 5th International Conference on Logic Programming (ICLP 1988), pp. 1070–1080 (1988)
6. Kahl, P., Watson, R., Balai, E., Gelfond, M., Zhang, Y.: The language of epistemic specifications (refined) including a prototype solver. J. Log. Comput. (2015). (Special issue article)
7. Leclerc, A.P., Kahl, P.T.: Epistemic logic programs with world view constraints. In: Technical communication of the 34th International Conference on Logic Programming (ICLP 2018) (2018)
8. Leclerc, A.P., Kahl, P.T.: A survey of advances in epistemic logic program solvers. In: Proceedings of the 11th International Workshop on Answer Set Programming and Other Computer Paradigms (ASPOCP 2018) (2018)
9. Lifschitz, V., Turner, H.: Splitting a logic program. In: Proceedings of the International Conference on Logic Programming (ICLP 1994), pp. 23–37. MIT Press (1994)
10. Shen, Y., Eiter, T.: Evaluating epistemic negation in answer set programming (extended abstract). In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2017), pp. 5060–5064 (2017)
11. Son, T.C., Le, T., Kahl, P.T., Leclerc, A.P.: On computing world views of epistemic logic programs. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2017), pp. 1269–1275 (2017). https://www.ijcai.org

12. Truszczyński, M.: Revisiting epistemic specifications. In: Balduccini, M., Son, T.C. (eds.) Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning. LNCS (LNAI), vol. 6565, pp. 315–333. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20832-4_20
13. Watson, R.: A splitting set theorem for epistemic specifications. CoRR: Proceedings of the 8th International Workshop on Non-monotonic Reasoning, NMR 2000 cs.AI/0003038 (2000). http://arxiv.org/abs/cs.AI/0003038