Artificial Intelligence 237 (2016) 115–135

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Evaluating epistemic negation in answer set programming

Yi-Dong Shen [a,*], Thomas Eiter [b]

[a] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[b] Institut für Informationssysteme, Technische Universität Wien, Favoritenstrasse 9-11, A-1040 Vienna, Austria

ARTICLE INFO

Article history:
Received 15 August 2015
Received in revised form 31 December 2015
Accepted 12 April 2016
Available online 22 April 2016

Keywords:
Answer set programming
Epistemic negation
Semantics

ABSTRACT

Epistemic negation **not** along with default negation ¬ plays a key role in knowledge representation and nonmonotonic reasoning. However, the existing epistemic approaches such as those by Gelfond [13,15,14], Truszczynski [33] and Kahl et al. [18] behave not satisfactorily in that they suffer from the problems of unintended world views due to recursion through the epistemic modal operator **K** or **M** (**K***F* and **M***F* are shorthands for ¬**not** *F* and **not**¬*F*, respectively). In this paper we present a new approach to handling epistemic negation which is free of unintended world views and thus offers a solution to the long-standing problem of epistemic specifications which were introduced by Gelfond [13] over two decades ago. We consider general logic programs consisting of rules of the form $H \leftarrow B$, where H and B are arbitrary first-order formulas possibly containing epistemic negation, and define a general epistemic answer set semantics for general logic programs by introducing a novel program transformation and a new definition of world views in which we apply epistemic negation to minimize the knowledge in world views. The general epistemic semantics is applicable to extend any existing answer set semantics, such as those defined in [26,27,32,1,8,12,29], with epistemic negation. For illustration, we extend FLP answer set semantics of Faber et al. [8] for general logic programs with epistemic negation, leading to epistemic FLP semantics. We also extend the more restrictive well-justified FLP semantics of Shen et al. [29], which is free of circularity for default negation, to an epistemic well-justified semantics. We consider the computational complexity of epistemic FLP semantics and show that for a propositional program Π with epistemic negation, deciding whether Π has epistemic FLP answer sets is Σ_3^p-complete and deciding whether a propositional formula F is true in Π under epistemic FLP semantics is Σ_4^p-complete in general, but has lower complexity for logic programs that match normal epistemic specifications, where the complexity of world view existence and query evaluation drops by one level in the polynomial hierarchy.

1. Introduction

Answer set programming (ASP) is a major logic programming paradigm rooted in knowledge representation and reasoning (KR) for modeling and solving knowledge-intensive search and optimization problems such as product configuration and planning [2]. In ASP, the semantics of a logic program is given by a set of intended models, called stable models or answer sets [16,17]. Such answer sets can be defined in different ways; Lifschitz [21] listed thirteen of them in the literature. These

* Corresponding author.
E-mail addresses: ydshen@ios.ac.cn (Y.-D. Shen), eiter@kr.tuwien.ac.at (T. Eiter).

http://dx.doi.org/10.1016/j.artint.2016.04.004

semantics agree for normal logic programs, but show discrepancies for more general logic programs such as logic programs with aggregates [30,8], with external sources such as description logic programs (dl-programs) [7], and with propositional or first-order formulas [26,27,32,1,12]. Most recently Shen et al. [29] introduced a new one called the *well-justified FLP answer set semantics*, which is fundamentally distinct from other existing answer set semantics in that every answer set of a general logic program is justified by having a level mapping and thus is free of circular justifications. This semantics has been implemented over the well-known ASP reasoner DLVHEX.[1]

Negation is a key mechanism in ASP for reasoning with incomplete knowledge. There are two major types of negation, *default negation* and *epistemic negation*. A third, called *strong negation*, also appears in the literature; when default negation is available, strong negation is easily compiled away using new predicate symbols [17] and thus it can be omitted. By abuse of notation, in this paper we use $\neg$, **not** and $\sim$ to denote the three negation operators, respectively.[2] For a formula F, the default negation $\neg F$ of formula F expresses that there is no *justification* for adopting F in an answer set and thus F can be assumed false by default in the answer set; in contrast, the epistemic negation **not** F of F expresses that there is no evidence *proving* that F is true, i.e., F is false in some answer set. *Justification* in ASP is a concept defined over every individual answer set, while *provability* is a meta-level concept defined over a collection of answer sets, called a *world view*. This means the two types of negation are orthogonal operations, where default negation works *locally* on each individual answer set, and epistemic negation works *globally* at a meta level on each world view.

With both default and epistemic negation, ASP is enabled to reason with different incomplete knowledge. For example, we can use the rule

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

to concisely express the *presumption of innocence*, which states that one is presumed innocent if there is no evidence proving s/he is guilty. We can also use rules of the form

$$\neg p(X) \leftarrow \mathbf{not}\, p(X)$$

to explicitly state Reiter's *closed-world assumption* (CWA) [28], i.e., if there is no evidence proving $p(X)$ is true we jump to the conclusion that $p(X)$ is false.

However, observe that most of the existing answer set semantics, such as those defined in [17,26,27,32,1,8,12,29], only support default negation and they do not allow for epistemic negation.

Epistemic negation and specifications. In fact, the need for epistemic negation was long recognized in ASP by Gelfond in the early 1990s [13,15] and recently revisited in [14,33,19,18,4]. In particular, Gelfond [13] showed that formalization of CWA using default and strong negations with rules of the form

$$\sim p(X) \leftarrow \neg p(X)$$

as presented in [17], is problematic.[3] He then proposed to address the problem using two epistemic modal operators **K** and **M**. Informally, for a formula F, $\mathbf{K}F$ expresses that F is true in every answer set, and $\mathbf{M}F$ expresses that F is true in some answer set. Note that $\mathbf{M}F$ can be viewed as shorthand for $\neg\mathbf{K}\neg F$.[4]

In the sequel, by an *object literal* we refer to an atom A or its strong negation $\sim A$; a *default negated literal* is of the form $\neg L$, and a *modal literal* is of the form $\mathbf{K}L$, $\neg\mathbf{K}L$, $\mathbf{M}L$ or $\neg\mathbf{M}L$, where L is an object literal.

Gelfond [13] considered disjunctive logic programs with modal literals, called *epistemic specifications*, which consist of rules of the form

$$L_1 \vee \cdots \vee L_m \leftarrow G_1 \wedge \cdots \wedge G_n \tag{1}$$

where each L is an object literal and each G is an object literal, a default negated literal, or a modal literal. A *normal epistemic specification* consists of rules of the above form with $m = 1$. Given a collection $\mathcal{A}$ of interpretations as an *assumption*, a logic program Π is transformed into a *modal reduct* $\Pi^{\mathcal{A}}$ w.r.t. the assumption $\mathcal{A}$ by first removing all rules with a modal literal G that is not true in $\mathcal{A}$, then removing the remaining modal literals. The assumption $\mathcal{A}$ is defined to be a *world view* of Π if it coincides with the collection of answer sets of $\Pi^{\mathcal{A}}$ under the semantics defined in [17].

The problem with recursion through K. More recently, Gelfond [14] addressed the problem that applying the above approach to handle modal literals may produce unintuitive world views due to recursion through **K**. For example, consider a logic program $\Pi = \{p \leftarrow \mathbf{K}p\}$. The rule expresses that for any collection $\mathcal{A}$ of answer sets of Π and any $I \in \mathcal{A}$, if p is true in all answer sets in $\mathcal{A}$, then p is true in I. This amounts to saying that if p is true in all answer sets, then p is always true (in particular in all answer sets). Obviously, this rule is not informative and does not contribute to constructively building any answer set; thus it can be eliminated from Π, leading to $\Pi = \emptyset$. As a result, Π is expected to have a unique answer

[1] www.kr.tuwien.ac.at/research/systems/dlvhex.

[2] In many texts, *not* and $\neg$ are used to denote the default and strong negation operators, respectively.

[3] We will further discuss this issue in Remark 3 following Example 4.

[4] Note that $\neg\mathbf{K}F$ and $\sim\mathbf{K}F$ are semantically equivalent. In [13], $\mathbf{M}F$ is shorthand for $\sim\mathbf{K}\sim F$, while in [14], it is shorthand for $\sim\mathbf{K}\neg F$, which is semantically equivalent to $\neg\mathbf{K}\neg F$.