Artificial Intelligence 213 (2014) 1–41

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

# FLP answer set semantics without circular justifications for general logic programs ☆

Yi-Dong Shen [a,*], Kewen Wang [b], Thomas Eiter [c], Michael Fink [c], Christoph Redl [c], Thomas Krennwallner [c], Jun Deng [a]

[a] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[b] School of Computing and Information Technology, Griffith University, Brisbane, QLD 4111, Australia
[c] Institut für Informationssysteme, Technische Universität Wien, Favoritenstrasse 9-11, A-1040 Vienna, Austria

ARTICLE INFO

*Article history:*
Received 14 February 2013
Received in revised form 28 April 2014
Accepted 1 May 2014
Available online 9 May 2014

*Keywords:*
Answer set programming
Knowledge representation
Nonmonotonic reasoning
Logic programs with first-order formulas
Level mappings
Circular justifications

ABSTRACT

The answer set semantics presented by Faber et al. [27] has been widely used to define so called FLP answer sets for different types of logic programs. However, it was recently observed that when being extended from normal to more general classes of logic programs, this approach may produce answer sets with circular justifications that are caused by self-supporting loops. The main reason for this behavior is that the FLP answer set semantics is not fully constructive by a bottom up construction of answer sets. In this paper, we overcome this problem by enhancing the FLP answer set semantics with a level mapping formalism such that every answer set $I$ can be built by fixpoint iteration of a one-step provability operator (more precisely, an extended van Emden–Kowalski operator for the FLP reduct $fI\Pi^I$). This is inspired by the fact that under the standard answer set semantics, each answer set $I$ of a normal logic program $\Pi$ is obtainable by fixpoint iteration of the standard van Emden–Kowalski one-step provability operator for the Gelfond–Lifschitz reduct $\Pi^I$, which induces a level mapping. The enhanced FLP answer sets, which we call well-justified FLP answer sets, are thanks to the level mapping free of circular justifications. As a general framework, the well-justified FLP answer set semantics applies to logic programs with first-order formulas, logic programs with aggregates, description logic programs, HEX-programs etc., provided that the rule satisfaction is properly extended to such general logic programs. We study in depth the computational complexity of FLP and well-justified FLP answer sets for general classes of logic programs. Our results show that the level mapping does not increase the worst-case complexity of FLP answer sets. Furthermore, we describe an implementation of the well-justified FLP answer set semantics, and report about an experimental evaluation, which indicates a potential for performance improvements by the level mapping in practice.



☆ This article contains revised and significantly extended work presented at IJCAI-2011 [54] and AAAI-2012 [56].

* Corresponding author.
*E-mail addresses:* ydshen@ios.ac.cn (Y.-D. Shen), k.wang@griffith.edu.au (K. Wang), fink@kr.tuwien.ac.at (M. Fink), redl@kr.tuwien.ac.at (C. Redl), tkren@kr.tuwien.ac.at (T. Krennwallner), dengj@ios.ac.cn (J. Deng).

http://dx.doi.org/10.1016/j.artint.2014.05.001
0004-3702/

## 1. Introduction

Answer set programming (ASP) is a major logic programming paradigm rooted in knowledge representation and reasoning [47,50,43]. It is an emerging approach to modeling and solving search and optimization problems arising in many application areas of AI including planning, reasoning about actions, diagnosis, abduction, and beyond [3,9]. In ASP, the semantics of a logic program is given by a set of intended models, called stable models or answer sets [33,34]. In fact, answer sets can be equivalently defined in many different ways (which indicates intrinsic richness of the concept); Lifschitz [44] listed 13 of them, and yet more exist.

In this paper, we focus on one of the equivalent definitions of answer sets, called FLP answer sets [26,27], which is widely used. Like the seminal definition by Gelfond and Lifschitz [33], it uses a program reduct, but in contrast it does not modify the rules of a logic program. Informally, given an interpretation $I$ of a logic program $\Pi$, its FLP reduct w.r.t. $I$, denoted $f\Pi^I$, consists of all ground instances of rules in $\Pi$ whose bodies are satisfied by $I$; in analogy to Gelfond and Lifschitz [33], $I$ is then an FLP answer set of $\Pi$ if $I$ is a minimal model of $f\Pi^I$.

This definition has been motivated by giving an answer set semantics to logic programs with aggregates, and due to its simplicity and attractive properties (minimality of models is guaranteed), it can be easily deployed to other extensions of logic programs, provided that rule satisfaction is properly defined. This has been exploited for a variety of logic programs, including logic programs with aggregates or abstract constraint atoms (c-atoms) [26,27], description logic programs (dl-programs) [25,24], HEX-programs [25], tightly coupled dl-programs [46], modular logic programs [12], and logic programs with first-order formulas [4]. For convenience, we refer to all such extensions of normal logic programs in the unifying framework of the FLP answer set semantics as general logic programs.

However, it was recently observed that for general logic programs, the FLP answer set semantics may produce answer sets with circular justifications that are caused by self-supporting loops [58,45]. The following two examples well illustrate this behavior.

**Example 1.** Consider the following logic program with aggregates (borrowed from [60]):

$$\begin{aligned} \Pi_1: \quad & p(1). && r_1\\ & p(2) \leftarrow p(-1). && r_2\\ & p(-1) \leftarrow \mathrm{SUM}\big\langle X : p(X)\big\rangle \geq 1. && r_3 \end{aligned}$$

For any interpretation $I$ of $\Pi_1$, the aggregate function $\mathrm{SUM}\langle X : p(X)\rangle$ yields the sum $S$ of all $X \in \{-1, 1, 2\}$ such that $p(X)$ is true in $I$. The aggregate $\mathrm{SUM}\langle X : p(X)\rangle \geq 1$ is satisfied by $I$ if $S \geq 1$. Let $I = \{p(1), p(-1), p(2)\}$. Since $p(X)$ is true in $I$ for each $X \in \{-1, 1, 2\}$, the aggregate $\mathrm{SUM}\langle X : p(X)\rangle \geq 1$ is satisfied by $I$. The FLP reduct of $\Pi_1$ w.r.t. $I$ is $\Pi_1$ itself; i.e., $f\Pi_1^I = \Pi_1$. It is easy to check that $I$ is a minimal model of $f\Pi_1^I$, so $I$ is an answer set of $\Pi_1$ under the FLP answer set semantics [27]. Observe that this FLP answer set has a circular justification caused by the following self-supporting loop:

$$p(2) \quad \Leftarrow \quad p(-1) \quad \Leftarrow \quad \mathrm{SUM}\big\langle X : p(X)\big\rangle \geq 1 \quad \Leftarrow \quad p(2).$$

That is, $p(2)$ being in $I$ is due to $p(-1)$ being in $I$ (via $r_2$), while $p(-1)$ being in $I$ is due to $I$ satisfying the aggregate $\mathrm{SUM}\langle X : p(X)\rangle \geq 1$ (via $r_3$). Since the domain of $X$ in the aggregate function is $\{-1, 1, 2\}$, $I$ satisfying $\mathrm{SUM}\langle X : p(X)\rangle \geq 1$ is due to $p(2)$ being in $I$ (i.e., without $p(2)$, $I$ would not satisfy this aggregate). As a result, $p(2)$ is circularly supported (justified) in $I$ by itself.

**Example 2.** Consider the following logic program with classical logic formulas[1]:

$$\begin{aligned} \Pi_2: \quad & p(2) \leftarrow p(2) \wedge \big(\neg p(-1) \vee p(1)\big). && r_1\\ & p(-1) \leftarrow \neg p(-1) \vee p(1) \vee p(2). && r_2\\ & p(1) \leftarrow p(-1). && r_3 \end{aligned}$$

Note that the body and head of each rule in $\Pi_2$ are classical logic formulas. Consider the interpretation $I = \{p(-1), p(1)\}$. Since the body of rule $r_1$ is not satisfied by $I$, the FLP reduct of $\Pi_2$ w.r.t. $I$ is $f\Pi_2^I = \{r_2, r_3\}$. $I$ is a minimal model of $f\Pi_2^I$ and thus is an answer set of $\Pi_2$ under the FLP answer set semantics [4]. Observe that this FLP answer set has a circular justification caused by the following self-supporting loop:

$$p(1) \quad \Leftarrow \quad p(-1) \quad \Leftarrow \quad \neg p(-1) \vee p(1) \vee p(2) \quad \Leftarrow \quad p(1).$$

[1] Logic programs with classical logic formulas were recently introduced by Bartholomew et al. [4], which consist of rules of the form $H \leftarrow B$, where $H$ and $B$ are arbitrary first-order formulas. Normal logic programs can be viewed as a special form of logic programs with first-order formulas, where the negation *not* is identified with $\neg$, each rule head $H$ with an atom, and each rule body $B$ with a conjunction of literals. Answer sets of such logic programs are defined by the FLP answer set semantics.

That is, $p(1)$ being in $I$ is due to $p(-1)$ being in $I$ (via $r_3$), which in turn is due to $I$ satisfying $\neg p(-1) \vee p(1) \vee p(2)$ (via $r_2$). Since both $\neg p(-1)$ and $p(2)$ are false in $I$, $I$ satisfying $\neg p(-1) \vee p(1) \vee p(2)$ is due to $p(1)$ being in $I$. Therefore, $p(1)$ is circularly justified in $I$ by itself.

Our careful study reveals that the fundamental reason behind the circular justification problem for general logic programs is that FLP answer sets can not always be constructed in a bottom up fashion by iterated applications of rules; that is, they might lack a level mapping such that atoms in an answer set at upper levels are derived from atoms at lower levels by iterated applications of rules. We would like to stress that it is such a level mapping on answer sets that makes each *if-then* rule $H \leftarrow B$ in a logic program essentially different from an implication $B \supset H$ in classical logic. In fact, for normal logic programs Fages [28] showed that the standard answer set semantics of Gelfond and Lifschitz [33] has a level mapping on its answer sets. Since the FLP answer set semantics agrees with the standard answer set semantics for normal logic programs, answer sets of normal logic programs under the FLP answer set semantics are free of circular justifications.

In this paper, we remedy the circular justification problem of FLP answer sets for general logic programs by enhancing them with a level mapping formalism. Observe that for a normal logic program $\Pi$, each standard answer set $I$ is obtained by fixpoint iteration of the van Emden and Kowalski [64] one-step provability operator for the well-known Gelfond and Lifschitz [33] reduct $\Pi^I$; this process naturally induces a level mapping on $I$. Inspired by this, we define for a general logic program $\Pi$ answer sets $I$ by fixpoint iteration in a similar way such that a level mapping on $I$ is induced. We first extend the van Emden–Kowalski operator from positive to general logic programs, and then adapt the fixpoint construction of the standard answer set semantics from normal to general logic programs. To this end, we replace the Gelfond–Lifschitz reduct $\Pi^I$ with the FLP reduct $f\Pi^I$ and iterate the extended van Emden–Kowalski operator on $f\Pi^I$ to obtain its least fixpoint; $I$ is then an answer set if it coincides with this fixpoint.

We show that such defined answer sets are in fact FLP answer sets which, due to the naturally induced level mapping, are free of circular justifications; furthermore, such answer sets exclude only those FLP answer sets that have circular justifications. For this reason, we call such answer sets *well-justified FLP answer sets* and the according semantics the *well-justified FLP answer set semantics*.

The main contributions of this paper are summarized as follows:

(1) We define the well-justified FLP answer set semantics for logic programs with first-order formulas. To the best of our knowledge, this is the first answer set semantics that is free of circular justifications for logic programs of this kind. We further extend the well-justified FLP answer set semantics to logic programs with aggregates or c-atoms, i.e. logic programs with rules of the form $H \leftarrow B$, where $H$ and $B$ are first-order formulas extended with aggregate atoms or c-atoms. This is also the first answer set semantics that is free of circular justifications for such general logic programs. For logic programs whose rule heads are atoms, in [16,53] a three-valued fixpoint semantics was introduced, which defines answer sets (called *two-valued stable models*) that are free of circular justifications. We show that for this class of logic programs the two-valued stable models are well-justified FLP answer sets, but the converse is not true. This means that the two-valued stable models may exclude some FLP answer sets that are free of circular justifications. For normal programs with aggregates or c-atoms, in [60,59] a conditional satisfaction based answer set semantics was presented that agrees with the three-valued fixpoint semantics. We show that for this particular class of logic programs, the well-justified FLP answer set semantics agrees with the conditional satisfaction based semantics and thus agrees with the three-valued fixpoint semantics.

(2) We apply the well-justified FLP answer set semantics to dl-programs, which were introduced in [24] as a framework for combining answer set programming with description logics (DLs) [2] for the Semantic Web. A dl-program can be viewed as a normal logic program enhanced with an interface to query an external DL knowledge base. Weak, strong and FLP answer sets are three increasingly restrictive notions of answer sets for dl-programs in [25,24] that incorporate increasing levels of foundedness. As weak answer sets may be unfounded due to circular justifications by self-supporting positive loops, Eiter et al. [24] introduced strong answer sets which eliminate such unfoundedness. However, strong answer sets might not be minimal models in general, which motivated Eiter et al. to consider FLP answer sets [25]. However, both strong and FLP answer sets admit circular justifications in general, which might be undesired. We therefore introduce well-justified FLP answer sets for dl-programs; this is the first notion of answer sets for dl-programs that are free of circular justifications.

(3) We study in depth the computational complexity of the ordinary FLP and the well-justified FLP answer set semantics on the problems of answer set existence, cautious reasoning and brave reasoning. Since first-order logic is undecidable, it is clearly undecidable whether an arbitrary general logic program has an ordinary resp. a well-justified FLP answer set, even in absence of function symbols. We focus here on propositional logic programs and consider aggregates that are computable in polynomial time. For dl-programs, we consider three expressive DLs: $\mathcal{SHIF}(\mathbf{D})$, $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SROIQ}(\mathbf{D})$, which are the logical underpinnings of the Web ontology languages OWL Lite, OWL DL [40] and OWL 2 [38,36], respectively. Our results show that the level mapping of well-justified FLP answer sets does not increase the worst-case complexity.

(4) We describe an implementation of the well-justified FLP answer set semantics and report about an experimental evaluation which compares the ordinary and the well-justified FLP answer set semantics on a benchmark suite. The results indicate an interesting potential of the well-justified FLP-answer set semantics for performance improvements in prac-

tice. Indeed, in a number of cases well-justified FLP answer sets are computed faster than ordinary FLP answer sets; in some cases, few well-justified FLP answer sets exist while in others they coincide with all FLP answer sets. Intuitively, this is connected to the number of iterations in the deterministic fixpoint construction of well-justified FLP answer sets, compared to the non-constructive minimality check for ordinary FLP answer sets. This suggests to search for well-justified FLP answer sets first, and then fall back to ordinary FLP answer sets if no well-justified ones have been found.

As a general framework, the well-justified FLP answer set semantics can be easily deployed to other kinds of logic programs, such as HEX-programs, tightly coupled dl-programs, modular logic programs, etc., provided that the satisfaction relation is extended to these general logic programs.

**Structure** The rest of this paper is organized as follows. In Section 2, we introduce logic programs with first-order formulas and define the ordinary FLP answer set semantics for them. In Section 3, we define level mappings for logic programs with first-order formulas. In Section 4, we introduce the well-justified FLP answer set semantics for such logic programs, while in Sections 5 and 6, we extend the well-justified FLP answer set semantics to logic programs with aggregates and to dl-programs, respectively. In Section 7, we study the computational complexity of the ordinary and the well-justified FLP answer set semantics. We describe in Section 8 our implementation and present an experimental evaluation. In Section 9, we review related work, while in Section 10 we give a summary and present issues for future work. For clarity and in order not to distract from reading, proofs of the results have been moved to Appendix A.

## 2. A first-order logic language

In this section, we first recall concepts and fix notation for first-order logic under the standard names assumptions, and then introduce logic programs with first-order formulas and their FLP answer set semantics.

### 2.1. First-order logic

We denote by $\mathcal{L}_\Sigma$ the first-order logic language with equality over *signature* $\Sigma = (\mathcal{P}, \mathcal{F})$, where $\mathcal{P}, \mathcal{F}$ are countable sets of *predicate* and *function* symbols of arities $\geq 0$, respectively; $\mathcal{C} \subseteq \mathcal{F}$ denotes the set of 0-ary function symbols, which are called *constants*. Given a countable set $\mathcal{V}$ of *variables*, *terms* and *atoms* are defined as usual, and *formulas* are constructed from atoms with connectives $\neg, \wedge, \vee, \supset, \equiv$ and quantifiers $\exists$ and $\forall$. *Literals* are atoms $A$ or their negation $\neg A$.

A term, atom or formula is *ground* if no variable occurs in it; we denote by $\mathcal{N}_\Sigma$ and $\mathcal{H}_\Sigma$ the sets of all ground terms and ground atoms of $\Sigma$, respectively. A formula is *closed* if it has no free variables (i.e., all variables are in the scope of a quantifier). A (first-order) *theory* is a set of closed formulas.

An *interpretation* of $\mathcal{L}_\Sigma$ is a pair $I = \langle U, .^I \rangle$, where $U$ is a *domain*, and $.^I$ is a *mapping* which assigns to each $n$-ary predicate symbol $p \in \mathcal{P}$ a relation $p^I \subseteq U^n$, and each $m$-ary function symbol $f \in \mathcal{F}$ a function $f^I : U^m \to U$. A *variable assignment* $B$ for $I$ is a mapping which assigns an element $X^B \in U$ to each variable $X \in \mathcal{V}$. The interpretation of a term $t$, denoted $t^{I,B}$, is defined as usual, where $B$ is omitted when $t$ is ground. *Satisfaction* of a formula $F$ in $I$ relative to $B$ is defined as usual; $I$ is a *model* of $F$ if $I$ satisfies $F$ for every variable assignment $B$, and is a *model of* (or satisfies) a theory $O$ if $I$ is a model of every formula in $O$. A theory is *satisfiable* (or *consistent*) if it has some model. The *entailment* relation is defined in terms of satisfaction as usual; i.e., a theory $O$ *entails* a formula $F$, or $F$ is *true* in $O$, denoted $O \models F$, if every model of $O$ is a model of $F$.

#### 2.1.1. Standard Names Assumption (SNA)

In order to access in $I = \langle U, .^I \rangle$ all elements of the domain $U$ by name, we employ the *standard names assumption*, cf. [49,14], i.e., (1) $\Sigma$ includes a countably infinite set of constants and the binary equality predicate symbol $\approx$, (2) $U = \mathcal{N}_\Sigma$ and $t^{\mathcal{I}} = t$ for each $t \in \mathcal{N}_\Sigma$, and (3) $\approx^I$ is a congruence relation over $U$, i.e., a reflexive, symmetric and transitive relation that allows replacement of equals by equals.

In such interpretations $I$, which are called *SNA interpretations*, every variable assignment over the domain $U$ amounts to a substitution of variables over $\mathcal{N}_\Sigma$. Moreover, since $p^I \subseteq \mathcal{N}_\Sigma^n$ for each $n$-ary predicate symbol $p \in \mathcal{P}$, the SNA interpretations of $\mathcal{L}_\Sigma$ are in 1–1 correspondence with the subsets of $\mathcal{H}_\Sigma$. We thus view each SNA interpretation $I$ as a subset of $\mathcal{H}_\Sigma$, such that $I$ satisfies a ground atom $A$ if $A \in I$, and satisfies $\neg A$ if $A \notin I$.

It is well-known that SNA interpretations preserve satisfiability, i.e., a first-order formula is satisfiable if and only if it is satisfiable in a model employing the standard name assumption, cf. [32]. Therefore, in the sequel we consider only SNA interpretations $I \subseteq \mathcal{H}_\Sigma$; for convenience, we let $I^- = \mathcal{H}_\Sigma \setminus I$ and $\neg I^- = \{\neg A \mid A \in I^-\}$ and refer with "function symbols" tacitly to function symbols of positive arity ($m > 0$).

### 2.2. Logic programs with first-order formulas

We extend the language $\mathcal{L}_\Sigma$ with *rules* of the form $H \leftarrow B$, where $H$ and $B$ are first order formulas. Such a rule $r$ expresses an *if-then* statement, saying that *if* the logic property $B$ holds, *then* infer $H$. We then define:

**Definition 1.** A *logic program with first-order formulas* (briefly, *logic program*) is a finite set of rules. It is a *normal logic program* if each rule is of the form

$$A_0 \leftarrow A_1 \wedge \cdots \wedge A_m \wedge \neg A_{m+1} \wedge \cdots \wedge \neg A_n, \tag{1}$$

where each $A_i$ is an atom without equality and function symbols, and a *positive logic program* if moreover $m = n$.

For a rule $r$ of the form $H \leftarrow B$, we use $body(r)$ to refer to $B$, which may be empty (in that case, we omit $\leftarrow$), and $head(r)$ to refer to $H$; if $r$ is of the form (1), we use $pos(r)$ and $neg(r)$ to denote the conjunctions $A_1 \wedge \cdots \wedge A_m$ and $\neg A_{m+1} \wedge \cdots \wedge \neg A_n$ of positive and negative literals, respectively.

Rules in a logic program $\Pi$ may have free variables. In ASP, these free variables will be instantiated over an application specific domain $\mathcal{C}_\Pi$ which is a non-empty, finite subset of $\mathcal{C}$ and includes all constants occurring in $\Pi$. A *closed instance* of a rule $r$ over $\mathcal{C}_\Pi$ is obtained by replacing every free variable in $r$ with some constant in $\mathcal{C}_\Pi$. The *grounding* of a rule $r$ w.r.t. $\mathcal{C}_\Pi$ is the set $ground(r, \mathcal{C}_\Pi)$ of all closed instances of $r$ over $\mathcal{C}_\Pi$, and the grounding of $\Pi$ is $ground(\Pi, \mathcal{C}_\Pi) = \bigcup_{r\in\Pi} ground(r, \mathcal{C}_\Pi)$. Note that $ground(\Pi, \mathcal{C}_\Pi)$ is finite.

With no loss in generality, we assume that the domain $\mathcal{C}_\Pi$ consists of all constants in $\Pi$ (in case that some constant $a$ of the domain does not appear in $\Pi$, we may have it by adding to $\Pi$ a dummy rule $p(a) \leftarrow p(a)$). Then for any logic program $\Pi$, $\mathcal{C}_\Pi$ is unique, and for convenience we omit $\mathcal{C}_\Pi$ from $ground(r, \mathcal{C}_\Pi)$ and $ground(\Pi, \mathcal{C}_\Pi)$.

**Remark.** In a logic program $\Pi$, each rule $H \leftarrow B$ with the set $S$ of free variables may also be viewed as a *globally* universally quantified rule $\forall_S(H \leftarrow B)$, where the domain of each variable in $S$ is $\mathcal{C}_\Pi$ while the domain of the other (*locally* quantified) variables is $\mathcal{N}_\Sigma$. Only globally universally quantified variables will be instantiated over their domain $\mathcal{C}_\Pi$ for the grounding $ground(\Pi)$.

An interpretation $I$ *satisfies* a closed instance $r$ of a rule if it either satisfies $head(r)$ or it does not satisfy $body(r)$; $I$ is a *model* of a logic program $\Pi$ if $I$ satisfies every $r \in ground(\Pi)$. Moreover, a model $I$ is *minimal* if $\Pi$ has no model $J$ that is a proper subset of $I$.

Thus semantically, we may view a logic program $\Pi$ as shorthand for $ground(\Pi)$, where each free variable in $\Pi$ is viewed as shorthand for constants in $\mathcal{C}_\Pi$ and each rule $r \in \Pi$ is viewed as shorthand for $ground(r)$.

A *propositional theory/formula* is a special first-order theory/formula that contains no variables and no function symbols. A *propositional logic program* is a logic program whose rule heads and bodies are propositional formulas. A normal logic program $\Pi$ can be viewed as a special propositional logic program by grounding, i.e., $ground(\Pi)$ is a propositional logic program.

The *Herbrand base* of a propositional logic program $\Pi$ w.r.t. the domain $\mathcal{C}_\Pi$, denoted $\mathcal{HB}_\Pi$, is the set of ground atoms $p(a_1, \ldots, a_n)$, where $p$ occurs in $\Pi$ and each $a_i$ is in $\mathcal{C}_\Pi$. Any $I \subseteq \mathcal{HB}_\Pi$ is a *Herbrand interpretation* of $\Pi$. Herbrand models are defined as usual, where the equality $\approx$ is interpreted as identity under the unique name assumption (UNA); i.e., for all distinct $a_i, a_j \in \mathcal{C}_\Pi$ we assume UNA axioms of the form $\neg(a_i \approx a_j)$ to be implicitly present in $\Pi$. For a Herbrand interpretation $I$, we denote $I^-$ for $\mathcal{HB}_\Pi \setminus I$ and $\neg I^-$ for $\{\neg A \mid A \in I^-\}$.

We now define the *ordinary FLP answer set semantics* (briefly, the FLP answer set semantics) for logic programs as follows.

**Definition 2.** Let $\Pi$ be a logic program and $I$ an interpretation. The *FLP-reduct* of $\Pi$ w.r.t. $I$ is $f\Pi^I = \{r \in ground(\Pi) \mid I \text{ satisfies } body(r)\}$, and $I$ is an *FLP answer set* of $\Pi$ if $I$ is a minimal model of $f\Pi^I$.

**Example 3.** Consider the logic program $\Pi_2$ in Example 2. By Definition 2, the interpretation $I = \{p(-1), p(1)\}$ is an FLP answer set of $\Pi_2$, where $ground(\Pi_2) = \Pi_2$ and $f\Pi_2^I = \{r_2, r_3\}$.

## 3. Level mappings for logic programs with first-order formulas

As mentioned in the introduction, *if-then* rules $H \leftarrow B$ in a logic program essentially differ from material implications $B \supset H$ in classical logic because rules induce a level mapping on each answer set such that answers at upper levels are derived from answers at lower levels by applying the rules in the way that *if* the body of a rule is true in answers at lower levels *then* infer its head. A typical example is that the formula $\neg A \supset A$ is equivalent to $A$ in classical logic, but the rule $A \leftarrow \neg A$ fundamentally differs from $A$ in logic programs because we can never infer $A$ by applying the rule $A \leftarrow \neg A$ unless we are given $\neg A$.

For a normal logic program $\Pi$, Fages [28] introduced a notion of well-supportedness. An interpretation $I$ is *well-supported* if there exists a strict well-founded partial order $\prec$ on $I$ such that for any $A \in I$ there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$, where $I$ satisfies $body(r)$ and $B \prec A$ for every positive literal $B$ in $body(r)$. A binary relation $\leq$ is *well-founded* if there is no infinite decreasing chain $A_0 \geq A_1 \geq \cdots$. Clearly, a well-supported interpretation $I$ induces a level mapping via the partial order $\prec$, i.e., for any $A \in I$, there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $I$ satisfies $body(r)$ and for every positive literal $B$ in $body(r)$, $B$ is at a lower level than $A$.

Fages' definition of well-supportedness cannot be applied to general logic programs whose rule heads or bodies are arbitrary first-order formulas. For instance, the well-supportedness is not applicable to the logic program $\Pi = \{A \leftarrow A \vee \neg A\}$, where $A$ is a ground atom and $A \vee \neg A$ is a tautology that is always true in first-order logic. Intuitively, $I = \{A\}$ should be a well-supported interpretation of $\Pi$, where $A$ is supported by the tautological body $A \vee \neg A$ of the rule. However, $I$ is not well-supported under Fages' definition since $A \prec A$ does not hold for any strict well-founded partial order $\prec$ on $I$.

In this section, we define level mappings of interpretations for logic programs with first-order formulas. We first introduce a notion of partitioning of an interpretation.

**Definition 3.** A *partitioning* of an interpretation $I$ is of the form $\langle S_0, S_1, \cdots, S_m\rangle$, where $S_0 = \neg I^-$, $\bigcup_{1\leq i\leq m} S_i = I$, $S_i \neq \emptyset$ for every $i > 0$, and $S_i \cap S_j = \emptyset$ for every $i \neq j$.

Note that $\neg I^-$ is included in every partitioning of $I$ since it is the negative half of $I$. We then define level mappings over such partitionings.

For logic programs whose rule heads are atoms, level mappings of an interpretation can be formalized as follows.

**Definition 4.** Let $\Pi$ be a logic program whose rule heads are atoms, $I$ an interpretation of $\Pi$, and $S = \langle S_0, S_1, \cdots, S_m\rangle$ a partitioning of $I$. $S$ is a *level mapping* of $I$ if for every $A \in S_k$ where $k > 0$, there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $body(r)$ is true in $\bigcup_{0\leq i\leq k-1} S_i$ (i.e., $\bigcup_{0\leq i\leq k-1} S_i \models body(r)$, where $\models$ is the entailment relation).

For a level mapping $\langle S_0, S_1, \cdots, S_m\rangle$, the atoms in $S_i$ are said to be at a higher (resp. lower) level than the atoms in $S_j$ if $i > j$ (resp. $i < j$). When an interpretation $I$ has such a level mapping, every $A \in I$ with $A \in S_k$ is supported/justified by the body of a rule $r$ in $ground(\Pi)$, where $head(r) = A$ and $body(r)$ is true in $\bigcup_{0\leq i\leq k-1} S_i$. Since all atoms in $\bigcup_{0\leq i\leq k-1} S_i$ are at lower levels than $A$, $A$ is non-circularly justified. Thus $I$ is non-circularly justified.

**Example 4.** Consider the logic program $\Pi = \{A \leftarrow A \vee \neg A\}$ and let $I = \{A\}$. $I$ has a level mapping $\langle \neg I^-, \{A\}\rangle$, where $A$ is justified by the tautology $A \vee \neg A$ that is true in $\neg I^-$. This justification is clearly non-circular.

Consider another logic program $\Pi = \{A \leftarrow \neg C,\ B \leftarrow A \wedge \neg C\}$ and let $I = \{A, B\}$, where $A$, $B$, $C$ are ground atoms. Then, $\neg C$ is in $\neg I^-$. $I$ has a level mapping $\langle \neg I^-, \{A\}, \{B\}\rangle$. Note that $B$ is justified (via the second rule) by $A$ and $\neg C$ at lower levels, while $A$ is justified (via the first rule) by $\neg C \in \neg I^-$. These justifications are non-circular.

When $I$ has no level mapping, for any partitioning $\langle S_0, S_1, \cdots, S_m\rangle$, there must be some $A \in I$ with $A \in S_k$ for which we cannot find a rule $r \in ground(\Pi)$ to non-circularly support $A$ under the partitioning (i.e., $head(r) = A$ and $body(r)$ is true in $\bigcup_{0\leq i\leq k-1} S_i$). For instance, consider a logic program $\Pi = \{A \leftarrow \neg A\}$ and let $I = \{A\}$. The only partitioning of $I$ is $\langle \neg I^-, \{A\}\rangle$. Since $\neg A$ is not in $\neg I^-$, $\Pi$ has no rule to support $A$ under the partitioning. Hence $I$ has no level mapping.

For a logic program whose rule bodies and heads are arbitrary first-order formulas, the definition of a level mapping is a bit more complicated. The key idea of our approach is: to define level mappings of an interpretation $I$, we first define level mappings of rule heads w.r.t. $I$ in a way similar to Definition 4 by viewing each rule head as a *macro-atom*; then level mappings of $I$ are defined in terms of the level mappings of rule heads w.r.t. $I$.

**Definition 5.** Let $\Pi$ be a logic program. A *partitioning of rule heads* of $\Pi$ is of the form $S' = \langle S'_1, \cdots, S'_n\rangle$, where each $S'_i$ is a nonempty set of rule heads in $ground(\Pi)$ and $S'_i \cap S'_j = \emptyset$ for every $i \neq j$.

For a given interpretation $I$, let $S'_0 = \neg I^-$. Then for a partitioning $S'$ of rule heads of $\Pi$ to be a level mapping w.r.t. $I$, we expect that for each $k > 0$ the rule heads in $S'_k$ are obtainable from $\bigcup_{0\leq i\leq k-1} S'_i$ by applying rules in $ground(\Pi)$.

**Definition 6.** Let $\Pi$ be a logic program, $I$ an interpretation of $\Pi$, and $S' = \langle S'_1, \cdots, S'_n\rangle$ a partitioning of rule heads of $\Pi$. Let $S'_0 = \neg I^-$. Then $S'$ is a *level mapping of rule heads of $\Pi$ w.r.t. $I$* if for every $H \in S'_k$ where $k > 0$, there is a rule $H \leftarrow body(r)$ in $ground(\Pi)$ such that $body(r)$ is true in $\bigcup_{0\leq i\leq k-1} S'_i$. Furthermore, $S'$ is called a *total* level mapping if in addition $\bigcup_{1\leq i\leq n} S'_i$ consists of all rule heads $head(r)$ in $ground(\Pi)$ such that $body(r)$ is true in $\bigcup_{0\leq i\leq n} S'_i$; otherwise, $S'$ is called a *partial* level mapping.

When $S' = \langle S'_1, \cdots, S'_n\rangle$ is a level mapping of rule heads w.r.t. $I$, then every rule head $H \in S'_k$ where $k > 0$ is supported by the body of a rule $r$ in $ground(\Pi)$ such that $head(r) = H$ and $body(r)$ is true in $\bigcup_{0\leq i\leq k-1} S'_i$. Note that all rule heads in $\bigcup_{1\leq i\leq k-1} S'_i$ are at lower levels than $H$ (at level $k$) unless $H$ is a negative literal from $\neg I^-$ (hence it is also in $S'_0$). This means $H$ is non-circularly justified.

Since all rule heads in $\bigcup_{1\leq i\leq n} S'_i$ are non-circularly justified under the level mapping $S'$ w.r.t. $I$, all atoms in $I$ that are entailed by $\bigcup_{0\leq i\leq n} S'_i$ are non-circularly justified. This leads us to defining level mappings of an interpretation $I$ for arbitrary logic programs in terms of level mappings of rule heads w.r.t. $I$.

**Definition 7.** Let $\Pi$ be a logic program, $I$ an interpretation of $\Pi$, and $S = \langle S_0, S_1, \cdots, S_m\rangle$ a partitioning of $I$. Let $S'_0 = \neg I^-$. Then $S$ is a *level mapping* of $I$ if there is a total level mapping $S' = \langle S'_1, \cdots, S'_n\rangle$ of rule heads of $\Pi$ w.r.t. $I$ and integers $l_1, \cdots, l_m$, where $1 \le l_1 < \cdots < l_m \le n$, such that for every $A \in S_k$ where $k > 0$, $\bigcup_{0\le i\le l_k} S'_i \models A$ but $\bigcup_{0\le i\le l_k-1} S'_i \not\models A$.

Note that the level index $l_k$ where $1 \le k \le m$ is used to connect $S_k$ with the atoms that are entailed by $\bigcup_{0\le i\le l_k} S'_i$ but not entailed by $\bigcup_{0\le i\le l_k-1} S'_i$. Obviously when $m = n$, $l_k = k$.

**Example 5.** Consider the following propositional logic program:

$$\Pi: \quad A \vee (\neg B \wedge C) \leftarrow \neg A \wedge (\neg C \vee C). \qquad r_1$$
$$D \leftarrow C. \qquad r_2$$

Let $I = \{C, D\}$ be an interpretation of $\Pi$; then $\neg A$, $\neg B$ are in $\neg I^-$. Consider a partitioning $S = \langle \neg I^-, S_1, S_2\rangle$ of $I$, where $S_1 = \{C\}$ and $S_2 = \{D\}$. Let $S' = \langle S'_1, S'_2\rangle$ be a partitioning of rule heads of $\Pi$, where $S'_1 = \{A \vee (\neg B \wedge C)\}$ and $S'_2 = \{D\}$. Since $\neg I^- \models body(r_1)$ and $\neg I^- \cup S'_1 \models body(r_2)$, $S'$ is a total level mapping of rule heads of $\Pi$ w.r.t. $I$. Let $l_1 = 1$ and $l_2 = 2$. $C \in S_1$ is entailed by $\neg I^- \cup S'_1$ but not by $\neg I^-$ and $D \in S_2$ is entailed by $\neg I^- \cup S'_1 \cup S'_2$ but not by $\neg I^- \cup S'_1$. By Definition 7 $S$ is a level mapping of $I$.

For logic programs whose rule heads are atoms, Definition 7 coincides with Definition 4.

**Proposition 1.** *Let $\Pi$ be a logic program whose rule heads are atoms, $I$ a model of $\Pi$, and $S = \langle S_0, S_1, \cdots, S_m\rangle$ a partitioning of $I$. Then $S$ is a level mapping of $I$ under Definition 4 if and only if it is a level mapping under Definition 7.*

The following characterization is useful in determining whether an interpretation of a logic program has a level mapping.

**Proposition 2.** *An interpretation $I$ of a logic program $\Pi$ has a level mapping as in Definition 7 if and only if there is a total level mapping $S' = \langle S'_1, \cdots, S'_n\rangle$ of rule heads of $\Pi$ w.r.t. $I$ such that $\bigcup_{0\le i\le n} S'_i \models A$ for every $A \in I$, where $S'_0 = \neg I^-$.*

## 4. Well-justified FLP answer sets for logic programs with first-order formulas

For logic programs with first-order formulas, the FLP answer set semantics from Definition 2 does not induce a level mapping for its answer sets. For an interpretation $I$ to be an answer set of a logic program $\Pi$, the FLP answer set semantics only requires $I$ to be a minimal model of the FLP reduct $f\Pi^I$. This amounts to treating all rules $H \leftarrow B$ in $f\Pi^I$ as material implications $B \supset H$ in classical logic, because $I$ is a model of the rules $H \leftarrow B$ in $f\Pi^I$ if and only if $I$ is a model of the corresponding implications $B \supset H$ in classical logic. As classical logic does not induce level mappings for its models, an FLP answer set (i.e., a minimal model of $f\Pi^I$) may not have a level mapping.

**Example 6.** Consider $\Pi_2$ in Example 2. Let $I = \{p(-1), p(1)\}$ be an interpretation; then $\neg p(2)$ is in $\neg I^-$. $I$ is an FLP answer set of $\Pi_2$ since it is a minimal model of the FLP reduct $f\Pi_2^I = \{r_2, r_3\}$, where $r_2$ is $p(-1) \leftarrow \neg p(-1) \vee p(1) \vee p(2)$ and $r_3$ is $p(1) \leftarrow p(-1)$. $I$ has in total three partitionings: $\langle \neg I^-, \{p(-1)\}, \{p(1)\}\rangle$, $\langle \neg I^-, \{p(1)\}, \{p(-1)\}\rangle$ and $\langle \neg I^-, \{p(-1), p(1)\}\rangle$. It turns out that none of these partitionings is a level mapping as in Definition 4. Therefore, the FLP answer set $I$ has no level mapping.

A way to overcome the circular justification problem of FLP answer sets is thus *to enhance the FLP answer set semantics with level mappings for the FLP reduct*, treating the reduct as a set of rules instead of a set of classical implications. To this end, let us first review how the standard answer set semantics induces a level mapping for answer sets of a normal logic program.

The seminal definition of an answer set $I$ of a normal logic program $\Pi$ involves three steps [33]:

1. Eliminate all rules from $ground(\Pi)$ whose bodies contain a negative literal that is not satisfied by $I$.
2. Eliminate from the bodies of the remaining rules in $ground(\Pi)$ all negative literals. Note that these negative literals are satisfied by $I$ and thus belong to $\neg I^-$.
   The rule set resulting from the two steps is called the *Gelfond–Lifschitz reduct* of $\Pi$ w.r.t. $I$ and is denoted by $\Pi^I$; note that $\Pi^I$ is a positive logic program.
3. Check whether $I$ is the least model of $\Pi^I$. To this end, compute the latter as the least fixpoint $lfp(T_{\Pi^I}(\emptyset))$ of the operator $T_{\Pi^I}$ by iteration via the sequence $\langle T^i_{\Pi^I}(\emptyset)\rangle_{i=0}^{\infty}$, where $T^0_{\Pi^I}(\emptyset) = \emptyset$ and for $i \ge 0$, $T^{i+1}_{\Pi^I}(\emptyset) = T_{\Pi^I}(T^i_{\Pi^I}(\emptyset))$. Here $T_P(S)$, where $P$ is a positive logic program and $S$ is a set of ground atoms, is the van Emden–Kowalski one-step provability operator [64] defined by

$$T_P(S) = \{head(r) \mid r \in ground(P) \text{ and } body(r) \text{ is satisfied by } S\}.$$

Summarizing, an interpretation $I$ is an answer set of a normal logic program $\Pi$ under the standard answer set semantics if $I = lfp(T_{\Pi^I}(\emptyset))$.

This process naturally induces a level mapping on each answer set $I$, which assigns a level $k > 0$ to each $A \in I$ if $A \in T^k_{\Pi^I}(\emptyset)$ but $A \notin T^{k-1}_{\Pi^I}(\emptyset)$. Let $S = \langle S_0, S_1, \cdots, S_m\rangle$ be a partitioning of $I$, where $S_0 = \neg I^-$, for $i > 0$ $S_i = T^i_{\Pi^I}(\emptyset) \setminus T^{i-1}_{\Pi^I}(\emptyset)$, and $\bigcup_{1\le i\le m} S_i = lfp(T_{\Pi^I}(\emptyset)) = I$. Note that for any $k > 0$, $\bigcup_{1\le i\le k} S_i = T^k_{\Pi^I}(\emptyset)$. Then, for every $A \in I$ at level $k > 0$, there exists some rule $r \in ground(\Pi)$ with $head(r) = A$ such that all negative literals in $body(r)$ are in $\neg I^-$ and all positive literals in $body(r)$ are in $T^{k-1}_{\Pi^I}(\emptyset)$. This means $body(r)$ is true in $\bigcup_{0\le i\le k-1} S_i$. By Definition 4, $S$ is a level mapping of $I$. This leads to the following immediate result.

**Theorem 1.** *Under the standard answer set semantics, every answer set of a normal logic program has a level mapping as in Definition 4 and thus is free of circular justifications.*

For logic programs with first-order formulas, the above three step definition of answer sets is not applicable in general, since rule heads and bodies of such general logic programs can be arbitrary first-order formulas. For example, consider again the logic program $\Pi = \{A \leftarrow A \vee \neg A\}$, where $A$ is a ground atom. Since the rule body $A \vee \neg A$ is a tautology that is always true in first-order logic, $I = \{A\}$ is supposed to be an answer set of $\Pi$. Apparently, this answer set cannot be obtained following literally the three steps above.

In order to handle arbitrary first-order formulas in rule heads and bodies of a general logic program, we propose to extend the first two steps of the Gelfond–Lifschitz definition of answer sets as follows:

1. Instead of eliminating all rules whose bodies contain some negative literal that is not satisfied by $I$, we extend the first step by eliminating from $ground(\Pi)$ all rules whose bodies are not satisfied by $I$. This yields the FLP reduct $f\Pi^I$.
2. Instead of directly eliminating from $f\Pi^I$ all negative literals that appear in $\neg I^-$, we adapt the second step to first-order formulas by adding the negative literals in $\neg I^-$ as constraints on $f\Pi^I$.

To extend the third step of the Gelfond–Lifschitz definition to first-order formulas, we first extend the van Emden–Kowalski operator $T_P(S)$, which is applicable only to a positive logic program $P$ parameterized with a set $S$ of ground atoms, to a new operator $T_\Pi(O, N)$ that is applicable to a general logic program $\Pi$ parameterized with two first-order theories $O$ and $N$. As shall be seen below, the first parameter $O$ of the extended operator $T_\Pi(O, N)$ is used to express a set of rule heads in $ground(\Pi)$, while the second parameter $N$ used to express some constrains. Intuitively, by applying $T_\Pi(O, N)$ we infer all heads of rules from $ground(\Pi)$ whose bodies are true in $O$ under the constraints $N$, i.e., $O \cup N \models body(r)$. Formally, we have

**Definition 8.** Let $\Pi$ be a logic program, and $O$ and $N$ be two first-order theories. Define the following one-step provability operator:

$$T_\Pi(O, N) = \{head(r) \mid r \in ground(\Pi) \text{ and } O \cup N \models body(r)\}.$$

When the constraints $N$ are fixed, the entailment relation $\models$ is monotone in $O$, so $T_\Pi(O, N)$ is monotone w.r.t. $O$. That is, for any first-order theories $O_1, O_2$ with $O_1 \subseteq O_2$, $T_\Pi(O_1, N) \subseteq T_\Pi(O_2, N)$. Therefore, the sequence $\langle T^i_\Pi(\emptyset, N)\rangle^\infty_{i=0}$, where $T^0_\Pi(\emptyset, N) = \emptyset$ and for $i \ge 0$ $T^{i+1}_\Pi(\emptyset, N) = T_\Pi(T^i_\Pi(\emptyset, N), N)$, will converge to a least fixpoint, denoted $lfp(T_\Pi(\emptyset, N))$.

Thus, when replacing the constraints $N$ with $\neg I^-$, we obtain a fixpoint $lfp(T_\Pi(\emptyset, \neg I^-))$; and when further replacing $\Pi$ with the FLP reduct $f\Pi^I$, we obtain a fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$.

With the new operator $T_\Pi(O, N)$, we then extend the third step of the Gelfond–Lifschitz definition of answer sets to first-order formulas as follows:

3. Compute the least fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ of the operator $T_{f\Pi^I}$ via the sequence $\langle T^i_{f\Pi^I}(\emptyset, \neg I^-)\rangle^\infty_{i=0}$, where $T^0_{f\Pi^I}(\emptyset, \neg I^-) = \emptyset$ and for $i \ge 0$, $T^{i+1}_{f\Pi^I}(\emptyset, \neg I^-) = T_{f\Pi^I}(T^i_{f\Pi^I}(\emptyset, \neg I^-), \neg I^-)$.

The following example illustrates the above extension to the Gelfond–Lifschitz three step definition.

**Example 7.** Consider again the logic program $\Pi = \{r_1, r_2\}$ from Example 5, where $r_1$ is $A \vee (\neg B \wedge C) \leftarrow \neg A \wedge (\neg C \vee C)$ and $r_2$ is $D \leftarrow C$. Let $I = \{C, D\}$ be an interpretation of $\Pi$; then $\neg A$, $\neg B$ are in $\neg I^-$. Since $I$ satisfies the bodies of the two rules, the FLP reduct $f\Pi^I$ of $\Pi$ w.r.t. $I$ is $\Pi$ itself. Let $T^0_{f\Pi^I}(\emptyset, \neg I^-) = \emptyset$. Since the body of $r_1$ is entailed by $T^0_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^-$, $T^1_{f\Pi^I}(\emptyset, \neg I^-) = \{A \vee (\neg B \wedge C)\}$. Since the bodies of $r_1$ and $r_2$ are entailed by $T^1_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^-$, $T^2_{f\Pi^I}(\emptyset, \neg I^-) = \{A \vee (\neg B \wedge C), D\}$. It is easy to check that $T^3_{f\Pi^I}(\emptyset, \neg I^-) = T^2_{f\Pi^I}(\emptyset, \neg I^-)$; thus we have the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = \{A \vee (\neg B \wedge C), D\}$.

The first important result about the extended van Emden–Kowalski operator $T_\Pi(O, N)$ is that when $I$ is a model of a logic program $\Pi$, applying the operator to $\Pi$ and $f\Pi^I$ derives the same rule heads. This justifies the above first step extension to the Gelfond–Lifschitz answer set definition, where the FLP reduct $f\Pi^I$ is used as a simplified form of $\Pi$.

**Theorem 2.** *Let $I$ be a model of a logic program $\Pi$. For every $i \geq 0$, $T^i_\Pi(\emptyset, \neg I^-) = T^i_{f\Pi^I}(\emptyset, \neg I^-)$ and thus $lfp(T_\Pi(\emptyset, \neg I^-)) = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$.*

The proof of this theorem is based on the following lemma.

**Lemma 1.** *If $I$ is a model of a logic program $\Pi$, then for every $i \geq 0$, $I$ is a model of $T^i_\Pi(\emptyset, \neg I^-)$.*

The next result shows that $T_\Pi(O, N)$ is a proper generalization of the original van Emden–Kowalski operator $T_P(S)$.

**Theorem 3.** *Let $I$ be a model of a normal logic program $\Pi$ and $\Pi^I$ be the Gelfond–Lifschitz reduct of $\Pi$. Then for every $i \geq 0$, $T^i_{\Pi^I}(\emptyset) = T^i_\Pi(\emptyset, \neg I^-)$, and thus $lfp(T_{\Pi^I}(\emptyset)) = lfp(T_\Pi(\emptyset, \neg I^-))$.*

The following characterization of the standard answer set semantics follows immediately from Theorems 2 and 3.

**Corollary 1.** *A model $I$ of a normal logic program $\Pi$ is an answer set under the standard answer set semantics if and only if $I = lfp(T_{\Pi^I}(\emptyset))$ if and only if $I = lfp(T_\Pi(\emptyset, \neg I^-))$ if and only if $I = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$.*

The conditions listed in Corollary 1 for an answer set of a normal logic program do not apply to a logic program with first-order formulas, because in the latter case the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ (resp. $lfp(T_\Pi(\emptyset, \neg I^-))$) would be a first-order theory instead of a set of ground atoms (e.g., see Example 7). However, these conditions suggest that instead of requiring each $A \in I$ be included in $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$, *answer sets of a logic program $\Pi$ with first-order formulas can be defined by requiring that each $A \in I$ is true in the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ under the constraints $\neg I^-$*; i.e., for each $A \in I$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$. This leads to the following principal definition.

**Definition 9** *(well-justified answer set).* Let $I$ be a model of a logic program $\Pi$. Then $I$ is an *answer set* of $\Pi$ if $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$ for every $A \in I$.

**Example 8.** In Example 7, $I = \{C, D\}$ is a model of $\Pi$ and the fixpoint is $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = \{A \vee (\neg B \wedge C), D\}$. Obviously, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models D$. Since $\neg A, \neg B$ are in $\neg I^-$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models C$. Thus, $I$ is an answer set of $\Pi$ under Definition 9.

**Example 9.** In Example 2, $I = \{p(-1), p(1)\}$ is an FLP answer set of $\Pi_2$ and $f\Pi_2^I = \{r_2, r_3\}$. Since neither of the bodies of $r_2$ and $r_3$ is entailed by $\neg I^-$, $T^1_{f\Pi_2^I}(\emptyset, \neg I^-) = T^0_{f\Pi_2^I}(\emptyset, \neg I^-) = \emptyset$; thus we have the fixpoint $lfp(T_{f\Pi_2^I}(\emptyset, \neg I^-)) = \emptyset$. Neither $p(-1)$ nor $p(1)$ can be proved true in $lfp(T_{f\Pi_2^I}(\emptyset, \neg I^-))$ under the constraints $\neg I^-$; therefore, $I$ is not an answer set of $\Pi_2$ under Definition 9.

By Theorem 2, it is immediate that a model $I$ of a logic program $\Pi$ is an answer set of $\Pi$ if and only if for each $A \in I$, $lfp(T_\Pi(\emptyset, \neg I^-)) \cup \neg I^- \models A$. The next result shows such answer sets are minimal models.

**Theorem 4.** *Every answer set $I$ of a logic program $\Pi$ is a minimal model of $\Pi$ and furthermore, a minimal model of the FLP reduct $f\Pi^I$.*

It is immediate from Theorem 4 and Definition 2:

**Corollary 2.** *Every answer set $I$ of a logic program $\Pi$ is an FLP answer set of $\Pi$.*

Like the standard answer set semantics, answer sets of Definition 9 induce a level mapping from the FLP reduct $f\Pi^I$ via the sequence $\langle T^i_{f\Pi^I}(\emptyset, \neg I^-)\rangle_{i=0}^{\infty}$. The following result extends Theorem 1 from normal logic programs to logic programs with first-order formulas.

**Theorem 5.** *Every answer set of a logic program has a level mapping as in Definition 7 and thus is free of circular justifications.*

By Proposition 1 the following corollary is immediate.

**Corollary 3.** *For a logic program $\Pi$ whose rule heads are atoms, every answer set of $\Pi$ has a level mapping as in Definition 4.*

Hence, answer sets of Definition 9 are FLP answer sets (by Corollary 2) enhanced with a level mapping, which makes them free of circular justifications (by Theorem 5). For this reason, we call answer sets of Definition 9 *well-justified FLP answer sets* and the associated semantics the *well-justified FLP answer set semantics*.

In the well-justified FLP answer set semantics, we treat an FLP reduct $f\Pi^I$ as *if-then* rules $H \leftarrow B$, instead of classical implications $B \supset H$, by iteratively applying these rules to compute the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$. This process induces a level mapping as in Definition 7 on well-justified FLP answer sets (Theorem 5). In contrast, the FLP answer set semantics (Definition 2) identifies $f\Pi^I$ with a set of classical implications by computing minimal models of these implications; this process does not induce a level mapping on FLP answer sets. The following example further illustrates the essential difference.

**Example 10.** Consider the following two logic programs:

$$\begin{aligned}
\Pi: \quad & p \vee q. && r_1\\
& p \leftarrow q. && r_2\\
& q \leftarrow p. && r_3\\
\Pi': \quad & p \vee q. && r'_1\\
& q \supset p. && r'_2\\
& p \supset q. && r'_3
\end{aligned}$$

Note that $\Pi$ is a logic program with first-order formulas, instead of a *disjunctive logic program* as introduced in [34], since $p \vee q$ is a classical disjunction instead of an *epistemic disjunction*.[2]

$I = \{p, q\}$ is the only minimal model of the two programs. $f\Pi^I = \Pi$ and $f\Pi'^I = \Pi'$. Under the FLP answer set semantics, $\Pi$ is identified with $\Pi'$ and $I$ is an FLP answer set of $\Pi$ if and only if it is a minimal model of $\Pi'$. Consequently, $I$ is both an FLP answer set of $\Pi$ and of $\Pi'$.

Under the well-justified FLP answer set semantics, however, the two logic programs function rather differently. For $\Pi$, $I = \{p, q\}$ is a well-justified FLP answer set if and only if the two rules $r_2, r_3$ are applicable if and only if one of the two rule bodies is entailed by $r_1$, i.e. $p \vee q \models q$ or $p \vee q \models p$. This condition, if satisfied, will induce a level mapping on $I$. The condition is precisely conveyed by the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = \{p \vee q\}$ which entails neither $p$ nor $q$, meaning that $I = \{p, q\}$ is not a well-justified FLP answer set of $\Pi$. For $\Pi'$, since all rules $r'_1, r'_2, r'_3$ are first-order formulas, $I = \{p, q\}$ is a well-justified FLP answer set if and only if $\Pi' \models q$ and $\Pi' \models p$. This condition is precisely captured by the fixpoint $lfp(T_{f\Pi'^I}(\emptyset, \neg I^-)) = \Pi'$ which entails both $p$ and $q$, meaning that $I = \{p, q\}$ is a well-justified FLP answer set of $\Pi'$.

The following result shows that the well-justified FLP answer set semantics excludes only those FLP answer sets that have no level mapping.

**Theorem 6.** *Let $I$ be an FLP answer set of a logic program $\Pi$. $I$ is a well-justified FLP answer set of $\Pi$ if and only if it has a level mapping as in Definition 7.*

The proof of this theorem is based on the following lemma.

**Lemma 2.** *Let $I$ be a model of a logic program $\Pi$ and $S' = \langle S'_1, \cdots, S'_m \rangle$ be a total level mapping of rule heads of $\Pi$ w.r.t. $I$. Then $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = \bigcup_{1 \le i \le m} S'_i$.*

Note that for a model $I$ of a normal logic program $\Pi$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ is a set of ground atoms; by Definition 9 $I$ is a well-justified FLP answer set of $\Pi$ if and only if $I = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$. Then, by Corollary 1 the well-justified FLP answer set semantics coincides with the standard answer set semantics and thus coincides with the FLP answer set semantics. The following result shows that a similar coincidence holds for logic programs whose rule bodies are all empty.

**Theorem 7.** *Let $\Pi$ be a logic program whose rule bodies are all empty. Then a model $I$ of $\Pi$ is a well-justified FLP answer set of $\Pi$ if and only if $I$ is a minimal model of $\Pi$ if and only if $I$ is an FLP answer set of $\Pi$.*

[2] Epistemic disjunctions are usually expressed using the epistemic operator | in the literature. A classical disjunction $A \vee \neg A$ is a tautology, but an epistemic disjunction $A \mid \neg A$ is not a tautology since it does not follow the law of the excluded middle (see [31] for detailed explanations).

## 5. Well-justified FLP answer sets for logic programs with aggregates

We first extend the first-order language $\mathcal{L}_\Sigma$ defined in Section 2 to aggregate functions, such as COUNT(), SUM(), TIMES(), MIN() and MAX(). An aggregate function maps a finite set of elements in a domain to a value in a range. For simplicity, we assume the range of each aggregate function is a set of (positive and negative) integers and the signature $\Sigma$ of $\mathcal{L}_\Sigma$ contains all integers (as constants).

Aggregates involve comparison operators, such as $=, \leq, \geq, <, >$, etc., which define binary relations over integers. We assume that aggregate function symbols and comparison operators are not included in $\Sigma$.

**Aggregate atoms** An *aggregate atom* (aggregate for short) in $\mathcal{L}_\Sigma$ is of the form

$$\mathrm{OP}\big\langle (D, X) : F(X) \big\rangle \succeq b$$

where (1) OP is an aggregate function symbol; (2) $D \subseteq \mathcal{N}_\Sigma$ is the domain of OP; (3) $X$ is an *aggregate variable*, which takes on values from $D$; (4) $F(X)$ is a first-order formula; (5) $\succeq$ is a comparison operator; and (6) $b$ is an integer.

Note that only one aggregate variable $X$ is used in an aggregate atom; it can easily be extended to a list of aggregate variables. $X$ is bounded by the domain $D$; we may omit $D$ when it is clear from context. $F(X)$ may contain the aggregate variable $X$ or other variables; if $F(X)$ contains no variable, the aggregate atom is called a *ground aggregate atom*.

**Definition 10.** A *logic program $\Pi$ with aggregate atoms* is a finite set of rules of the form $H \leftarrow B$, where $H$ and $B$ are first-order formulas extended with aggregate atoms.

The grounding $\mathit{ground}(\Pi)$ of a logic program $\Pi$ with aggregate atoms is obtained by replacing every free variable except aggregate variables in $\Pi$ with a constant in $\mathcal{C}_\Pi$. We assume that the domain $D$ of each aggregate function consists of constants from $\mathcal{C}_\Pi$, and for each aggregate atom $\mathrm{OP}\langle (D, X) : F(X) \rangle \succeq b$ in $\mathit{ground}(\Pi)$, except $X$ all variables in $F(X)$ are in the scope of a quantifier in $F(X)$.

For an interpretation $I$, satisfaction of an aggregate atom $A = \mathrm{OP}\langle (D, X) : F(X) \rangle \succeq b$ in $\mathit{ground}(\Pi)$ w.r.t. $I$ is defined as follows. Let

$$S_A^I = \big\{ a \mid a \in D \text{ such that } I \text{ satisfies } F(a) \big\}.$$

Then $I$ *satisfies* $A$ if $\mathrm{OP}(S_A^I) \succeq b$ holds, and $I$ *satisfies* $\neg A$ if $I$ does not satisfy $A$.[3]

Now that the satisfaction relation of $\mathcal{L}_\Sigma$ is extended to aggregate atoms, the entailment relation $\models$ is extended accordingly. Thus the operator $T_\Pi(O, N)$ (Definition 8) can be applied to logic programs with aggregate atoms in the same way as logic programs with first-order formulas, and Definition 9 directly extends to such logic programs, i.e., a model $I$ of a logic program $\Pi$ with aggregate atoms is a *well-justified FLP answer set* of $\Pi$ if for every $A \in I$, $\mathit{lfp}(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$. All results (Theorems, Lemmas, and Corollaries) obtained in Section 4 for logic programs with first-order formulas hold with the same proofs for logic programs with aggregate atoms. By Corollary 2 and Theorem 6, well-justified FLP answer sets of logic programs with aggregate atoms are FLP answer sets enhanced with a level mapping as in Definition 7 and thus are free of circular justifications.

**Example 11.** Consider the following logic program with aggregate atoms (borrowed from [4]):

$$\begin{aligned} \Pi_3: \quad & p(2) \leftarrow \neg \mathrm{SUM}\big\langle \big(\{-1, 1, 2\}, X\big) : p(X) \big\rangle < 2. && r_1 \\ & p(-1) \leftarrow \mathrm{SUM}\big\langle \big(\{-1, 1, 2\}, X\big) : p(X) \big\rangle \geq 0. && r_2 \\ & p(1) \leftarrow p(-1). && r_3 \end{aligned}$$

$\mathrm{SUM}\langle (\{-1, 1, 2\}, X) : p(X) \rangle$ is an aggregate function, where $X$ is an aggregate variable with the domain $\{-1, 1, 2\}$, which sums up all $X$ in the domain such that $p(X)$ is true. For simplicity, let $A_1$ and $A_2$ represent the two aggregate atoms $\mathrm{SUM}\langle (\{-1, 1, 2\}, X) : p(X) \rangle < 2$ and $\mathrm{SUM}\langle (\{-1, 1, 2\}, X) : p(X) \rangle \geq 0$ respectively. For an interpretation $I = \{p(-1), p(1)\}$,

$$S_{A_1}^I = S_{A_2}^I = \big\{ a \mid a \in \{-1, 1, 2\} \text{ such that } I \text{ satisfies } p(a) \big\} = \{-1, 1\}$$

and

$$\mathrm{SUM}\big(S_{A_1}^I\big) = \mathrm{SUM}\big(S_{A_2}^I\big) = \mathrm{SUM}\big(\{-1, 1\}\big) = 0,$$

so $I$ satisfies $A_1$ and $A_2$. Then the FLP reduct is $f\Pi_3^I = \{r_2, r_3\}$.

[3] Note that aggregates over multisets can be readily supported using a list $X = X_1, \ldots, X_n$ of aggregate variables and defining $\mathrm{OP}(S_A^I)$ to work on the first component of the tuples in $S_A^I = \{(a_1, \ldots, a_n) \in D^n \mid I \text{ satisfies } F(a_1, \ldots, a_n)\}$ (as in the DLV system).

We have $lfp(T_{f\Pi_3^I}(\emptyset, \neg I^-)) = \emptyset$. Since $lfp(T_{f\Pi_3^I}(\emptyset, \neg I^-)) \cup \neg I^- \not\models p(-1)$ and $lfp(T_{f\Pi_3^I}(\emptyset, \neg I^-)) \cup \neg I^- \not\models p(1)$, $I = \{p(-1), p(1)\}$ is not a well-justified FLP answer set of $\Pi_3$.

As $I = \{p(-1), p(1)\}$ is a minimal model of $f\Pi_3^I$, it is an FLP answer set of $\Pi_3$ (by Definition 2). This FLP answer set has a circular justification similar to that of $\Pi_2$ in Example 2. Observe that $\Pi_3$ represents the same knowledge as $\Pi_2$ because the aggregate atom $\neg\text{SUM}\langle(\{-1, 1, 2\}, X) : p(X)\rangle < 2$ in $\Pi_3$ can be interpreted as the first-order formula $p(2) \wedge (\neg p(-1) \vee p(1))$ in $\Pi_2$, while $\text{SUM}\langle(\{-1, 1, 2\}, X) : p(X)\rangle \geq 0$ interpreted as $\neg p(-1) \vee p(1) \vee p(2)$.

**Example 12.** Consider again the logic program $\Pi_1$ in Example 1. The aggregate function $\text{SUM}\langle X : p(X)\rangle$ has an aggregate variable $X$ whose domain is implicitly assumed to be $\{-1, 1, 2\}$. Let $A$ represent the aggregate atom $\text{SUM}\langle X : p(X)\rangle \geq 1$ in $\Pi_1$. For an interpretation $I = \{p(1), p(-1), p(2)\}$,

$$S_A^I = \{a \mid a \in \{-1, 1, 2\} \text{ such that } I \text{ satisfies } p(a)\} = \{1, -1, 2\}$$

and $\text{SUM}(S_A^I) = 2$, so $I$ satisfies $A$ and the FLP reduct of $\Pi_1$ w.r.t. $I$ is $\Pi_1$ itself. $I$ is an FLP answer set of $\Pi_1$, but it is not a well-justified FLP answer set. The fixpoint is $lfp(T_{f\Pi_1^I}(\emptyset, \neg I^-)) = \{p(1)\}$; neither $p(-1)$ nor $p(2)$ in $I$ is true in the fixpoint under the constraints $\neg I^-$.

Many aggregate atoms can be represented in an abstract form as *abstract constraint atoms* (or *c-atoms*) [48]. Next we further extend the first-order language $\mathcal{L}_\Sigma$ to encompass c-atoms.

**Constraint atoms** A *c-atom* is a pair $(V, C)$, where $V$, the domain of the c-atom, is a finite subset of $\mathcal{H}_\Sigma$, and $C$, the admissible solutions of the c-atom, is a collection of sets of atoms in $V$. For instance, the aggregate atom $\text{SUM}\langle(\{-1, 1, 2\}, X) : p(X)\rangle < 2$ in $\Pi_3$ can be represented as a c-atom $(V, C)$, where $V = \{p(-1), p(1), p(2)\}$ and $C = \{\emptyset, \{p(-1)\}, \{p(1)\}, \{p(-1), p(1)\}, \{p(-1), p(2)\}\}$. The first solution $\emptyset$ in $C$ means that none of $p(-1)$, $p(1)$, $p(2)$ in the domain is true, while the last solution $\{p(-1), p(2)\}$ in $C$ means that only $p(-1)$ and $p(2)$ are true. Clearly, in all such cases $\text{SUM}\langle(\{-1, 1, 2\}, X) : p(X)\rangle < 2$ holds.

**Definition 11.** A *logic program $\Pi$ with c-atoms* is a finite set of rules of the form $H \leftarrow B$, where $H$ and $B$ are first-order formulas extended with c-atoms.

An interpretation $I$ *satisfies* a c-atom $(V, C)$ if $I \cap V \in C$; $I$ *satisfies* $\neg(V, C)$ if $I$ does not satisfy $(V, C)$; the entailment relation $\models$ and the operator $T_\Pi(O, N)$ extend accordingly to logic programs with c-atoms. All definitions and results from above for aggregate atoms, including the notion of well-justified FLP answer sets, carry over to logic programs with c-atoms.

As far as we can determine, the well-justified FLP answer set semantics is the first answer set semantics that is free of circular justifications for logic programs with first-order formulas as well as aggregate atoms or c-atoms. Two notable exceptions are the three-valued fixpoint semantics of Pelov et al. [53] for the class of logic programs whose rule heads are atoms and the conditional satisfaction-based semantics of Son et al. [60] for a special class of logic programs with aggregate atoms resp. c-atoms called positive basic logic programs. (The latter semantics is essentially a reformulation of the former.) Since answer sets under the two semantics are free of circular justifications, we next study their relationship with the well-justified FLP answer set semantics.

### 5.1. Relation to the three-valued fixpoint semantics of Pelov et al. [53]

For simplicity of presentation, in this subsection we consider only propositional logic programs and disregard aggregate atoms, which will be addressed in Section 5.2.

For a logic program $\Pi$ whose rule heads are atoms, in [16,53] a three-valued fixpoint semantics was introduced based on a three-valued fixpoint operator $\Phi_\Pi$. Answer sets under this semantics are called *two-valued stable models*.

A three-valued (Herbrand) interpretation of $\Pi$ is $\hat{I} = (I_1, I_2)$, where $I_1 \subseteq I_2 \subseteq \mathcal{HB}_\Pi$. Intuitively, atoms in $I_1$ are assigned the truth value **t**, atoms in $I_2 \setminus I_1$ assigned **u**, and atoms in $\mathcal{HB}_\Pi \setminus I_2$ assigned **f**. These truth values are ordered by the *truth order* $\leq_t$ with $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{t}$. Negation on these truth values is defined as $\neg\mathbf{f} = \mathbf{t}$, $\neg\mathbf{u} = \mathbf{u}$ and $\neg\mathbf{t} = \mathbf{f}$. The truth value of a propositional formula $F$ under $\hat{I}$, denoted $\hat{I}(F)$, is defined recursively as follows:

$$\hat{I}(F) = \begin{cases} \mathbf{t} \text{ (resp. } \mathbf{u} \text{ and } \mathbf{f}) & \text{if } F \text{ is in } I_1 \text{ (resp. } I_2 \setminus I_1 \text{ and } \mathcal{HB}_\Pi \setminus I_2) \\ \min_{\leq_t}\{\hat{I}(F_1), \hat{I}(F_2)\} & \text{if } F = F_1 \wedge F_2 \\ \max_{\leq_t}\{\hat{I}(F_1), \hat{I}(F_2)\} & \text{if } F = F_1 \vee F_2 \\ \neg\hat{I}(F_1) & \text{if } F = \neg F_1 \end{cases}$$

Note that $F_1 \supset F_2$ is an abbreviation for $\neg F_1 \vee F_2$. Then $\hat{I}$ *satisfies* $F$ if $\hat{I}(F) = \mathbf{t}$.

Given a three-valued interpretation $\hat{I} = (I_1, I_2)$, the three-valued operator $\Phi_\Pi(I_1, I_2) = (I_1', I_2')$ is defined such that

$$I'_1 = \{head(r) \mid r \in \Pi \text{ and } \hat{I}(body(r)) = \mathbf{t}\}, \quad \text{and}$$
$$I'_2 = \{head(r) \mid r \in \Pi \text{ and } \hat{I}(body(r)) = \mathbf{t} \text{ or } \hat{I}(body(r)) = \mathbf{u}\}.$$

Let $\Phi^1_\Pi(I_1, I_2)$ denote the first element of $\Phi_\Pi(I_1, I_2)$, i.e. $I'_1$, and $\Phi^2_\Pi(I_1, I_2)$ denote the second element $I'_2$. When $I_2$ is fixed, we compute a sequence $x_0 = \emptyset, x_1 = \Phi^1_\Pi(x_0, I_2), \cdots, x_{i+1} = \Phi^1_\Pi(x_i, I_2), \cdots$, until a fixpoint, denoted $St^{\downarrow}_\Phi(I_2)$, is reached. Similarly, when $I_1$ is fixed, we compute a sequence $x_0 = I_1, x_1 = \Phi^2_\Pi(I_1, x_0), \cdots, x_{i+1} = \Phi^2_\Pi(I_1, x_i), \cdots$, until a fixpoint $St^{\uparrow}_\Phi(I_1)$ is reached. The *stable revision operator* $St_\Phi$ on $\hat{I} = (I_1, I_2)$ is defined as

$$St_\Phi(I_1, I_2) = \big(St^{\downarrow}_\Phi(I_2), St^{\uparrow}_\Phi(I_1)\big).$$

By iteratively applying $St_\Phi$ such that $St^0_\Phi(I_1, I_2) = (I_1, I_2)$ and for $i > 0$, $St^i_\Phi(I_1, I_2) = St_\Phi St^{i-1}_\Phi(I_1, I_2)$, we obtain a fixpoint $lfp(St_\Phi(I_1, I_2))$ of $St_\Phi$.

The three-valued fixpoint semantics of Denecker et al. [16] and of Pelov et al. [53] is then defined in terms of the fixpoint $lfp(St_\Phi)$. Let $I$ be a two-valued model of $\Pi$ (as defined in Section 2.2). $I$ is called a *two-valued stable model* of $\Pi$ if $lfp(St_\Phi(I, I)) = (I, I)$.

We observe that there are at least three significant differences between the three-valued fixpoint semantics and the well-justified FLP answer set semantics. First, the three-valued fixpoint semantics is defined over three-valued interpretations, while the well-justified FLP answer set semantics is defined over two-valued interpretations. Second, the three-valued fixpoint semantics is applicable only to logic programs whose rule heads are atoms, while the well-justified FLP answer set semantics applies to logic programs whose rule heads are arbitrary first-order formulas. Third, as shown below the three-valued fixpoint semantics is more conservative than the well-justified FLP answer set semantics in the sense that two-valued stable models of the three-valued fixpoint semantics are well-justified FLP answer sets, which by Corollary 2 are also FLP answer sets, but the converse does not hold.

**Theorem 8.** *Let $\Pi$ be a propositional logic program whose rule heads are atoms and let $I$ be a two-valued stable model of $\Pi$ under the three-valued fixpoint semantics. Then $I$ is also a well-justified FLP answer set of $\Pi$.*

However, a well-justified FLP answer set is not necessarily a two-valued stable model. As an example, consider the logic program $\Pi = \{p \leftarrow \neg p \vee p\}$. $I = \{p\}$ is a two-valued model of $\Pi$. Since the rule body $\neg p \vee p$ is a tautology in classical logic, $I$ is a well-justified FLP answer set of $\Pi$. However, $I$ is not a two-valued stable model under the three-valued fixpoint semantics since $lfp(St_\Phi(\{p\}, \{p\})) = (\emptyset, \{p\})$.

Recall that the well-justified FLP answer set semantics excludes only those FLP answer sets that have no level mapping (Theorem 6). The above example shows that the three-valued fixpoint semantics eliminates some FLP answer sets that have level mappings.

### 5.2. Relation to conditional satisfaction-based semantics of Son et al. [60]

Son et al. [60] defined an answer set semantics for a special class of logic programs with c-atoms called positive basic logic programs.

**Definition 12.** A *positive basic logic program* is a finite set of function and equality free rules of the form $A \leftarrow A_1 \wedge \cdots \wedge A_m$, where $A$ is a ground atom and each $A_i$ is a c-atom.

Note that any ground atom $A$ can be represented as an *elementary c-atom* $(\{A\}, \{\{A\}\})$, and $\neg A$ represented as a c-atom $(\{A\}, \{\emptyset\})$. For any c-atom $(V, C)$, $\neg(V, C)$ can be represented as a c-atom $(V, 2^V \setminus C)$, where $2^V$ is the power set of $V$. Therefore, for any normal logic program $\Pi$ with c-atoms, its grounding $ground(\Pi)$ can be represented in this way by an equivalent positive basic logic program.

Son et al. [60] defined answer sets for positive basic logic programs based on a notion of conditional satisfaction. Let $R$ and $S$ be two sets of ground atoms with $R \subseteq S$. For a c-atom $A = (V, C)$, $R$ *conditionally satisfies* $A$ w.r.t. $S$, denoted $R \models_S A$, if for every $F$ with $R \cap V \subseteq F \subseteq S \cap V$, $F \in C$; for a ground atom $A$, $R \models_S A$ if $R \models_S (\{A\}, \{\{A\}\})$.

For a positive basic logic program $\Pi$, define the following one-step provability operator:

$$\Gamma_\Pi(R, S) = \{A \mid A \leftarrow body(r) \in \Pi \text{ and } R \models_S body(r)\}.$$

Son et al. proved that if the second argument $S$ of $\Gamma_\Pi(R, S)$ is a model of $\Pi$, then the sequence $\langle \Gamma^i_\Pi(\emptyset, S)\rangle^\infty_{i=0}$, where $\Gamma^0_\Pi(\emptyset, S) = \emptyset$ and for $i > 0$ $\Gamma^i_\Pi(\emptyset, S) = \Gamma_\Pi(\Gamma^{i-1}_\Pi(\emptyset, S), S)$, is monotone and will converge to a fixpoint $lfp(\Gamma_\Pi(\emptyset, S))$. Based on this, a model $I$ of $\Pi$ is a *conditional satisfaction based answer set* of $\Pi$ if $I = lfp(\Gamma_\Pi(\emptyset, I))$.

It is not hard to see that conditional satisfaction of a c-atom is closely related to our notion of entailment as follows.

**Lemma 3.** *Let $I$ be a model of a positive basic logic program $\Pi$. For every $R \subseteq I$ and c-atom $A$ occurring in $\Pi$, $R \models_I A$ if and only if $R \cup \neg I^- \models A$.*

Consequently, the stages $lfp(T_{\Pi}(\emptyset, \neg I^-))$ and $lfp(\Gamma_{\Pi}(\emptyset, I))$ of the operators $T_{\Pi}$ and $\Gamma_{\Pi}$, respectively, coincide for models $I$ of $\Pi$; we thus obtain:

**Theorem 9.** *A model of a positive basic logic program is a well-justified FLP answer set if and only if it is a conditional satisfaction based answer set.*

As positive basic logic programs are a class of logic programs with c-atoms, this result suggests that (1) the well-justified FLP answer set semantics is a proper extension of the conditional satisfaction-based answer set semantics, and (2) answer sets according to the latter are free of circular justifications.

For positive basic logic programs, Son et al. [60] showed that the conditional satisfaction-based answer set semantics agrees with the three-valued fixpoint semantics of Denecker et al. [16] and of Pelov et al. [53]. By Theorem 9, for such logic programs the well-justified FLP answer set semantics also agrees with the three-valued fixpoint semantics and thus can be also regarded as an extension of the latter to logic programs with c-atoms.[4]

## 6. Well-justified FLP answer sets for description logic programs

In principle, the above method of defining well-justified FLP answer sets can be applied to different types of logic programs, provided that the satisfaction relation of $\mathcal{L}_{\Sigma}$ is extended to those logic programs. As another important application, in this section we define well-justified FLP answer sets for dl-programs [25,24]. Other well-known types of logic programs, such as HEX-programs [25], tightly coupled dl-programs [46], and modular logic programs [12], can be handled in a similar way.

A dl-program can be viewed as a normal logic program enhanced with an interface to access an external DL knowledge base, so we begin by briefly introducing DL knowledge bases.

### 6.1. DL knowledge bases

We assume familiarity with the basics of description logics [2], and for simplicity consider $\mathcal{SHOIN}$, a DL underlying the Web ontology language OWL DL [40]. The approach presented in this paper can easily be extended to other more expressive DLs such as $\mathcal{SROIQ}$ (a logical underpinning for OWL 2) [38,36], and to DLs with *datatypes* such as $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SROIQ}(\mathbf{D})$. As well, it can be adjusted for light-weight description logics such as DL-Lite [10] or $\mathcal{EL}^{++}$ [1].

Consider a signature $\Psi = (\mathbf{A} \cup \mathbf{R}, \mathbf{I})$, where $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{I}$ are pairwise disjoint (denumerable) sets of *atomic concepts, atomic roles* and *individuals*, respectively. A *role* is either an atomic role $R$ from $\mathbf{R}$ or its inverse, denoted $R^-$. General *concepts* $C$ are formed from atomic concepts, roles and individuals, according to the following syntax:

$$C ::= \top \mid \bot \mid A \mid \{a\} \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \exists R.C \mid \forall R.C \mid \geq_n R \mid \leq_n R$$

where $A$ is an atomic concept from $\mathbf{A}$, $R$ is a role, $a$ is an individual from $\mathbf{I}$, $C$ and $D$ are concepts, and $n$ is a non-negative integer. An *axiom* is of the form $C \sqsubseteq D$ (*concept inclusion axiom*), $R \sqsubseteq R_1$ (*role inclusion axiom*), $\mathrm{Trans}(R)$ (*transitivity axiom*), $C(a)$ (*concept membership axiom*), $R(a, b)$ (*role membership axiom*), $a \approx b$ (*equality axiom*), or $a \not\approx b$ (*inequality axiom*), where $R$, $R_1$ are atomic roles in $\mathbf{R}$, and $a, b$ are individuals in $\mathbf{I}$. We use $C \equiv D$ to denote $C \sqsubseteq D$ and $D \sqsubseteq C$.

Note that for a concept inclusion axiom $C \sqsubseteq D$, we can express its negation $\neg(C \sqsubseteq D)$ by a concept membership axiom $(C \sqcap \neg D)(b)$, where $b$ is a fresh individual in $\mathbf{I}$.

A *DL knowledge base* $L$ is a finite set of axioms. Since DLs are fragments of first-order logic with equality, where atomic concepts (resp. roles) are unary (resp. binary) predicate symbols, and individuals are constants, $L$ has first-order semantics. Therefore, $L$ is consistent (or satisfiable) if $L$ has a first-order model. For an axiom $F$, the entailment relation $L \models F$ is defined as in first-order logic. Note that if $L$ is inconsistent, then $L \models F$ for every formula $F$. When we say 'predicate symbols in $L$', we refer to atomic concepts or atomic roles in $L$.

### 6.2. Dl-programs

Let $L$ be a DL knowledge base built over a signature $\Psi = (\mathbf{A} \cup \mathbf{R}, \mathbf{I})$. Let $\Phi = (\mathbf{P}, \mathbf{C})$ be a signature built from the signature $\Sigma$ of the first-order language $\mathcal{L}_{\Sigma}$ of Section 2, where $\mathbf{P} \subseteq \mathcal{P}$ is a finite set of predicate symbols and $\mathbf{C} \subseteq \mathcal{C}$ a nonempty finite set of constants, such that $\mathbf{P} \cap (\mathbf{A} \cup \mathbf{R}) = \emptyset$ and $\mathbf{C} \subseteq \mathbf{I}$. Terms and atoms are defined only using constants in $\mathbf{C}$, variables in $\mathcal{V}$, and predicate symbols in $\mathbf{P}$. An equality (resp. inequality) is of the form $t_1 \approx t_2$ (resp. $t_1 \not\approx t_2$), where $t_1$ and $t_2$ are terms.

A *dl-query* is built over the signatures $\Psi$ and $\Phi$, which is either (i) a concept inclusion axiom $F$ or its negation $\neg F$; or (ii) of the form $C(t)$ or $\neg C(t)$, where $C$ is a concept, and $t$ is a term; or (iii) of the form $R(t_1, t_2)$ or $\neg R(t_1, t_2)$, where $R$ is a role, and $t_1$ and $t_2$ are terms; or (iv) of the form $t_1 \approx t_2$ or $t_1 \not\approx t_2$, where $t_1$ and $t_2$ are terms. For convenience, we denote a dl-query by $Q(\mathbf{t})$, where $\mathbf{t}$ is all terms of the dl-query (e.g., $t_1$ and $t_2$ in (iii)), and $Q$ is the other part (e.g., $R$ or $\neg R$ in (iii)).

[4] Technically, for atomic rule heads it can be captured in the framework of Denecker et al. [16] and of Pelov et al. [53].

A *dl-atom* is of the form $DL[S_1op_1p_1, \ldots, S_mop_mp_m; Q](\mathbf{t})$, where each $S_i$ is a concept or role built from $\mathbf{A} \cup \mathbf{R}$, or an equality/inequality symbol; $op_i \in \{\uplus, \cup, \cap\}$ is an operator; $p_i \in \mathbf{P}$ is a unary predicate symbol if $S_i$ is a concept, and a binary predicate symbol otherwise; and $Q(\mathbf{t})$ is a dl-query. Note that each $S_iop_ip_i$ maps a predicate symbol $p_i$ in $\mathbf{P}$ to a concept or role $S_i$ over $\mathbf{A} \cup \mathbf{R}$ via a special interface operator $op_i$. Each $p_i$, $1 \leq i \leq m$, is called an input predicate symbol, and each atom with a predicate symbol $p_i$ called an input atom.

A *dl-rule* (or rule) is of the form

$$H \leftarrow A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n$$

where $H$ is an atom, each $A_i$ is either an atom, an equality/inequality or a dl-atom, and each $B_i$ is an atom or a dl-atom. Each $\neg B_i$ is also called a negative literal.

**Definition 13.** (See [25,24].) A *dl-program* $\Pi$ relative to an external DL knowledge base $L$ is a finite set of dl-rules.

A *ground instance* of a rule $r$ is obtained by first replacing every variable in $r$ with a constant from $\mathbf{C}$, then removing all valid equalities and inequalities (under the unique name assumption). A ground instance of $r$ is *consistent* if it contains no equalities or inequalities. Let $ground(\Pi)$ denote the set of all consistent ground instances of rules in $\Pi$.

### 6.3. The well-justified FLP answer set semantics

The semantics of a dl-program $\Pi$ relative to $L$ is defined in terms of Herbrand interpretations, where the Herbrand base $\mathcal{HB}_\Pi$ of $\Pi$ is the set of ground atoms $p(a_1, \ldots, a_n)$ such that $p \in \mathbf{P}$ occurs in $\Pi$ and each $a_i$ is in $\mathbf{C}$.

The satisfaction relation is extended to dl-atoms in the following way. Let $I$ be a Herbrand interpretation of $\Pi$ and $A = DL[S_1op_1p_1, \ldots, S_mop_mp_m; Q](\mathbf{c})$ be a dl-atom occurring in $ground(\Pi)$. Then $I$ *satisfies* the dl-atom $A$ if $L \cup \bigcup_{i=1}^{m} A_i(I) \models Q(\mathbf{c})$, where for $1 \leq i \leq m$, $A_i(I)$ is defined by the predicate mapping $S_iop_ip_i$ such that

$$A_i(I) = \begin{cases} \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, & \text{if } op_i = \uplus; \\ \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, & \text{if } op_i = \cup; \\ \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \notin I\}, & \text{if } op_i = \cap. \end{cases}$$

This extended satisfaction relation to dl-atoms is called *satisfaction under* $L$, denoted $\models_L$, in [24]. Therefore, a Herbrand interpretation $I$ is a model of a dl-program $\Pi$ relative to $L$ if $I$ satisfies all rules in $ground(\Pi)$.

A ground dl-atom $A$ is *monotonic* relative to $\Pi$ and $L$ if for every $I \subseteq J \subseteq \mathcal{HB}_\Pi$, it holds that $I$ satisfies $A$ implies $J$ satisfies $A$; otherwise, $A$ is *nonmonotonic*. A dl-program $\Pi$ is *positive* if it has no negative literals in rule bodies and every dl-atom occurring in $ground(\Pi)$ is monotonic. Note that a positive dl-program $\Pi$ has a least model.

For a Herbrand interpretation $I$, let $s\Pi_L^I$ be the *(strong) reduct* obtained from $ground(\Pi)$ by deleting (i) every rule $r$ whose body is not satisfied by $I$, and (ii) from the remaining rules all negative literals and all nonmonotonic dl-atoms. Furthermore, let $w\Pi_L^I$ be the reduct defined like $s\Pi_L^I$ except that in (ii) all negative literals and all dl-atoms are deleted.

Eiter et al. [24] defined the weak answer set semantics in terms of the reduct $w\Pi_L^I$. A Herbrand interpretation $I$ is a *weak answer set* of $\Pi$ relative to $L$ if $I$ is the least model of $w\Pi_L^I$.

However, Eiter et al. noted as an obvious disadvantage of the weak answer set semantics that it may produce "unfounded" answer sets with circular justifications by self-supporting loops, which they illustrated on the following example.

**Example 13.** Consider the dl-program $\Pi = \{p(a) \leftarrow DL[c \uplus p; c](a)\}$ relative to a DL knowledge base $L = \emptyset$. $\Pi$ has two weak answer sets: $I_1 = \emptyset$ and $I_2 = \{p(a)\}$. The atom $p(a)$ is circularly justified in $I_2$ by the self-supporting loop: $p(a) \Leftarrow DL[c \uplus p; c](a) \Leftarrow p(a)$.

To overcome the circular justification problem, Eiter et al. defined the answer set semantics in terms of the reduct $s\Pi_L^I$. A Herbrand interpretation $I$ is a *strong answer set* of $\Pi$ relative to $L$ if $I$ is the least model of $s\Pi_L^I$.

In general, strong answer sets are not minimal models of the underlying dl-programs. To handle this, Eiter et al. [25] considered the FLP answer set semantics in terms of the FLP reduct $f\Pi_L^I$, which consists of all rules $r \in ground(\Pi)$ such that $I$ satisfies $body(r)$. A Herbrand interpretation $I$ is an *FLP answer set* of $\Pi$ relative to $L$ if $I$ is a minimal model of $f\Pi_L^I$.

However, we observe that the problem of circular justifications persists in both the strong answer set semantics and the FLP answer set semantics. The next two dl-programs well illustrate this.

**Example 14.** Let $\Pi = \{p(a) \leftarrow DL[c \uplus p, b \cap q; c \sqcap \neg b](a)\}$ be a dl-program relative to $L = \emptyset$. The dl-atom $DL[c \uplus p, b \cap q; c \sqcap \neg b](a)$ in $\Pi$ queries $L$ whether $a$ is an instance of the concept $c$ but not of the concept $b$, under the assumption that for any $x$, if $p(x)$ is true,then $x$ is in $c$, and if $q(x)$ is false, then $x$ is not in $b$. This dl-atom is nonmonotonic, so both $I_1 = \emptyset$ and $I_2 = \{p(a)\}$ are strong answer sets of $\Pi$. Observe that $p(a)$ is circularly justified in $I_2$ by the self-supporting loop: $p(a) \Leftarrow DL[c \uplus p, b \cap q; c \sqcap \neg b](a) \Leftarrow p(a) \wedge \neg q(a)$.

**Example 15.** Let $L = \emptyset$ and $\Pi$ consist of two rules: $p(a) \leftarrow q(a)$ and $q(a) \leftarrow DL[c \uplus p, b \cap q; c \sqcup \neg b](a)$. $\Pi$ has only one model $I = \{p(a), q(a)\}$. The FLP reduct $f\Pi_L^I$ of $\Pi$ w.r.t. $I$ is $f\Pi_L^I = \Pi$. Therefore, $I$ is an FLP answer set of $\Pi$. We see that $p(a)$ is circularly justified in $I$ by the self-supporting loop: $p(a) \Leftarrow q(a) \Leftarrow DL[c \uplus p, b \cap q; c \sqcup \neg b](a) \Leftarrow p(a) \vee \neg q(a) \Leftarrow p(a)$.

The intuitive reason behind the circular justification problem of the above three answer set semantics for dl-programs is that these semantics do not induce a level mapping on their answer sets. Therefore we overcome the circular justification problem by extending our well-justified FLP answer set semantics from logic programs with first-order formulas to dl-programs.

Since dl-programs are logic programs extended with dl-atoms, given the above extension of the satisfaction relation to dl-atoms, the entailment relation $\models$ and the operator $T_\Pi(O, N)$ extend accordingly to dl-programs. Furthermore, the notion of well-justified FLP answer sets for logic programs with first-order formulas, as well as the properties in Section 4, carries over to dl-programs. In particular, Definition 9 is extended as follows.

**Definition 14.** A Herbrand model $I$ of a dl-program $\Pi$ relative to an external DL knowledge base $L$ is a *well-justified FLP answer set* if for every $A \in I$, $lfp(T_{f\Pi_L^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$.

By Corollary 2 and Theorem 6, such well-justified FLP answer sets for dl-programs are FLP answer sets enhanced with a level mapping and thus are free of circular justifications.

As the head of each rule in the grounding of a dl-program $\Pi$ is a ground atom, the fixpoint $lfp(T_{f\Pi_L^I}(\emptyset, \neg I^-))$ is a set of ground atoms. Thus for each $A$ in a Herbrand model $I$, $lfp(T_{f\Pi_L^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$ if and only if $A \in lfp(T_{f\Pi_L^I}(\emptyset, \neg I^-))$. This immediately leads to the following result.

**Corollary 4.** *A Herbrand model $I$ of a dl-program $\Pi$ relative to a DL knowledge base $L$ is a well-justified FLP answer set if and only if $I = lfp(T_{f\Pi_L^I}(\emptyset, \neg I^-))$.*

**Example 16.** For the two dl-programs $\Pi$ in Examples 13 and 14, $I_1 = \emptyset$ is a well-justified FLP answer set, but $I_2 = \{p(a)\}$ is not. For the dl-program $\Pi$ in Example 15, $I = \{p(a), q(a)\}$ is not a well-justified FLP answer set.

As it turns out, the weak, the strong, the FLP and the well-justified FLP answer set semantics constitute a hierarchy of more restrictive notions of answer sets.

**Theorem 10.** *Every well-justified FLP answer set of a dl-program $\Pi$ is an FLP answer set of $\Pi$, which in turn is a strong answer set of $\Pi$ which in turn is a weak answer set of $\Pi$.*

While this hierarchy is strict in general, for fragments of dl-programs some of its classes may coincide, and in particular well-justified FLP answer sets coincide with other notions of answer sets. We present some important classes with this property.

A rich such fragment is the class of dl-programs in which only monotonic dl-atoms occur in the rules. Note that a sufficient condition for this property is that no $\cap$ operators occurs in dl-atoms (which can be efficiently checked). Three out of the four answer set semantics coincide in this case.

**Theorem 11.** *Let $\Pi$ be a dl-program relative to a DL knowledge base $L$ such that $\Pi$ contains no nonmonotonic dl-atoms. Then $I \subseteq \mathcal{HB}_\Pi$ is a well-justified FLP answer set of $\Pi$ if and only if $I$ is an FLP answer set of $\Pi$ if and only if $I$ is a strong answer set of $\Pi$.*

Theorem 11 does not extend to weak answer sets. In Example 13, relative to $L = \emptyset$ the dl-atom in $\Pi = \{p(a) \leftarrow DL[c \uplus p; c](a)\}$ is monotonic; while $\{p(a)\}$ is a weak answer set of $\Pi$, the program has no strong answer set.

Theorem 11 can be extended to another well-known class of dl-programs, called stratified dl-programs [24]. The notion of a stratification for dl-programs defines an ordered partition of the set of all ground atoms and ground dl-atoms as follows.

**Definition 15.** Let $\Pi$ be a dl-program relative to a DL knowledge base $L$. Let $S$ be the set of dl-atoms occurring in $ground(\Pi)$. A *stratification* of $\Pi$ is a mapping $\mu: \mathcal{HB}_\Pi \cup S \to \{0, 1, \ldots, k\}$ such that

1. For each rule $H \leftarrow A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n$ in $ground(\Pi)$, $\mu(H) \geq \mu(A_i)$ for $1 \leq i \leq m$, and $\mu(H) > \mu(B_i)$ for $1 \leq i \leq n$, and
2. $\mu(D) \geq \mu(l)$ (resp. $\mu(D) > \mu(l)$) for each input atom $l \in \mathcal{HB}_\Pi$ of each monotonic (resp. nonmonotonic) dl-atom $D$ in $S$.

We call $k$ the length of the stratification $\mu$. $ground(\Pi)$ is then partitioned into $k+1$ dl-programs $\Pi_0, \ldots, \Pi_k$ relative to $L$ (called *strata*), where for each $i \in \{0, \ldots, k\}$, $\Pi_i = \{r \in ground(\Pi) \mid \mu(head(r)) = i\}$ with $\mathcal{HB}_{\Pi_i} = \{l \in \mathcal{HB}_\Pi \mid \mu(l) = i\}$. Note that $\Pi_0$ is a positive dl-program.

**Definition 16.** (See [24].) A dl-program $\Pi$ is *stratified* if it has some stratification $\mu$ of some length $k \geq 0$.

For a stratified dl-program $\Pi$, recursions occur only within each stratum $\Pi_i$; no recursion occurs across two strata. This makes the inference of answers of predicates in lower strata independent of answers of predicates in higher strata. Therefore, for every nonmonotonic dl-atom $A$ (resp. every negative literal $\neg B$) occurring in a rule body of stratum $\Pi_i$, since all input atoms of $A$ (resp. $B$) are defined in lower strata than $\Pi_i$, the truth of $A$ (resp. $\neg B$) is determined by the answers $I_{i-1}$ derived from the first $i-1$ strata. This leads to the following transformation.

For a stratum $\Pi_i$ and a Herbrand interpretation $I_{i-1}$, let $\Pi_i(I_{i-1})$ be the set of rules obtained from $\Pi_i$ by deleting

(1) every rule $H \leftarrow A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n$ such that either some $A_j$ is a nonmonotonic dl-atom not satisfied by $I_{i-1}$, or some $B_j$ is satisfied by $I_{i-1}$, and
(2) from the remaining rules all negative literals and all nonmonotonic dl-atoms.

Note that $\Pi_i(I_{i-1})$ is a positive dl-program.

The following result is immediate from [24, Theorems 5.6 and 4.14].

**Theorem 12.** *Let $\Pi$ be a stratified dl-program relative to a DL knowledge base $L$ with $k+1$ strata $\Pi_0, \ldots, \Pi_k$. Then $I \subseteq \mathcal{HB}_\Pi$ is a strong answer set of $\Pi$ if and only if $I = I_k$, where $I_0$ is the least model of $\Pi_0$ and for each $1 \leq i \leq k$, $I_i$ is the least model of $\Pi_i(I_{i-1}) \cup I_{i-1}$.*

Observe that for each $i \in \{0, \ldots, k\}$, $I_i$ is unique and thus $I$ is unique, i.e., a stratified dl-program $\Pi$ has a unique strong answer set. On the other hand, it is immediate from Theorem 12 that for every strong answer set $I$ of $\Pi$, we have $I^- = I_k^-$, where $I_i^- = (\bigcup_{0 \leq j \leq i} \mathcal{HB}_{\Pi_j}) \setminus I_i$, for $i = 0, \ldots, k$. Based on this, the next result shows that for stratified dl-programs, all notions of answer sets that we consider except weak answer sets coincide.

**Theorem 13.** *Let $\Pi$ be a stratified dl-program relative to a DL knowledge base $L$. Then $I \subseteq \mathcal{HB}_\Pi$ is a well-justified FLP answer set of $\Pi$ if and only if $I$ is an FLP answer set of $\Pi$ if and only if $I$ is a strong answer set of $\Pi$.*

Furthermore, all semantics in Theorem 13 are canonical in the sense that they yield a single answer set.

Theorem 13 can be extended to weak answer sets using a stronger notion of stratification, which requires that in item 2 of Definition 15, $\mu(D) > \mu(l)$ for each input atom $l$ of each (monotonic or nonmonotonic) dl-atom $D$. This ensures that there is no cycle through any dl-atoms. In such a *dl-acyclic* stratified program $\Pi$, the truth values of dl-atoms in rule bodies are completely known when the rules should be applied. Therefore, for a stratum $\Pi_i$ of $\Pi$ and a Herbrand interpretation $I_{i-1}$, we can transform $\Pi_i$ to $\Pi_i(I_{i-1})$ by deleting

(1) every rule $H \leftarrow A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n$ such that either some $A_j$ is a dl-atom not satisfied by $I_{i-1}$, or some $B_j$ is satisfied by $I_{i-1}$, and
(2) from the remaining rules all negative literals and all dl-atoms.

Then using the same proof techniques for strong answer sets we can extend Theorems 12 and 13 to weak answer sets.

## 7. Complexity of the well-justified FLP semantics

Observe that a first-order theory amounts to a special logic program in which all rules have an empty body. As it is undecidable to determine whether a given arbitrary first-order theory is satisfiable, it is thus undecidable to determine whether a logic program has an FLP answer set resp. a well-justified FLP answer set. Therefore, we concentrate in this paper on the complexity of propositional logic programs (which are at the core of richer languages) and consider only aggregates that are computable in polynomial time (i.e., for any Herbrand interpretation $I$, checking whether $I$ satisfies an aggregate atom is feasible in polynomial time).

Recall that NP are the decision problems solvable by a nondeterministic Turing machine in polynomial time, and that $\Sigma_2^p = \mathrm{NP}^{\mathrm{NP}}$ is likewise but with the help of an NP oracle. Furthermore, for every complexity class $\mathcal{C}$, the class co-$\mathcal{C}$ is the class of complementary problems (with yes-no answers reversed); in particular, $\Pi_2^p = \text{co-}\Sigma_2^p$. We encounter in addition the complexity classes NEXP (nondeterministic time $2^{poly(n)}$) and N2EXP (nondeterministic time $2^{2^{poly(n)}}$), where $poly(n) = \bigcup_{k \geq 1} O(n^k)$, and $\mathrm{P}^{\mathrm{NEXP}}$ and $\mathrm{P}^{\mathrm{N2EXP}}$, which contain all decision problems solvable in polynomial time with an NEXP resp. N2EXP oracle. As shown by Hemachandra [37], $\mathrm{P}^{\mathrm{NEXP}}$ coincides with its nondeterministic counterpart $\mathrm{NP}^{\mathrm{NEXP}}$; with his proof technique, the same is easily established for $\mathrm{P}^{\mathrm{N2EXP}}$ and $\mathrm{NP}^{\mathrm{N2EXP}}$.

**Table 1**
Complexity of reasoning tasks for propositional logic programs under the FLP and the well-justified FLP semantics.

| | Answer set existence | Cautious reasoning | Brave reasoning |
|---|---|---|---|
| FLP | $\Sigma_2^p$-complete | $\Pi_2^p$-complete | $\Sigma_2^p$-complete |
| Well-justified FLP | $\Sigma_2^p$-complete | $\Pi_2^p$-complete | $\Sigma_2^p$-complete |

We consider the following canonical reasoning problems:

1. *Answer set existence*: The problem of deciding whether a given logic program $\Pi$ has an answer set.
2. *Cautious reasoning*: The problem of deciding whether a ground atom is in all answer sets of $\Pi$.
3. *Brave reasoning*: The problem of deciding whether a ground atom is in some answer set of $\Pi$.

### 7.1. Complexity of propositional logic programs

For a propositional logic program $\Pi$, we consider the FLP and the well-justified FLP semantics defined over Herbrand models of $\Pi$. Our main complexity results are summarized in Table 1. It is interesting to note that for all of the three reasoning tasks, the FLP and the well-justified FLP answer set semantics fall in the same complexity classes. This means that the well-justified FLP answer set semantics enhances the FLP answer set semantics with a level mapping formalism without affecting the worst-case complexity.

The following theorem shows that for propositional logic programs, deciding the existence of ordinary and well-justified FLP answer sets is complete for $\text{NP}^{\text{NP}}$.

**Theorem 14.** *Given a propositional logic program $\Pi$, deciding whether $\Pi$ has an FLP answer set or a well-justified FLP answer set is both $\Sigma_2^p$-complete.*

The next theorem shows that for propositional logic programs, deciding whether a ground atom is in every (resp. some) FLP/well-justified FLP answer set is complete for co-$\text{NP}^{\text{NP}}$ (resp. $\text{NP}^{\text{NP}}$).

**Theorem 15.** *Given a propositional logic program $\Pi$ and an atom $l \in \mathcal{HB}_\Pi$, deciding whether $l$ is in every (resp. some) FLP answer set of $\Pi$ is $\Pi_2^p$-complete (resp. $\Sigma_2^p$-complete). The same holds for well-justified FLP answer sets.*

The $\Sigma_2^p$- resp. $\Pi_2^p$-hardness of the FLP and the well-justified FLP answer set semantics for propositional logic programs is inherited to some particular fragments, e.g., to propositional logic programs with rules $H \leftarrow B$ with an atomic head $H$. On the other hand, for some natural fragments the FLP answer set semantics still remains $\Sigma_2^p$- resp. $\Pi_2^p$-hard, while the well-justified FLP answer semantics has presumably lower complexity; for example, for propositional logic programs with rules $H \leftarrow B$, where $H$ is a disjunction of atoms and $B$ a conjunction of literals, it is easily seen that the well-justified FLP answer semantics for this fragment is NP- resp. co-NP-complete for the above reasoning tasks.

The results for propositional logic programs are easily lifted to logic programs with quantifier-free rules, i.e., rules of the form $H \leftarrow B$ where $H$ and $B$ are quantifier-free formulas. The complexity in Theorems 14 and 15 increases by one exponential to $\text{NEXP}^{\text{NP}}$ resp. co-$\text{NEXP}^{\text{NP}}$; intuitively, like for normal logic programs this increase is due to the exponentially more succinct representation using variables, whose elimination by grounding causes a blowup, cf. [11].

### 7.2. Complexity of propositional logic programs with aggregates

When propositional logic programs are extended with polynomially computable aggregates, the complexity under the FLP and the well-justified FLP answer set semantics falls in the same classes as that of propositional logic programs without aggregates.

**Theorem 16.** *Given a propositional logic program $\Pi$ with polynomially computable aggregate atoms, deciding (i) whether $\Pi$ has some FLP answer set is $\Sigma_2^p$-complete; (ii) whether a given atom $l \in \mathcal{HB}_\Pi$ is in every (resp. some) FLP answer set of $\Pi$ is $\Pi_2^p$-complete (resp. $\Sigma_2^p$-complete). The same holds for well-justified FLP answer sets.*

The $\Sigma_2^p$- resp. $\Pi_2^p$-hardness holds even for particular fragments such as ground normal and ground Horn logic programs with polynomially computable aggregates. A *ground normal logic program $\Pi$ with aggregate atoms* consists of rules of the form $H \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n$, where $H$ is a ground atom, and each $B_i$ and $C_i$ is either a ground atom or a ground aggregate atom; $\Pi$ is a *ground Horn logic program with aggregate atoms* if $n = 0$ for every rule in $\Pi$.

Faber et al. [27] showed that determining whether a given ground normal or Horn logic program with polynomially computable aggregates has an FLP answer set is both $\Sigma_2^p$-complete. This result also holds for the well-justified FLP answer set semantics.

**Table 2**
Complexity of reasoning tasks for dl-programs $\Pi$ relative to a DL knowledge base $L$ under the FLP and the well-justified FLP answer set semantics.

| | Answer set existence | Cautious reasoning | Brave reasoning |
|---|---|---|---|
| $L$ in $\mathcal{SHIF}(\mathbf{D})$ | NEXP-complete | co-NEXP-complete | NEXP-complete |
| $L$ in $\mathcal{SHOIN}(\mathbf{D})$ | $\mathrm{P}^{\mathrm{NEXP}}$-complete | $\mathrm{P}^{\mathrm{NEXP}}$-complete | $\mathrm{P}^{\mathrm{NEXP}}$-complete |
| $L$ in $\mathcal{SROIQ}(\mathbf{D})$ | $\mathrm{P}^{\mathrm{N2EXP}}$-complete | $\mathrm{P}^{\mathrm{N2EXP}}$-complete | $\mathrm{P}^{\mathrm{N2EXP}}$-complete |

**Theorem 17.** *For a ground normal logic program $\Pi$ with polynomially computable aggregate atoms, deciding whether $\Pi$ has some well-justified FLP answer set is $\Sigma_2^p$-complete. Furthermore, $\Sigma_2^p$-hardness holds already for ground Horn logic programs $\Pi$ with polynomially computable aggregates.*

It is immediate that the complexity classes of cautious and brave reasoning for ground normal or Horn logic programs with polynomially computable aggregates are the same as those classes for propositional logic programs with polynomially computable aggregates; i.e., $\Pi_2^p$-complete for cautious reasoning and $\Sigma_2^p$-complete for brave reasoning.

It is worth noting that while the results are analogous to those in [27], the setting of aggregates is different. In [27]'s formalism, ground aggregate atoms are essentially of the form

$$A = \mathrm{OP}\langle\{(a_i : F_i) \mid 1 \le i \le m\}\rangle \succeq b$$

where $a_i$ is a constant and each $F_i$ is a conjunction of ground atoms; an interpretation $I$ satisfies $A$ if $\mathrm{OP}\{a_i \mid I \text{ satisfies } F_i, 1 \le i \le m\} \succeq b$ evaluates to true. Thus $A$ amounts in our framework to an aggregate atom

$$A' = \mathrm{OP}\left\langle (\{a_1, \ldots, a_m\}, X) : \bigvee_{i=1}^{m} X \approx a_i \wedge F_i \right\rangle \succeq b.$$

[27] showed that $\Sigma_2^p$-hardness of the FLP answer set semantics is present already for a ground normal logic program $\Pi$ with polynomially computable ground aggregate atoms of form $A$, where $m$ and the size of each $F_i$ are bounded by a constant $k$. However, for $\Pi'$ that is $\Pi$ with all aggregate atoms $A$ replaced by $A'$, deciding the existence of a well-justified answer set of $\Pi'$ lies in NP. Informally, this holds because in this case, in the fixpoint computation $T^i_{f\Pi'^I}(\emptyset, \neg I^-)$ all possible values of aggregation sets $S^J_{A'}$ for all interpretations $J$ that satisfy $T^i_{f\Pi'^I}(\emptyset, \neg I^-) \cup \neg I^-$ can be determined in polynomial time (in the bound $k$).

As the semantics of a nonground normal logic program $\Pi$ with aggregates is defined in terms of its grounding $ground(\Pi)$, it is natural to view a nonground aggregate atom $A$ as polynomially computable if each ground instance of $A$ is polynomially computable. Intuitively, since grounding causes an exponential blowup, the complexity of nonground normal logic programs with polynomially computable aggregates is exponentially higher than in the ground case, and thus complete for $\mathrm{NEXP}^{\mathrm{NP}}$ resp. co-$\mathrm{NEXP}^{\mathrm{NP}}$.

Although it is interesting to study in what cases an aggregate atom is polynomially computable, the topic is beyond the scope of the current paper. As a showcase, however, we mention a class of aggregate atoms of the form

$$A'' = \mathrm{OP}\langle (D_1, X_1), \ldots, (D_n, X_n) : F(X)\rangle \succeq b$$

where $X = X_1, \ldots, X_n$ is a list of aggregate variables with corresponding domains $D_1, \ldots, D_n$, and $F(X)$ is a formula with no quantifiers, no function symbols, and no variables other than the $X_i$s. Note that aggregate atoms of form $A'$ above are in this class. Such an aggregate atom $A''$ is polynomially computable if deciding whether a given (Herbrand) interpretation $I$ satisfies $A''$ can be done in time polynomial in the size of $I$ and $A''$; this is ensured if the number $n$ of aggregate variables is bounded by a constant and $OP$ can be calculated in polynomial time.

### 7.3. Complexity of Dl-programs

The complexity of the FLP and the well-justified FLP answer set semantics for a dl-program $\Pi$ relative to a DL knowledge base $L$ depends on the class of DL that $L$ belongs to. Table 2 summarizes the complexity results when $L$ is in $\mathcal{SHIF}(\mathbf{D})$, $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SROIQ}(\mathbf{D})$, respectively.

We first prove the following two theorems for the well-justified FLP answer set semantics.

**Theorem 18.** *Given a dl-program $\Pi$ relative to a DL knowledge base $L$, deciding whether $\Pi$ has some well-justified FLP answer set is (i)* NEXP*-complete if $L$ is in $\mathcal{SHIF}(\mathbf{D})$, (ii)* $\mathrm{P}^{\mathrm{NEXP}}$*-complete if $L$ is in $\mathcal{SHOIN}(\mathbf{D})$, and (iii)* $\mathrm{P}^{\mathrm{N2EXP}}$*-complete if $L$ is in $\mathcal{SROIQ}(\mathbf{D})$.*

**Theorem 19.** *Given a dl-program $\Pi$ relative to a DL knowledge base $L$ and an atom $l \in \mathcal{HB}_\Pi$, deciding whether $l$ is in every (resp. some) well-justified FLP answer set of $\Pi$ is complete for (i) co-*NEXP *(resp.* NEXP*) if $L$ is in $\mathcal{SHIF}(\mathbf{D})$, (ii)* $\mathrm{P}^{\mathrm{NEXP}}$ *(resp.* $\mathrm{P}^{\mathrm{NEXP}}$*) if $L$ is in $\mathcal{SHOIN}(\mathbf{D})$, and (iii)* $\mathrm{P}^{\mathrm{N2EXP}}$ *(resp.* $\mathrm{P}^{\mathrm{N2EXP}}$*) if $L$ is in $\mathcal{SROIQ}(\mathbf{D})$.*

The next theorem shows that the FLP answer set semantics has the same complexity classes on the problem of answer set existence as the well-justified FLP answer set semantics.

**Theorem 20.** *Given a dl-program $\Pi$ relative to a DL knowledge base $L$, deciding whether $\Pi$ has some FLP answer set is (i)* NEXP*-complete if $L$ is in $\mathcal{SHIF}(\mathbf{D})$, (ii)* $\mathrm{P}^{\mathrm{NEXP}}$*-complete if $L$ is in $\mathcal{SHOIN}(\mathbf{D})$, and (iii)* $\mathrm{P}^{\mathrm{N2EXP}}$*-complete if $L$ is in $\mathcal{SROIQ}(\mathbf{D})$.*

Since cautious (resp. brave) reasoning for dl-programs falls in the same complexity classes as the non-existence (resp. existence) of FLP answer sets, it immediately follows from Theorem 20 that the complexity classes of cautious (resp. brave) reasoning under the FLP answer set semantics is the same as those classes under the well-justified FLP answer set semantics.

We finally note that analogous upper bounds (i.e., membership results) to those in Table 2 hold for dl-programs over description logics where knowledge base satisfiability has the same complexity as for $\mathcal{SHIF}(\mathbf{D})$, $\mathcal{SHOIN}(\mathbf{D})$ or $\mathcal{SROIQ}(\mathbf{D})$ (which is EXP-, NEXP- and N2EXP-complete, respectively). However, matching lower bounds (i.e., corresponding hardness results) are not entailed by our results.

## 8. Implementation of the well-justified FLP semantics

We have implemented the well-justified FLP answer set semantics and developed a system that hosts normal logic programs with aggregates, dl-programs and, moreover, HEX-programs (which we did not consider here). In this section, we describe the algorithm used for the implementation and the architecture of the system. We also describe an experimental evaluation of the performance of computing both FLP and well-justified FLP answer sets over some benchmark logic programs.

For simplicity, in the following description we restrict to ground logic programs. A *complex atom* is either an aggregate atom, a dl-atom, or an external atom, so by a *normal logic program with complex atoms* we refer to a normal logic program with aggregate atoms, a dl-program, or a HEX-program. All complex atoms $A$ are assumed to be decidable, i.e., for any Herbrand interpretation $I$, checking whether $I$ satisfies $A$ is feasible in finite time.

### 8.1. Implementation description

Our algorithm for computing well-justified FLP answer sets consists of two main parts: a guessing and a checking part. Given a normal logic program $\Pi$ with complex atoms, the guessing part computes models of $\Pi$ that serve as answer set candidates. For each such model $I$, the checking part then computes the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ for the FLP reduct $f\Pi^I$; if $I = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$, then $I$ is a well-justified FLP answer set of $\Pi$.

The implementation realizes the guessing part by first transforming $\Pi$ into a normal logic program $\hat{\Pi}$ without complex atoms (called the guessing program). The result of this step will be sent to an ASP solver, which computes the stable models of $\hat{\Pi}$ under the standard answer set semantics. These models are used as input to the checking part, which selects the well-justified FLP answer sets as output.

**Definition 17.** Let $\Pi$ be a normal logic program with complex atoms. The *guessing program* of $\Pi$, denoted $\hat{\Pi}$, is obtained from $\Pi$ as follows. For each complex atom $A$ in $\Pi$, (1) replace $A$ with a fresh atom $E_A$, and (2) add two new rules to $\Pi$, $E_A \leftarrow \neg E'_A$ and $E'_A \leftarrow \neg E_A$, where $E'_A$ is a fresh atom.

For convenience, in the above definition we call $E_A$ the *replacement atom* of $A$, and call $A$ the *source complex atom* of $E_A$.

For an interpretation $\hat{I}$ of $\hat{\Pi}$, its *projection $I$ on $\Pi$* is $\hat{I}$ with all replacement atoms $E_A$ along with $E'_A$ removed. Observe that when $\hat{I}$ is an answer set of $\hat{\Pi}$, $I$ may not be an FLP or a well-justified FLP answer set of $\Pi$, and even not be a model of $\Pi$. So we make use of the concept of compatible sets as introduced by Eiter et al. [20].

**Definition 18.** Let $\Pi$ be a normal logic program with complex atoms and $\hat{\Pi}$ be its guessing program. Let $\hat{I}$ be an answer set of $\hat{\Pi}$ and $I$ be its projection on $\Pi$. We call $\hat{I}$ a *compatible set* of $\Pi$, if for every replacement atom $E_A$ in $\hat{\Pi}$, $E_A \in \hat{I}$ if and only if $I$ satisfies the source complex atom $A$ of $E_A$.

It is not hard to see that for every compatible set $\hat{I}$ of $\Pi$, its projection $I$ on $\Pi$ is a model of $\Pi$, and that the projections of all compatible sets include all FLP answer sets of $\Pi$ (see the proof of Theorem 21), hence all well-justified FLP answer sets of $\Pi$.

However, there may exist compatible sets whose projections are not FLP answer sets. Therefore, we need to check whether the projection $I$ of each compatible set $\hat{I}$ is an FLP resp. well-justified FLP answer set of $\Pi$. This check amounts to verifying that $I$ is a minimal model of the FLP reduct $f\Pi^I$ in case of FLP answer sets [21]; for well-justified FLP answer sets, the checking part permits only candidates $I$ that are equal to $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$.

Algorithm 1 summarizes the process of computing all well-justified FLP answer sets using compatible sets.

**Algorithm 1:** Computing well-justified FLP answer sets.

**Input**: A normal logic program with aggregates, a dl-program, or a HEX-program $\Pi$
**Output**: All well-justified FLP answer sets of $\Pi$

Construct the guessing program $\hat{\Pi}$ from $\Pi$
$AS = \emptyset$
**for** *each answer set* $\hat{I}$ *of* $\hat{\Pi}$ **do** `// check if` $\hat{I}$ `is a compatible set`
  Let $I$ be the projection of $\hat{I}$ on $\Pi$
  compatible := true
  **for** *each replacement atom* $E_A$ *in* $\hat{\Pi}$ **do**
    Let $A$ be the source complex atom of $E_A$
    **if** $E_A \in \hat{I}$ *but* $I$ *does not satisfy* $A$, *or* $E_A \notin \hat{I}$ *but* $I$ *satisfies* $A$ **then**
      compatible := false
  **if** *compatible* = *true* **then**
    `//` $\hat{I}$ `is a compatible set; do fixpoint check`
    **if** $I = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ **then**
      $AS := AS \cup \{I\}$
**return** $AS$

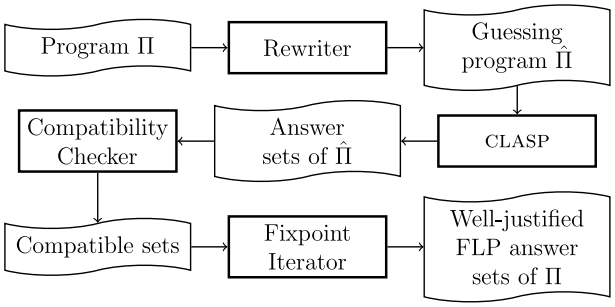


**Fig. 1.** System architecture of DLVHEX.

**Example 17.** Consider the logic program $\Pi_1$ from Example 1. In the guessing part, we construct the following guessing program:

$$\hat{\Pi}_1: \quad p(1). \qquad r_1$$
$$p(2) \leftarrow p(-1). \qquad r_2$$
$$p(-1) \leftarrow E_{SUM\langle X:p(X)\rangle \geq 1}. \qquad r_3$$
$$E_{SUM\langle X:p(X)\rangle \geq 1} \leftarrow \neg E'_{SUM\langle X:p(X)\rangle \geq 1}. \qquad r_4$$
$$E'_{SUM\langle X:p(X)\rangle \geq 1} \leftarrow \neg E_{SUM\langle X:p(X)\rangle \geq 1}. \qquad r_5$$

The replacement atom $E_{SUM\langle X:p(X)\rangle \geq 1}$ has been introduced for the aggregate atom $\mathrm{SUM}\langle X : p(X)\rangle \geq 1$ occurring in $\Pi_1$. The program $\hat{\Pi}_1$ has two answer sets: $\hat{I}_1 = \{p(1), E'_{SUM\langle X:p(X)\rangle \geq 1}\}$ and $\hat{I}_2 = \{p(1), E_{SUM\langle X:p(X)\rangle \geq 1}, p(-1), p(2)\}$. Their projections on $\Pi_1$ are $I_1 = \{p(1)\}$ and $I_2 = \{p(1), p(-1), p(2)\}$, of which only $\hat{I}_2$ is a compatible set of $\Pi_1$; thus $I_2$ is the only answer set candidate of $\Pi_1$. The checking part then computes the fixpoint $lfp(T_{f\Pi_1^{I_2}}(\emptyset, \neg I_2^-)) = \{p(1)\}$; as it is different from $I_2$, the latter is not a well-justified FLP answer set of $\Pi_1$. Consequently, Algorithm 1 outputs for $\Pi_1$ no well-justified FLP answer sets, which is the correct result.

The following result shows that Algorithm 1 correctly computes the well-justified FLP answer set semantics.

**Theorem 21.** *Assume all complex atoms in logic programs are decidable. Then Algorithm 1 is sound and complete w.r.t. the well-justified FLP answer set semantics for normal logic programs with aggregates, dl-programs and* HEX*-programs.*

We implemented Algorithm 1 by extending our ASP reasoner DLVHEX.[5] The system architecture of DLVHEX is depicted in Fig. 1, which consists of four major components:

[5] DLVHEX is available at http://www.kr.tuwien.ac.at/research/systems/dlvhex.

- a *rewriter*, which constructs a guessing program $\hat{\Pi}$ from a normal logic program $\Pi$ with complex atoms;
- a state-of-the-art *ASP solver* CLASP,[6] which computes answer sets of $\hat{\Pi}$;
- a *compatibility checker*, which identifies the compatible sets among the answer sets of $\hat{\Pi}$; and
- a *fixpoint iterator*, which for the projection $I$ of each compatible set $\hat{I}$ computes the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$.

DLVHEX can also compute FLP answer sets, using the same architecture except that the fixpoint iterator is replaced by a *minimality checker*, which checks if $I$ is a minimal model of the FLP reduct $f\Pi^I$ [21].

### 8.2. Experimental evaluation

In Section 7, we show that the well-justified FLP answer set semantics enhances the FLP answer set semantics with a level mapping formalism without affecting the worst-case complexity. In this subsection, we present experimental results which show that computing well-justified FLP answer sets is faster than computing FLP answer sets on some benchmark programs. To this end, we use HEX-program encodings of several benchmark problems that have been developed in other contexts such that the FLP answer sets correspond to the solutions of the problems; as the well-justified FLP answer sets are particular FLP answer sets, the latter yield particular solutions obtained by a fixpoint construction that avoids a customary minimality check for a model candidate under the FLP semantics, which usually is expensive. It is thus interesting to see the effect of resorting to well-justified FLP answer sets of the encodings if one is just interested in some solutions (and less in the additional quality of avoiding circularity).

For the evaluation we compared runtimes in seconds under the FLP and under the well-justified FLP semantics using three benchmark domains: Abstract Argumentation, Inconsistency Explanation for Multi-Context Systems, and Set Partitioning, where $\mathcal{A}$, $\mathcal{M}$ and $\mathcal{P}$ are used to denote the set of all instances, respectively. The first and the second are motivated by applications in knowledge representation and reasoning, while the third benchmark is synthetic. Each benchmark instance $i \in \mathcal{A} \cup \mathcal{M} \cup \mathcal{P}$ with size $|i|$ has an associated parameter setting. We collect the total runtime $t_X(i)$, i.e., the time from startup to termination, and the runtime per answer set $p_X(i)$, i.e., the total runtime divided by the number of answer sets (which is only applicable if at least one answer set exists) of our system under the FLP answer set semantics ($X = flp$) and under the well-justified FLP answer set semantics ($X = wj$). We summarize for each benchmark $\mathcal{B} = \mathcal{A}, \mathcal{M}, \mathcal{P}$ and instance size $s$ the outcome in tables with maximum and average factors $t^{\mathcal{B}}_{\max}(s)$, $p^{\mathcal{B}}_{\max}(s)$, $t^{\mathcal{B}}_{\text{avg}}(s)$ and $p^{\mathcal{B}}_{\text{avg}}(s)$, which have been computed for $op = \max, \text{avg}$ and runtime $r = t, p$ as follows:

$$r^{\mathcal{B}}_{op}(s) = op\left\{ \frac{r_{wj}(i)}{r_{flp}(i)} \;\middle|\; i \in \mathcal{B} \text{ and } s = |i| \right\}.$$

We call $r^{\mathcal{B}}_{op}(s)$ a *speedup factor*, if $r^{\mathcal{B}}_{op}(s) < 1$ and a *slowdown factor*, if $r^{\mathcal{B}}_{op}(s) > 1$. For computing $p^{\mathcal{B}}_{op}(s)$ we consider only those instances which have at least one answer set under both semantics.

We evaluated the implementation on a Linux server with two 12-core AMD 6176 SE CPUs with 128 GB RAM. The runtimes are compared with a timeout of 300 seconds, and each run has been limited to use at most 4 GB main memory. Learning from external sources (cf. [20] for *external behavior learning*) is an important optimization mechanism for improving the performance of answer set computation. At the moment, however, it has been implemented in DLVHEX only for the FLP answer set semantics, not yet for the well-justified FLP semantics. Thus we decided to turn this optimization mechanism off for the benchmarks to ensure a fair comparison. If external behavior learning was enabled, due to the effect of optimization, the evaluation under the FLP answer set semantics would be faster than under the well-justified FLP semantics in all cases.[7] The problem encodings have been developed for the FLP semantics. The well-justified FLP answer set semantics delivers a subset of the FLP answer sets as an approximation therefore.

**Abstract argumentation** An *abstract argumentation framework* (AF) [18] is a pair $F = (A, R)$ of a set $A$ of arguments and a relation $R \subseteq A \times A$ that can be viewed as a directed graph, where the nodes represent arguments and an arc $a \to b$ represents that argument $a$ attacks argument $b$. The semantics of an AF is defined in terms of *extensions*, which are sets of arguments that fulfill certain criteria, depending on the particular semantics in use. As shown by Dung, a number of problems in artificial intelligence can be elegantly encoded as reasoning tasks on abstract argumentation frameworks.

In this benchmark, which was considered earlier in [21], we consider computing *ideal sets* [17], which serve to refine the seminal semantics in [18]. A set $I$ of arguments is an ideal set of an AF $F = (A, R)$ if $I$ is an admissible set that is contained in all preferred extensions, i.e., subset-maximal admissible set of $F$, where a set $S$ of arguments is admissible if $S$ does not contain self attacks, i.e., there are no arcs between nodes in $S$, and each argument attacking some argument in $S$ is attacked by some argument in $S$, i.e., if an arc leads from a node $a$ into $S$ then an arc leads from some node in $S$ to $a$. For example, if $F = (\{a, b, c\}, \{(a, b), (b, a)\})$, then $\emptyset$, $\{c\}$, $\{a, c\}$, and $\{b, c\}$ are the admissible sets and thus $\{a, c\}$, $\{b, c\}$ the preferred extensions; hence, $\emptyset$ and $\{c\}$ are the ideal sets. For further discussion and use of ideal sets, we refer to [17,5].

[6] Available at http://www.cs.uni-potsdam.de/clasp.

[7] All benchmark encodings, instances and results are available at http://www.kr.tuwien.ac.at/staff/redl/wjflp.

**Table 3**
Argumentation benchmark results (time for well-justified FLP over ordinary FLP answer sets; speedup if $< 1$, slowdown if $> 1$).

| $n$ | $t^{\mathcal{A}}_{\text{avg}}(n)$ | $t^{\mathcal{A}}_{\text{max}}(n)$ | Slower instances | Faster instances | $p^{\mathcal{A}}_{\text{avg}}(n)$ | $p^{\mathcal{A}}_{\text{max}}(n)$ |
|---|---|---|---|---|---|---|
| 5 | 0.87 | 0.65 | 34.00% | 66.00% | 11.11 | 20.00 |
| 6 | 0.65 | 0.42 | 10.00% | 90.00% | 14.29 | 25.00 |
| 7 | 0.44 | 0.22 | 2.00% | 98.00% | 14.29 | 25.00 |
| 8 | 0.29 | 0.15 | 2.00% | 98.00% | 11.11 | 25.00 |
| 9 | 0.25 | 0.13 | 0.00% | 100.00% | 5.56 | 10.00 |
| 10 | 0.46 | 0.20 | 0.00% | 74.00% | 1.72 | 9.09 |

The HEX-program encoding uses an external atom that allows to verify whether a given set $T$ of arguments is a preferred extension of the input AF $F$; a guess for an ideal set $I$ is then verified using this atom and a customary saturation technique to ensure that no preferred extension $T$ exists such that $I \not\subseteq T$. The FLP answer sets of the encoding correspond one-to-one to the ideal extensions of $F$.[8]

Each argumentation framework $F = (A, R)$ in our benchmark set $\mathcal{A}$ consists of $n = |A|$ arguments, and $R$ *is the attacks relation that* consists of edges independently chosen from $A \times A$ with probability $p$. For each parameter setting $(n, p)$, where $n = 5, 6, \ldots, 10$ and $p \in \{0.03, 0.05, 0.07, 0.09, 0.11\}$, we have created ten instances $F(n, p, 1), \ldots, F(n, p, 10)$; the size of each instance $F$ is $n = |A|$.

Table 3 summarizes the results grouped by the number $n$ of arguments. We also report the percentage of the instances within each group of $n$ arguments that are faster resp. slower under the well-justified FLP semantics compared to the FLP semantics. For the remaining instances, we either could not observe a speedup or slowdown, or the runs timed out under both semantics. We see that if we measure the total runtime for an instance, then the evaluation under the well-justified FLP semantics is more efficient than under the FLP semantics for a majority of instances.

Eiter et al. [21] noted that the minimality check for model candidates under the FLP semantics is costly for this benchmark problem. In contrast, the fixpoint iteration under the well-justified FLP semantics seems to be rather cheap. Our explanation for this effect is that many atoms in this benchmark domain can be computed deterministically by exploiting the program structure. Thus, checking satisfaction of external atoms under partial assignments is efficient for such instances, even though the worst case would require to make exponentially many calls until the fixpoint has been reached. On the other hand, the minimality check of the FLP reduct under the FLP semantics remains exponential.

If we measure the runtime per answer set as defined above, the picture is different. In this case the evaluation under the FLP semantics is more efficient, as there is a large number of FLP answer sets that are not well-justified FLP answer sets (for most instances, only one FLP answer set is well-justified). Thus, the FLP semantics requires to compute far more models, but with only slightly longer total runtime, which leads to a better average runtime.

**Inconsistency explanation for Multi-Context Systems** Nonmonotonic Multi-Context-Systems (MCSs) were proposed in [7] as a generic formalism for aligning knowledge bases called *contexts*, which emerged by an evolution of formalisms rooted in [35]. An MCS is a collection $M = (C_1, \ldots, C_n)$ of contexts $C_i$, each of which holds a knowledge base $kb_i$ in some logic $L_i$ whose semantics is given in terms of abstract acceptable belief sets (which usually are sets of formulas, or models). The contexts are interlinked via so called bridge rules, which enable belief exchange across contexts; for example, a bridge rule $br: \ (1{:}a) \leftarrow not(2{:}b)$ informally says that $a$ should be in $C_1$'s knowledge base, if $b$ is not in the local belief set of context $C_2$. The semantics of an MCS is defined in terms of equilibria, which are belief states $S = (S_1, \ldots, S_n)$ composed of local belief sets $S_i$ of the knowledge bases $kb_i$ satisfying the bridge rules.

However, compliance of the bridge rules with the knowledge bases may be impossible to achieve; that is, the MCS is inconsistent (even if the local knowledge bases are consistent). For example, if $M = (C_1, C_2)$, where $kb_1 = \{\bot \leftarrow a\}$ and $kb_2 = \{c\}$ are both logic programs and there is the single bridge rule $br$ from above, then $M$ has no equilibrium, although both $kb_1$ and $kb_2$ have an answer set (where answer sets are acceptable belief sets). To understand the reasons for inconsistency, Eiter et al. [22] introduced the notion of an *inconsistency explanation (IE)* for an MCS $M$, which aims at characterizing an inconsistency core through bridge rules, i.e., faulty interlinkage. Roughly speaking, an IE consists of bridge rules whose presence or inapplicability will necessarily entails inconsistency; the technical definition is involved, and we thus refrain from detailing it here. In the example above, the presence of the single bridge rule $br$ entails inconsistency, and hence amounts to an IE. For further background and discussion of MCS and applications, we refer to [7,8,22].

This benchmark set computes IEs, which correspond one-to-one to the FLP answer sets of an HEX-encoding of the problem. The encoding[9] as cycles through external atoms which intuitively evaluate the semantics of the context knowledge bases. We used the MCS benchmark instances generated for [19]. These random instances are grouped into consistent and inconsistent instances, and the contexts are interlinked with various fixed topologies that should resemble different scenarios: *ordinary* and *zig-zag diamond* stack, *house* stack, *ring*, and *binary tree*. A diamond stack combines multiple diamonds in

[8] See http://www.kr.tuwien.ac.at/staff/redl/wjflp. We note that the (unique) subset-maximal ideal set, the ideal extension, can be obtained using a further optimization constraint or using a more involved encoding.

[9] See http://www.kr.tuwien.ac.at/staff/redl/wjflp or [22].

**Table 4**
Multi-context systems benchmark results (time for well-justified FLP over ordinary FLP answer sets; speedup if $< 1$, slowdown if $> 1$).

| $c$ | $t^{\mathcal{M}}_{\text{avg}}(c)$ | $t^{\mathcal{M}}_{\text{max}}(c)$ | Slower instances | Faster instances | $p^{\mathcal{M}}_{\text{avg}}(c)$ | $p^{\mathcal{M}}_{\text{max}}(c)$ |
|---|---|---|---|---|---|---|
| 3 | 0.86 | 0.56 | 6.67% | 93.33% | 0.89 | 0.85 |
| 4 | 0.79 | 0.55 | 8.33% | 87.50% | 0.68 | 0.55 |
| 5 | 0.96 | 0.74 | 10.53% | 47.37% | 0.85 | 0.74 |
| 6 | 0.67 | 0.31 | 0.00% | 62.50% | 0.53 | 0.40 |
| 7 | 0.60 | 0.25 | 7.14% | 32.14% | 0.30 | 0.25 |
| 8 | 0.86 | 0.42 | 9.09% | 18.18% | 0.23 | 0.23 |
| 9 | 0.84 | 0.28 | 0.00% | 9.09% | 0.24 | 0.24 |

a row (stacking $m$ diamonds in a tower of $3m + 1$ contexts). Ordinary diamonds have, in contrast to zig-zag diamonds, no connection between the two middle contexts. A house consists of five nodes with six edges (the ridge context has directed edges to the two middle contexts, which form with the two base contexts a cycle with 4 edges); house stacks are subsequently built up by using the basement nodes as ridges for the next houses (thus, $m$ houses have $4m + 1$ contexts). Binary trees grow balanced, i.e., every level is complete except for the last level, which grows from the left-most context.

A parameter setting $(c, s, b, r)$ for an instance $M = (C_1, \ldots, C_c)$ from $\mathcal{M}$ specifies (i) the number $c$ of contexts, (ii) the local alphabet size $|\Sigma_i| = s$ (each $C_i$ has a random logic program on $s$ atoms with $2k$ answer sets, $0 \leq k \leq s/2$), (iii) the maximum interface size $b$ (number of atoms exported), and (iv) the maximum number $r$ of bridge rules per context, each having $\leq 2$ body literals. The benchmark set consists of instances with $c = 3, \ldots, 9$ contexts, each $|\Sigma_i| = 2$, $b = 1$, and $r = 2$. The instances have been created with the benchmark generator for DMCS [13], which is available from the benchmark homepage.

Table 4 summarizes the results grouped by the number of contexts $c$. We report the results only up to size 9 because all greater instances timeout under both semantics. We also report the percentage of the instances within each group of $c$ contexts that are faster resp. slower under the well-justified FLP semantics compared to the FLP semantics. For the remaining instances, we either could not observe a speedup or slowdown, or the runs timed out under both semantics.

As for the argumentation benchmarks, evaluation under the well-justified FLP semantics is mostly faster than under the FLP semantics if we measure the total runtime, but the speedup is smaller in this case. This is because the program structure does not allow for deriving as many literals deterministically as in the argumentation benchmark. This makes the fixpoint iteration more complex, as checking satisfaction of an external atom under a partial interpretation requires to consider all its possible completions.

Unlike in our argumentation benchmark, also the average runtime per answer set is for the well-justified FLP semantics smaller than for the FLP semantics. This is because in this benchmark most FLP answer sets are well-justified and the two semantics yield the same set of models in many cases. The better total runtime for the well-justified FLP semantics thus carries over to the average case.

**Set partitioning** This benchmark uses the following HEX-program:

$$sel(X) \leftarrow domain(X), \&diff[domain, nsel](X)$$

$$nsel(X) \leftarrow domain(X), \&diff[domain, sel](X)$$

$$\leftarrow sel(X), sel(Y), sel(Z), X \neq Y, X \neq Z, Y \neq Z$$

$$domain(1..N) \leftarrow$$

where $\&diff[p, q](X)$ computes the set of all elements $X$ which are in the extension of $p$ but not in the extension of $q$. It computes in its FLP answer sets all partitionings of a set into two (possibly empty) partitions where the first has size at most two, using an external atom for computing the set difference. In fact, each of the FLP answer sets is well-justified, and thus the two semantics coincide; this is because the derivation of any atom in an FLP answer set does not rely on other atoms except facts. Thus, fixpoint iteration can reproduce the answer set already in the first iteration.

The evaluation results are shown in Table 5. Note that we do not group benchmark instances in this case, thus the average and maximum speedup/slowdown is the same for each row. In this benchmark the computation under the well-justified FLP semantics is always faster than under the FLP semantics. This is because the constraints of kind $\neg sel(x)$ (resp. $\neg nsel(x)$) are added right at the beginning of the fixpoint iteration for all atoms which are not in the compatible set. This makes the corresponding atom $\&diff[domain, sel](x)$ (resp. $\&diff[domain, nsel](x)$) immediately satisfied in the first iteration. Thus, the fixpoint iteration always terminates after the first iteration, while the necessary minimality check for the FLP semantics is exponential. All FLP answer sets of this program are also well-justified FLP answer sets, thus the picture does not change if we measure the average runtime per answer set. For $c > 12$ the results do not change anymore.

**Table 5**
Set partitioning benchmark results (time for well-justified FLP over ordinary FLP answer sets; speedup if $< 1$, slowdown if $> 1$).

| $c$ | $t^{\mathcal{P}}_{\text{avg}}(c) = t^{\mathcal{P}}_{\text{max}}(c)$ | Slower instances | Faster instances | $p^{\mathcal{P}}_{\text{avg}}(c) = p^{\mathcal{P}}_{\text{max}}(c)$ |
|---|---|---|---|---|
| 1 | 0.67 | 0.00% | 100.00% | 0.67 |
| 2 | 0.73 | 0.00% | 100.00% | 0.74 |
| 3 | 0.67 | 0.00% | 100.00% | 0.67 |
| 4 | 0.24 | 0.00% | 100.00% | 0.24 |
| 5 | 0.07 | 0.00% | 100.00% | 0.07 |
| 6 | 0.02 | 0.00% | 100.00% | 0.02 |
| 7 | 0.02 | 0.00% | 100.00% | $< 0.005$ |
| 8 | 0.04 | 0.00% | 100.00% | $< 0.005$ |
| 9 | 0.10 | 0.00% | 100.00% | $< 0.005$ |
| 10 | 0.24 | 0.00% | 100.00% | $< 0.005$ |
| 11 | 0.56 | 0.00% | 100.00% | 0.01 |
| 12 | 1.00 | 0.00% | 0.00% | 1.00 |

## 9. Related work

The FLP answer set semantics, in the spirit of *minimal models of FLP reducts*, was first introduced in [26,27] for normal and disjunctive logic programs with aggregates. This method of defining answer sets has further been applied to description logic programs and HEX-programs [25,24], tightly coupled dl-programs [46], modular logic programs [12], etc. Since FLP reducts are treated as classical implications instead of rules, such FLP answer sets suffer from possible circular justifications (see Examples 1 and 15; Shen and Wang [55,56] illustrated the circular justification problem with the FLP answer set semantics of Lukasiewicz [46]).

For logic programs with first-order formulas, Bartholomew et al. [4] reformulated the FLP answer set semantics of Definition 2 in terms of a modified form of circumscription. Unlike Definition 2, this reformulation refers to no program grounding and employs no SNA assumption. As shown in Example 2, this FLP answer set semantics suffers from the circular justification problem.

Ferraris [29] defined answer sets for logic programs with propositional formulas and aggregates based on a new definition of equilibrium logic [51]. Ferraris et al. [30] further extended this answer set semantics to first-order formulas in terms of a modified circumscription. Pearce [52] proposed to identify answer sets with equilibrium models in equilibrium logic. de Bruijn et al. [15] further applied the semantics of Pearce [52], while Lee and Palla [42] applied the semantics of Ferraris et al. [30], to integrate rules and ontologies for the Semantic Web. It turns out that the answer set semantics of Pearce [52] coincides with that of Ferraris [29] in the propositional case and with that of Ferraris et al. [30] in the first-order case. We observe that these answer set semantics also suffer from circular justifications. As an example, for the propositional logic program $\Pi = \{p \leftarrow \neg\neg p\}$, $I = \{p\}$ is neither a well-justified FLP answer set nor an FLP answer set (Definition 2); however, $I$ is an answer set under the semantics of Ferraris [29], Ferraris et al. [30] and Pearce [52]. This answer set has a circular justification caused by the self-supporting loop $p \Leftarrow \neg\neg p \Leftarrow p$, i.e. $p$ being in $I$ is due to $I$ satisfying $\neg\neg p$, which in turn is due to $p$ being in $I$.

The well-justified FLP answer set semantics inherits the anti-chain property of the FLP answer set semantics, i.e. no well-justified FLP answer set is a proper subset of another well-justified FLP answer set (see Theorem 4); in contrast, none of the semantics of Ferraris [29], Ferraris et al. [30] and Pearce [52] has this property. As an alternative, Pearce [52] further proposed to use only minimal equilibrium models to define answer sets (see Section 6.1 of Pearce [52]). However, it turns out that applying the minimization method does not overcome the circular justification problem. To illustrate, consider the propositional logic program $\Pi = \{p \leftarrow \neg\neg p,\ \ p \leftarrow \neg p\}$. $I = \{p\}$ is not an FLP answer set of $\Pi$, but it is a minimal equilibrium model and thus is an answer set under the semantics of Pearce [52], Ferraris [29] and Ferraris et al. [30]. This answer set has a circular justification $p \Leftarrow \neg\neg p \Leftarrow p$.

The above examples show that an answer set of Ferraris [29], Ferraris et al. [30] and Pearce [52] is not necessarily an FLP or a well-justified FLP answer set; the following example illustrates that also the converse direction fails. Consider the propositional logic program $\Pi = \{p \leftarrow p \vee \neg p\}$. Since $p \vee \neg p$ is a tautology in classical logic, $I = \{p\}$ is both an FLP and a well-justified FLP answer set of $\Pi$. However, under the semantics of Pearce [52], Ferraris [29] and Ferraris et al. [30], $\Pi$ is identified with the normal logic program $\Pi' = \{p \leftarrow p,\ \ p \leftarrow \neg p\}$, thus $I = \{p\}$ is not an answer set of these semantics.

Truszczyński [63] defined an answer set semantics for logic programs with propositional formulas by introducing a different program transformation called FLPT reducts, which agrees with the FLP answer set semantics of Faber et al. [26,27] for normal and disjunctive logic programs. Such answer sets may also have circular justifications. For instance, the interpretation $I = \{p(1), p(-1)\}$ of $\Pi_2$ in Example 2, which has circular justifications, is an answer set under the semantics of Truszczyński [63]. Moreover, this semantics does not share the anti-chain property of the FLP answer set semantics. Hence, answer sets of Truszczyński [63] are neither FLP answer sets nor well-justified FLP answer sets in general. The following example (borrowed from [4]) disproves a converse inclusion. For a logic program $\Pi = \{\neg\neg p,\ \ p \vee \neg p \leftarrow \neg\neg p\}$, $I = \{p\}$ is both an FLP and a well-justified FLP answer set of $\Pi$, but $I$ is not an answer set under the semantics of Truszczyński [63].

For a logic program $\Pi$ whose rule heads are atoms, in [16,53] a three-valued fixpoint semantics was introduced based on three-valued operators. This fixpoint semantics defines answer sets, called two-valued stable models, which are free of circular justifications. As discussed in Section 5.1, there are at least three significant differences between the three-valued fixpoint semantics and the well-justified FLP answer set semantics. That is, the former is defined over three-valued interpretations, while the latter is defined over two-valued interpretations; the former is applicable only to logic programs whose rule heads are atoms, while the latter applies to logic programs whose rule heads are arbitrary first-order formulas; and as shown by Theorem 8, the former is more conservative than the latter in the sense that two-valued stable models of the three-valued fixpoint semantics are well-justified FLP answer sets, which by Corollary 2 are also FLP answer sets, but the converse does not hold. This means that the two-valued stable models may exclude some FLP answer sets that are free of circular justifications.

For normal logic programs with c-atoms or positive basic logic programs, in [60,59] it was shown that the conditional satisfaction-based answer set semantics agrees with the three-valued fixpoint semantics of Denecker et al. [16] and of Pelov et al. [53]. By Theorem 9, for such logic programs the well-justified FLP answer set semantics also agrees with the three-valued fixpoint semantics. Shen and You [57] gave an alternative characterization of the conditional satisfaction-based semantics in terms of a generalized Gelfond–Lifschitz transformation. Liu et al. [45] proposed a computation-based answer set semantics for normal logic programs with c-atoms, which proves to coincide with the conditional satisfaction-based semantics.

## 10. Summary and future work

The FLP answer set semantics [26,27] has been widely used to define answer sets for different types of logic programs. However, when being extended from normal logic programs to more general classes of logic programs, the FLP answer set semantics suffers from circular justifications. The intuitive reason behind the circular justification problem is that the FLP answer set semantics does not induce a level mapping for its answer sets. In this paper, we have overcome this shortcoming by enhancing the FLP answer set semantics with a suitable level mapping.

Inspired by the fact that each answer set $I$ of a normal logic program $\Pi$ under the standard answer set semantics has a level mapping that is induced by the fixpoint construction of $I$ using the van Emden–Kowalski one-step provability operator $T_{\Pi^I}(S)$ for the Gelfond–Lifschitz reduct $\Pi^I$, we define well-justified FLP answer sets $I$ of a general logic program $\Pi$ as fixpoints that are obtained by iteratively applying an extended van Emden–Kowalski operator $T_{f\Pi^I}(O, N)$ for the FLP reduct $f\Pi^I$; such FLP answer sets always have a level mapping and are thus free of circular justifications. As a generic approach, the well-justified answer set semantics applies to logic programs with first-order formulas, logic programs with aggregates or c-atoms, and description logic programs. It can easily be extended to other well-known types of logic programs, such as HEX-programs, tightly coupled dl-programs and modular logic programs, by a suitable adjustment of the satisfaction relation. To the best of our knowledge, the answer set semantics presented here is the first that is free of circular justifications for such general kinds of logic programs.

We have studied in depth the computational complexity of the FLP and the well-justified FLP answer set semantics for general logic programs. For the major reasoning tasks, the FLP and the well-justified FLP answer set semantics fall in the same complexity classes. This means that the well-justified FLP answer set semantics enhances the FLP answer set semantics with a level mapping formalism without affecting the worst-case complexity.

We have implemented the well-justified FLP answer set semantics by extending the ASP reasoner DLVHEX, which currently can compute well-justified FLP answer sets for normal logic programs with aggregates, dl-programs and HEX-programs. We also conducted an experimental evaluation, which shows on benchmark problems the potential of the well-justified FLP answer set semantics in two respects: it not only employs a stronger notion of foundedness than the FLP answer set semantics, but it is also faster to compute (due to its fixpoint design, which is beneficial for answer set checking). To find some FLP answer set, it thus seems attractive to start the search by finding a well-justified FLP answer set.

**Open issues** We focused in this article on logic programs with rules of the form $H \leftarrow B$, where $H$ and $B$ are first-order formulas, possibly with aggregates and/or dl-atoms. Such logic programs do not cover *disjunctive logic programs* introduced in [34], which consist of rules of the form $A_1 \mid \cdots \mid A_l \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n$, where each $A_i$, $B_i$ and $C_i$ is an atom, and $\mid$ is an *epistemic disjunction* operator that is different from the classical disjunction connective $\vee$ (see [31] for their differences). As future work, it is interesting to extend the well-justified FLP answer set semantics to logic programs with rules of the form $H_1 \mid \cdots \mid H_l \leftarrow B$, where $B$ and each $H_i$ are first-order formulas. In connection with this, it remains to deploy well-justified answer sets to further classes of logic programs.

On the computational side, a study of the decidability and computational complexity of first-order logic programs with formulas from various decidable fragments of first-order logic, under different notions of answer sets (including well-justified FLP answer sets) is an interesting issue. Moreover, to develop methods for further improving the efficiency of the current implementation of the well-justified FLP answer set semantics is a challenging task.

Finally, it is of practical significance to exploit real-world applications where the well-justified FLP answer set semantics yields more intuitive results than the existing state-of-the-art answer set semantics.

**Acknowledgements**

We would like to thank all anonymous reviewers for their constructive comments, which helped to significantly improve this paper. This work is supported in part by China National 973 program 2014CB340301 and NSFC grant 61379043, the Austrian Science Fund (FWF) project P24090, and the Australian Research Council (ARC) under DP1093652 and DP130102302.

## Appendix A. Proofs

**Proof of Proposition 1.** ($\Longrightarrow$) Assume that $S = \langle S_0, S_1, \cdots, S_m\rangle$ is a level mapping of $I$ under Definition 4. Since all rule heads of $\Pi$ are atoms, $S' = \langle S_1, \cdots, S_m\rangle$ is a level mapping of rule heads of $\Pi$ w.r.t. $I$. Since $I$ is a model of $\Pi$ and $\bigcup_{1\leq i\leq m} S_i = I$, there is no rule $r$ in $ground(\Pi)$ such that $head(r)$ is not in $\bigcup_{1\leq i\leq m} S_i$ and $body(r)$ is true in $\bigcup_{0\leq i\leq m} S_i$ (otherwise, $I$ would not be a model of $\Pi$). This means $S'$ is a total level mapping of rule heads of $\Pi$ w.r.t. $I$. For $k = 1, \cdots, m$, let $l_k = k$. Obviously, for every $A \in S_k$ where $k > 0$, $\bigcup_{0\leq i\leq l_k} S_i \models A$ but $\bigcup_{0\leq i\leq l_k-1} S_i \not\models A$. Therefore, $S$ is a level mapping of $I$ under Definition 7.

($\Longleftarrow$) Assume that $S = \langle S_0, S_1, \cdots, S_m\rangle$ is a level mapping of $I$ under Definition 7. Let $S'_0 = I^-$. Then there is a total level mapping $S' = \langle S'_1, \cdots, S'_n\rangle$ of rule heads of $\Pi$ w.r.t. $I$ and integers $l_1, \cdots, l_m$ where $1 \leq l_1 < \cdots < l_m \leq n$ such that for every $A \in S_k$ where $k > 0$, $\bigcup_{0\leq i\leq l_k} S'_i \models A$ but $\bigcup_{0\leq i\leq l_k-1} S'_i \not\models A$. Since all rule heads of $\Pi$ are atoms, for each $k > 0$ $S'_k$ consists of atoms. By the fact that for every $A \in S_k$, $\bigcup_{0\leq i\leq l_k} S'_i \models A$ but $\bigcup_{0\leq i\leq l_k-1} S'_i \not\models A$, it follows that for every $k = 1, \cdots, m$, $S_k \subseteq S'_{l_k}$. Since $\bigcup_{1\leq i\leq m} S_i = I$, $I$ is a subset of $\bigcup_{1\leq i\leq n} S'_i$; since $I$ is a model of $\Pi$, $\bigcup_{1\leq i\leq n} S'_i$ must be a subset of $I$; hence $\bigcup_{1\leq i\leq n} S'_i = I$. By the fact that $\bigcup_{1\leq i\leq m} S_i = I$, that $\bigcup_{1\leq k\leq m} S'_{l_k} \subseteq I$, and that for every $k = 1, \cdots, m$, $S_k \subseteq S'_{l_k}$, it follows that for every $k = 1, \cdots, m$, $S_k = S'_{l_k}$. This means $m = n$ and $l_k = k$ for every $k > 0$. Thus $S' = \langle S_1, \cdots, S_m\rangle$. Therefore $\langle S_1, \cdots, S_m\rangle$ is also a total level mapping of rule heads of $\Pi$ w.r.t. $I$. By Definition 6, for every $A \in S_k$ where $k > 0$, there is a rule $A \leftarrow body(r)$ in $ground(\Pi)$ such that $body(r)$ is true in $\bigcup_{0\leq i\leq k-1} S_i$. Hence $S$ is a level mapping of $I$ under Definition 4. □

**Proof of Proposition 2.** ($\Longrightarrow$) When $I$ has a level mapping, it is immediate from Definition 7 that there is a total level mapping $S' = \langle S'_1, \cdots, S'_n\rangle$ of rule heads of $\Pi$ w.r.t. $I$ such that $\bigcup_{0\leq i\leq n} S'_i \models A$ for every $A \in I$.

($\Longleftarrow$) Assume that there is a total level mapping $S' = \langle S'_1, \cdots, S'_n\rangle$ of rule heads of $\Pi$ w.r.t. $I$ such that $\bigcup_{0\leq i\leq n} S'_i \models A$ for every $A \in I$. For $k = 1, \cdots, n$, let $R_k$ consist of all $A \in I$ such that $\bigcup_{0\leq i\leq k} S'_i \models A$ but $\bigcup_{0\leq i\leq k-1} S'_i \not\models A$. Then $\bigcup_{1\leq i\leq n} R_i = I$. Among $R_1, \cdots, R_n$, assume only $R_{l_1}, \cdots, R_{l_m}$ are nonempty, where $1 \leq l_1 < \cdots < l_m \leq n$. For $k = 1, \cdots, m$, let $S_k = R_{l_k}$. Then by Definition 7, $S = \langle S_0, S_1, \cdots, S_m\rangle$ is a level mapping of $I$. □

**Proof of Lemma 1.** We prove the claim by induction on $i \geq 0$. It clearly holds for $i = 0$. For the induction step, assume that $I$ is a model of $T^i_\Pi(\emptyset, \neg I^-)$; we prove that $I$ is then also a model of $T^{i+1}_\Pi(\emptyset, \neg I^-)$.

Let $S = T^{i+1}_\Pi(\emptyset, \neg I^-) \setminus T^i_\Pi(\emptyset, \neg I^-)$. For each formula $H \in S$, there is a rule $r \in ground(\Pi)$ with $head(r) = H$ such that $T^i_\Pi(\emptyset, \neg I^-) \cup \neg I^- \models body(r)$. By the induction hypothesis, $I$ is a model of $T^i_\Pi(\emptyset, \neg I^-) \cup \neg I^-$, so $I$ is a model of $body(r)$. Since $I$ is a model of $\Pi$, $I$ is a model of $r$ and thus is a model of $H$. This shows that $I$ is a model of $S$, hence a model of $T^{i+1}_\Pi(\emptyset, \neg I^-)$. □

**Proof of Theorem 2.** We show that for every $r \in ground(\Pi) \setminus f\Pi^I$ and $i \geq 0$, it holds that $T^i_\Pi(\emptyset, \neg I^-) \cup \neg I^- \not\models body(r)$; hence $T^i_\Pi(\emptyset, \neg I^-) = T^i_{f\Pi^I}(\emptyset, \neg I^-)$ for all $i \geq 0$, which proves the result. Assume towards a contradiction that $T^i_\Pi(\emptyset, \neg I^-) \cup \neg I^- \models body(r)$. As by Lemma 1, $I$ is a model of $T^i_\Pi(\emptyset, \neg I^-) \cup \neg I^-$, it follows that $I$ satisfies $body(r)$. However, this means $r \in f\Pi^I$, which is a contradiction. □

**Proof of Theorem 3.** We prove it by induction on $i \geq 0$. It is trivial for $i = 0$. As induction step, assume that for some integer $n$, $T^n_{\Pi^I}(\emptyset) = T^n_\Pi(\emptyset, \neg I^-)$. We next show that this claim holds for $n + 1$.

For any rule $r \in \Pi^I$ such that $body(r)$ is satisfied by $T^n_{\Pi^I}(\emptyset)$, by definition of $\Pi^I$ there must be a rule $r' \in ground(\Pi)$ such that $head(r) = head(r')$ and $body(r)$ is $body(r')$ with all negative literals in $\neg I^-$ removed. This means all positive literals of $body(r')$ are in $T^n_{\Pi^I}(\emptyset)$ and all negative literals are in $\neg I^-$. By the induction hypothesis, $T^n_\Pi(\emptyset, \neg I^-) \cup \neg I^- \models body(r')$. This shows $T^{n+1}_{\Pi^I}(\emptyset) \subseteq T^{n+1}_\Pi(\emptyset, \neg I^-)$. Conversely, let $r' \in ground(\Pi)$ be a rule such that $T^n_\Pi(\emptyset, \neg I^-) \cup \neg I^- \models body(r')$. Since $I$ is a model of $\Pi$, by Lemma 1, $I$ is a model of $T^n_\Pi(\emptyset, \neg I^-) \cup \neg I^-$ and thus $I$ satisfies $body(r')$. This means (1) $T^n_\Pi(\emptyset, \neg I^-) \subseteq I$; (2) there is a rule $r \in \Pi^I$ such that $head(r) = head(r')$ and $body(r)$ is $body(r')$ with all negative literals removed; and (3) $body(r)$ is satisfied by $T^n_\Pi(\emptyset, \neg I^-)$. By the induction hypothesis, $body(r)$ is satisfied by $T^n_{\Pi^I}(\emptyset)$. This shows $T^{n+1}_\Pi(\emptyset, \neg I^-) \subseteq T^{n+1}_{\Pi^I}(\emptyset)$; hence $T^{n+1}_\Pi(\emptyset, \neg I^-) = T^{n+1}_{\Pi^I}(\emptyset)$. Consequently, for any $i \geq 0$ $T^i_{\Pi^I}(\emptyset) = T^i_\Pi(\emptyset, \neg I^-)$ and thus $lfp(T_{\Pi^I}(\emptyset)) = lfp(T_\Pi(\emptyset, \neg I^-))$. □

**Proof of Theorem 4.** Let $I$ be an answer set of a logic program $\Pi$, and assume, on the contrary, that $J \subset I$ is a minimal model of $\Pi$. Then, $\neg I^- \subset \neg J^-$. Since the entailment relation $\models$ is monotone, for every $i \geq 0$, $T^i_\Pi(\emptyset, \neg I^-) \subseteq T^i_\Pi(\emptyset, \neg J^-)$

and thus $lfp(T_\Pi(\emptyset, \neg I^-)) \subseteq lfp(T_\Pi(\emptyset, \neg J^-))$. By Theorem 2, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \subseteq lfp(T_{f\Pi^J}(\emptyset, \neg J^-))$. Since $I$ is an answer set, for each $A \in I$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$ and thus $lfp(T_{f\Pi^J}(\emptyset, \neg J^-)) \cup \neg J^- \models A$. Since $I \cap J^- \neq \emptyset$, this implies $lfp(T_{f\Pi^J}(\emptyset, \neg J^-)) \cup \neg J^-$ is inconsistent. This contradicts Lemma 1 that $J$ is a model of $lfp(T_{f\Pi^J}(\emptyset, \neg J^-))$. We then conclude that $I$ is a minimal model of $\Pi$.

For the second part, assume, on the contrary, that $J \subset I$ is a minimal model of $f\Pi^I$. Then, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \subseteq lfp(T_{f\Pi^I}(\emptyset, \neg J^-))$. Since $I$ is an answer set, for each $A \in I$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$ and thus $lfp(T_{f\Pi^I}(\emptyset, \neg J^-)) \cup \neg J^- \models A$. Since $I \cap J^- \neq \emptyset$, this implies $lfp(T_{f\Pi^I}(\emptyset, \neg J^-)) \cup \neg J^-$ is inconsistent. This contradicts Lemma 1 that $J$ is a model of $lfp(T_{f\Pi^I}(\emptyset, \neg J^-))$. We then conclude that $I$ is a minimal model of $f\Pi^I$. □

**Proof of Theorem 5.** Let $\Pi$ be a logic program and $I$ an answer set of $\Pi$. Let $S'_0 = \neg I^-$ and $S' = \langle S'_1, \cdots, S'_n \rangle$ be a partitioning of rule heads of $\Pi$, where for every $k > 0$ $S'_k = T^k_{f\Pi^I}(\emptyset, \neg I^-) \setminus T^{k-1}_{f\Pi^I}(\emptyset, \neg I^-)$. So for every $k > 0$, $\bigcup_{1 \le i \le k} S'_i = T^k_{f\Pi^I}(\emptyset, \neg I^-)$, and $\bigcup_{1 \le i \le n} S'_i = T^n_{f\Pi^I}(\emptyset, \neg I^-) = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$. As $I$ is an answer set of $\Pi$, the set $\bigcup_{1 \le i \le n} S'_i$ consists of all rule heads $head(r)$ in $ground(\Pi)$ such that $body(r)$ is true in $\bigcup_{0 \le i \le n} S'_i$. Furthermore for every $H \in S'_k$ where $k > 0$, there is a rule $r \in f\Pi^I \subseteq ground(\Pi)$ with $head(r) = H$ whose body is true in $T^{k-1}_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- = \bigcup_{0 \le i \le k-1} S'_i$. By Definition 6, $S'$ is a total level mapping of rule heads of $\Pi$ w.r.t. $I$. Since $I$ is an answer set, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$ and thus $\bigcup_{0 \le i \le n} S'_i \models A$ for every $A \in I$. By Proposition 2, $I$ has a level mapping as in Definition 7. □

**Proof of Lemma 2.** Let $S'_0 = \neg I^-$. Since $S' = \langle S'_1, \cdots, S'_m \rangle$ is a total level mapping of rule heads of $\Pi$ w.r.t. $I$, the set $\bigcup_{1 \le i \le m} S'_i$ consists of all rule heads $head(r)$ in $ground(\Pi)$ such that $\bigcup_{0 \le i \le m} S'_i \models body(r)$. As $I$ is a model of $\Pi$, it is by Theorem 2 sufficient to prove that $lfp(T_\Pi(\emptyset, \neg I^-)) = \bigcup_{1 \le i \le m} S'_i$.

We first prove that $lfp(T_\Pi(\emptyset, \neg I^-)) \subseteq \bigcup_{1 \le i \le m} S'_i$. To this end, we show by induction on $j \ge 1$ that $T^j_\Pi(\emptyset, \neg I^-) \subseteq \bigcup_{1 \le i \le m} S'_i$. For the base case $j = 1$, since $S'_0 = \neg I^-$, the set $\bigcup_{1 \le i \le m} S'_i$ includes all rule heads $head(r)$ in $ground(\Pi)$ such that $\neg I^- \models body(r)$. This means $T^1_\Pi(\emptyset, \neg I^-) \subseteq \bigcup_{1 \le i \le m} S'_i$. For the induction step, assume that $T^j_\Pi(\emptyset, \neg I^-) \subseteq \bigcup_{1 \le i \le m} S'_i$ holds for $j \ge 1$. We next show that this claim holds for $j + 1$.

Let $H$ be a rule head in $T^{j+1}_\Pi(\emptyset, \neg I^-) \setminus T^j_\Pi(\emptyset, \neg I^-)$. Then there must be a rule $r \in ground(\Pi)$ with $head(r) = H$ such that $T^j_\Pi(\emptyset, \neg I^-) \cup \neg I^- \models body(r)$. By the induction hypothesis, $\bigcup_{0 \le i \le m} S'_i \models body(r)$. Hence $H$ is in $\bigcup_{1 \le i \le m} S'_i$. This shows that for every $j > 0$, $T^j_\Pi(\emptyset, \neg I^-) \subseteq \bigcup_{1 \le i \le m} S'_i$. Hence it follows that $lfp(T_\Pi(\emptyset, \neg I^-)) \subseteq \bigcup_{1 \le i \le m} S'_i$.

Next we show that $\bigcup_{1 \le i \le j} S'_i \subseteq lfp(T_\Pi(\emptyset, \neg I^-))$ by induction on $j \ge 1$. For the base case $j = 1$, by Definition 6, for each $H \in S'_1$ there must exist a rule $r \in ground(\Pi)$ with $head(r) = H$ such that $\neg I^- \models body(r)$. Then $H$ must be in $T^1_\Pi(\emptyset, \neg I^-)$ and thus $S'_1 \subseteq lfp(T_\Pi(\emptyset, \neg I^-))$. For the induction step, assume that for $j \ge 1$, $\bigcup_{1 \le i \le j} S'_i \subseteq lfp(T_\Pi(\emptyset, \neg I^-))$. We next show that this claim holds for $j + 1$. For each $H \in S'_{j+1}$ there must exist a rule $r \in ground(\Pi)$ with $head(r) = H$ such that $\bigcup_{1 \le i \le j} S'_i \cup \neg I^- \models body(r)$. By the induction hypothesis, $lfp(T_\Pi(\emptyset, \neg I^-)) \cup \neg I^- \models body(r)$ and thus $H$ is in $lfp(T_\Pi(\emptyset, \neg I^-))$. This shows that $\bigcup_{1 \le i \le j+1} S'_j \subseteq lfp(T_\Pi(\emptyset, \neg I^-))$, and it follows that $\bigcup_{1 \le i \le m} S'_i \subseteq lfp(T_\Pi(\emptyset, \neg I^-))$. □

**Proof of Theorem 6.** ($\Longrightarrow$) When $I$ is a well-justified FLP answer set, by Theorem 5 it has a level mapping as in Definition 7.

($\Longleftarrow$) Let $I$ be an FLP answer set of $\Pi$ that has a level mapping $S = \langle S_0, S_1, \cdots, S_m \rangle$ as in Definition 7, where $S_0 = \neg I^-$ and $\bigcup_{1 \le i \le m} S_i = I$. Let $S'_0 = \neg I^-$; then there is a total level mapping $S' = \langle S'_1, \cdots, S'_n \rangle$ of rule heads of $\Pi$ w.r.t. $I$ and integers $l_1, \cdots, l_m$, where $1 \le l_1 < \cdots < l_m \le n$, such that for every $A \in S_k$ where $k > 0$, $\bigcup_{0 \le i \le l_k} S'_i \models A$ but $\bigcup_{0 \le i \le l_k - 1} S'_i \not\models A$. Then for every $A \in I$, $\bigcup_{0 \le i \le n} S'_i \models A$. By Lemma 2, for every $A \in I$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$. By Definition 9, $I$ is a well-justified FLP answer set of $\Pi$. □

**Proof of Theorem 7.** By Theorem 4, when $I$ is a well-justified FLP answer set of $\Pi$, $I$ is a minimal model of $\Pi$.

Conversely, assume that $I$ is a minimal model of $\Pi$. Then for each $A \in I$, $ground(\Pi) \cup \neg I^- \models A$. Since all rule bodies in $\Pi$ are empty, $lfp(T_\Pi(\emptyset, \neg I^-)) = ground(\Pi)$ and by Theorem 2, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = ground(\Pi)$. This means that for each $A \in I$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$. By Definition 9, $I$ is a well-justified FLP answer set of $\Pi$. This establishes the first equivalence. From this and since every well-justified FLP answer of an arbitrary logic program $\Pi$ is an FLP answer of $\Pi$ (Corollary 2) and every FLP answer set of $\Pi$ as a minimal model of $f\Pi^I$ also must be a minimal model of $\Pi$, the second equivalence follows. □

To prove Theorem 8, we introduce the following lemma.

**Lemma 4.** *Let $\hat{I} = (I_1, I_2)$ be a three-valued interpretation of a propositional logic program $\Pi$ and $F$ be a propositional formula. If $\hat{I}(F) = \mathbf{t}$, then $I_1 \cup \neg I_2^- \models F$; if $\hat{I}(F) = \mathbf{f}$, then $I_1 \cup \neg I_2^- \models \neg F$.*

**Proof.** Note that $I_2^- = \mathcal{HB}_\Pi \setminus I_2$; so for every atom $p \in I_2^-$, $\hat{I}(p) = \mathbf{f}$. We prove this lemma by induction on the logical connectives $\wedge, \vee, \neg$.

Induction base: Let $p$ be an propositional atom. If $\hat{I}(p) = \mathbf{t}$, then $p \in I_1$ and thus $I_1 \cup \neg I_2^- \models p$. If $\hat{I}(p) = \mathbf{f}$, then $p \in I_2^-$ and thus $I_1 \cup \neg I_2^- \models \neg p$.

Induction hypothesis: Assume that $\phi$ and $\psi$ are two arbitrary propositional formulas that satisfy the conditions of Lemma 4. We next prove that the formulas $\phi \wedge \psi$, $\phi \vee \psi$ and $\neg\phi$ also satisfy the conditions.

Induction step:

$\wedge$: If $\hat{I}(\phi \wedge \psi) = \mathbf{t}$, then $\hat{I}(\phi) = \mathbf{t}$ and $\hat{I}(\psi) = \mathbf{t}$. By induction hypothesis, $I_1 \cup \neg I_2^- \models \phi$ and $I_1 \cup \neg I_2^- \models \psi$. Thus, $I_1 \cup \neg I_2^- \models \phi \wedge \psi$. If $\hat{I}(\phi \wedge \psi) = \mathbf{f}$, then $\hat{I}(\phi) = \mathbf{f}$ or $\hat{I}(\psi) = \mathbf{f}$. By induction hypothesis, $I_1 \cup \neg I_2^- \models \neg\phi$ or $I_1 \cup \neg I_2^- \models \neg\psi$. This means $I_1 \cup \neg I_2^- \models \neg\phi \vee \neg\psi$, i.e. $I_1 \cup \neg I_2^- \models \neg(\phi \wedge \psi)$.

$\vee$: If $\hat{I}(\phi \vee \psi) = \mathbf{t}$, then $\hat{I}(\phi) = \mathbf{t}$ or $\hat{I}(\psi) = \mathbf{t}$. By induction hypothesis, $I_1 \cup \neg I_2^- \models \phi$ or $I_1 \cup \neg I_2^- \models \psi$, i.e. $I_1 \cup \neg I_2^- \models \phi \vee \psi$. If $\hat{I}(\phi \vee \psi) = \mathbf{f}$, then $\hat{I}(\phi) = \mathbf{f}$ and $\hat{I}(\psi) = \mathbf{f}$. By induction hypothesis, $I_1 \cup \neg I_2^- \models \neg\phi$ and $I_1 \cup \neg I_2^- \models \neg\psi$, i.e. $I_1 \cup \neg I_2^- \models \neg\phi \wedge \neg\psi$. Thus $I_1 \cup \neg I_2^- \models \neg(\phi \vee \psi)$.

$\neg$: If $\hat{I}(\neg\phi) = \mathbf{t}$, then $\hat{I}(\phi) = \mathbf{f}$. By induction hypothesis, $I_1 \cup \neg I_2^- \models \neg\phi$. If $\hat{I}(\neg\phi) = \mathbf{f}$, then $\hat{I}(\phi) = \mathbf{t}$. By induction hypothesis, $I_1 \cup \neg I_2^- \models \phi$. □

**Proof of Theorem 8.** Let $\hat{I} = (I, I)$ be a three-valued interpretation of $\Pi$. Since $I$ is a two-valued stable model of $\Pi$, we have $lfp(St_\Phi(I, I)) = (I, I)$. Then there is an iteration sequence of the operator $\Phi_\Pi$ w.r.t. $\hat{I}$:

$$x_0 = \emptyset, \qquad x_1 = \Phi_\Pi^1(x_0, I), \qquad \cdots, \qquad x_{i+1} = \Phi_\Pi^1(x_i, I), \qquad \cdots, \qquad x_\alpha = I$$

where $x_\alpha = I$ is the fixpoint $St_\Phi^\downarrow(I)$. Consider the following iteration sequence of the extended van Emden–Kowalski operator $T_\Pi$ w.r.t. $I$:

$$y_0 = \emptyset, \qquad y_1 = T_\Pi\big(y_0, \neg I^-\big), \qquad \cdots, \qquad y_{i+1} = T_\Pi\big(y_i, \neg I^-\big), \qquad \cdots, \qquad y_\beta = lfp\big(T_\Pi\big(\emptyset, \neg I^-\big)\big)$$

where $y_\beta = lfp(T_\Pi(\emptyset, \neg I^-))$ is the fixpoint. We next prove by induction that for every $i \geq 0$, $x_i \subseteq y_i$.

As induction base, for $i = 0$, $x_0 \subseteq y_0$. As induction hypothesis, assume that for some $i \geq 0$, $x_i \subseteq y_i$. We next prove $x_{i+1} \subseteq y_{i+1}$.

Let $\hat{J} = (x_i, I)$ be a three-valued interpretation. We have

$$x_{i+1} = \Phi_\Pi^1(x_i, I) = \big\{head(r) \;\big|\; r \in \Pi \text{ and } \hat{J}(body(r)) = \mathbf{t}\big\}, \quad \text{and}$$
$$y_{i+1} = T_\Pi(y_i, \neg I^-) = \big\{head(r) \;\big|\; r \in \Pi \text{ and } y_i \cup \neg I^- \models body(r)\big\}.$$

By Lemma 4, $\hat{J}(body(r)) = \mathbf{t}$ implies $x_i \cup \neg I^- \models body(r)$. By induction hypothesis that $x_i \subseteq y_i$, then $y_i \cup \neg I^- \models body(r)$. This shows that every $head(r)$ in $x_{i+1}$ is in $y_{i+1}$, i.e. $x_{i+1} \subseteq y_{i+1}$. This means $x_\alpha \subseteq y_\beta$ and thus $I \subseteq lfp(T_\Pi(\emptyset, \neg I^-))$.

Since $I$ is a model of $\Pi$, by Lemma 1 the sequence $y_0, y_1, \cdots, y_i, \cdots$ will not exceed $I$, i.e., $lfp(T_\Pi(\emptyset, \neg I^-)) \subseteq I$. Consequently, $lfp(T_\Pi(\emptyset, \neg I^-)) = I$ and by Theorem 2, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = I$. Thus $I$ is a well-justified FLP answer set of $\Pi$. □

**Proof of Lemma 3.** Let $A = (V, C)$. $R \models_I A$ if and only if for every $F$ with $R \cap V \subseteq F \subseteq I \cap V$, $F \in C$ if and only if for every $F$ with $R \cap V \subseteq F \subseteq I \cap V$, $F$ satisfies $A$ if and only if $R \cup \neg I^- \models A$. □

**Proof of Theorem 9.** Let $I$ be a model of a positive basic logic program $\Pi$. By Lemma 3, for every $R \subseteq I$ and any rule $r \in \Pi$, $R \models_I body(r)$ if and only if $R \cup \neg I^- \models body(r)$. This means for every $i \geq 0$, $T_\Pi^i(\emptyset, \neg I^-) = \Gamma_\Pi^i(\emptyset, I)$ and thus $lfp(T_\Pi(\emptyset, \neg I^-)) = lfp(\Gamma_\Pi(\emptyset, I))$. By Theorem 2, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = lfp(\Gamma_\Pi(\emptyset, I))$. Since $\Pi$ is a positive basic logic program, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ consists of ground atoms. Since $I$ is a model of $\Pi$, by Lemma 1 $I$ is a model of $lfp(T_\Pi(\emptyset, \neg I^-))$ and thus a model of $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$. This means $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ is disjoint from $I^-$. Therefore, $I$ is a well-justified FLP answer set of $\Pi$ if and only if for each $A \in I$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models A$ if and only if for each $A \in I$, $A \in lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ if and only if $I = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$ if and only if $I = lfp(\Gamma_\Pi(\emptyset, I))$ if and only if $I$ is a conditional satisfaction-based answer set of $\Pi$. □

**Proof of Theorem 10.** By Corollary 2, a well-justified FLP answer set for a dl-program is an FLP answer set. Eiter et al. [24] have shown that a strong answer set is a weak answer set.

Let $I$ be an FLP answer set of a dl-program $\Pi$ relative to a DL knowledge base $L$. Then, $I$ is a minimal model of the FLP reduct $f\Pi_L^I$. Consider the reduct $s\Pi_L^I$, which is $f\Pi_L^I$ with all negative literals and all nonmonotonic dl-atoms removed. Assume, on the contrary, that $I$ is not a strong answer set of $\Pi$; i.e., $I$ is not the least model of $s\Pi_L^I$. Let $J \subset I$ be the least model of $s\Pi_L^I$. Then, $f\Pi_L^I$ is inconsistent in $J$; i.e., there is a rule $r$ in $f\Pi_L^I$ such that $J$ satisfies $body(r)$ but $head(r)$ is not in $J$. In this case, there must be a rule $r'$ in $s\Pi_L^I$, which is $r$ with all negative literals and all nonmonotonic dl-atoms in $body(r)$ removed. Since $J$ satisfies $body(r)$, $J$ satisfies $body(r')$ and thus $head(r')$ is in $J$. Since $head(r) = head(r')$, $head(r)$ is

in $J$, a contradiction. This shows that $I$ is a strong answer set of $\Pi$. Hence, we conclude that an FLP answer set is a strong answer set. □

**Proof of Theorem 11.** By Theorem 10 it suffices to show that when $\Pi$ contains no nonmonotonic dl-atoms, if $I$ is a strong answer set then $I$ is a well-justified FLP answer set.

Let $I$ be a strong answer set of $\Pi$ relative to $L$. $I$ is the least model of the reduct $s\Pi_L^I$. Since $s\Pi_L^I$ is a positive dl-program, the least model $I$ can be computed from $s\Pi_L^I$ by applying the van Emden–Kowalski one-step provability operator $T_P(S)$ via the sequence $\langle T^i_{s\Pi_L^I}(\emptyset)\rangle_{i=0}^{\infty}$, where $T^0_{s\Pi_L^I}(\emptyset)=\emptyset$ and for $i\geq 0$ $T^{i+1}_{s\Pi_L^I}(\emptyset)=T_{s\Pi_L^I}(T^i_{s\Pi_L^I}(\emptyset))$. That is, $I$ is equal to the fixpoint $lfp(T_{s\Pi_L^I}(\emptyset))$. We next show that the least model $I$ can also be computed from the FLP reduct $f\Pi_L^I$ via the sequence $\langle T^i_{f\Pi_L^I}(\emptyset,\neg I^-)\rangle_{i=0}^{\infty}$. That is, $I$ is equal to the fixpoint $lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$.

We show by induction that for all $i\geq 0$, $T^i_{s\Pi_L^I}(\emptyset)=T^i_{f\Pi_L^I}(\emptyset,\neg I^-)$. When $i=0$, $T^0_{s\Pi_L^I}(\emptyset)=T^0_{f\Pi_L^I}(\emptyset,\neg I^-)=\emptyset$. As induction hypothesis, assume that for some integer $n$, $T^n_{s\Pi_L^I}(\emptyset)=T^n_{f\Pi_L^I}(\emptyset,\neg I^-)$. Next we show $T^{n+1}_{s\Pi_L^I}(\emptyset)=T^{n+1}_{f\Pi_L^I}(\emptyset,\neg I^-)$.

Since $I$ is a model of $\Pi$, by Theorems 2 and 3 and Lemma 1, for every $i\geq 0$, $T^i_{s\Pi_L^I}(\emptyset)\subseteq I$ and $T^i_{f\Pi_L^I}(\emptyset,\neg I^-)\subseteq I$.

By definition, $T^{n+1}_{s\Pi_L^I}(\emptyset)=T_{s\Pi_L^I}(T^n_{s\Pi_L^I}(\emptyset))=\{head(r)\mid r\in s\Pi_L^I$ such that $body(r)$ is satisfied by $T^n_{s\Pi_L^I}(\emptyset)\}$, and $T^{n+1}_{f\Pi_L^I}(\emptyset,\neg I^-)=\{head(r')\mid r'\in f\Pi_L^I$ such that $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\cup\neg I^-\models body(r')\}$. Note that $s\Pi_L^I$ has a rule $r$ if and only if $f\Pi_L^I$ has a rule $r'$, where $head(r)=head(r')$ and $body(r)$ is obtained from $body(r')$ by deleting all negative literals and all nonmonotonic dl-atoms. Since $\Pi$ contains no nonmonotonic dl-atoms, for simplicity let $body(r')=body(r)\wedge\neg A\wedge\neg B$, where $A$ is a ground monotonic dl-atom and $B$ is a ground atom.

Assume $head(r')\in T^{n+1}_{f\Pi_L^I}(\emptyset)$, due to $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\cup\neg I^-\models body(r')$. Then, $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\cup\neg I^-\models body(r)$. Since $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\subseteq I$, $body(r)$ is satisfied by $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)$. By the induction hypothesis, $body(r)$ is satisfied by $T^n_{s\Pi_L^I}(\emptyset,\neg I^-)$ and thus $head(r)\in T^{n+1}_{s\Pi_L^I}(\emptyset)$. Since $head(r)=head(r')$, $head(r')\in T^{n+1}_{s\Pi_L^I}(\emptyset)$. This shows $T^{n+1}_{s\Pi_L^I}(\emptyset)\supseteq T^{n+1}_{f\Pi_L^I}(\emptyset,\neg I^-)$.

Conversely, assume $head(r)\in T^{n+1}_{s\Pi_L^I}(\emptyset)$, due to that $body(r)$ is satisfied by $T^n_{s\Pi_L^I}(\emptyset)$. By the induction hypothesis, $body(r)$ is satisfied by $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)$. Since $body(r)$ is a conjunction of ground atoms and monotonic dl-atoms and $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\subseteq I$, $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\cup\neg I^-\models body(r)$.

Since $A$ is a monotonic dl-atom and $I$ does not satisfy $A$ ($I$ satisfies $\neg A$), no $J$ with $\emptyset\subseteq J\subseteq I$ satisfies $A$. This means all $J\subseteq I$ satisfies $\neg A$. Since $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\subseteq I$, $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\cup\neg I^-\models\neg A$.

Since $B$ is a ground atom and $I$ satisfies $\neg B$, $\neg B\in\neg I^-$. This means $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\cup\neg I^-\models\neg B$.

As a result, $T^n_{f\Pi_L^I}(\emptyset,\neg I^-)\cup\neg I^-\models body(r')$, so $head(r')\in T^{n+1}_{f\Pi_L^I}(\emptyset,\neg I^-)$. Since $head(r)=head(r')$, $head(r)\in T^{n+1}_{f\Pi_L^I}(\emptyset,\neg I^-)$. This shows $T^{n+1}_{s\Pi_L^I}(\emptyset)\subseteq T^{n+1}_{f\Pi_L^I}(\emptyset,\neg I^-)$.

Therefore, $T^{n+1}_{s\Pi_L^I}(\emptyset)=T^{n+1}_{f\Pi_L^I}(\emptyset,\neg I^-)$ and we conclude the proof. □

**Proof of Theorem 13.** By Theorem 10 it suffices to show that if $I$ is a strong answer set then $I$ is a well-justified FLP answer set. Assume that $\Pi$ has $k+1$ strata $\{\Pi_0,\ldots,\Pi_k\}$ and let $I$ be a strong answer set of $\Pi$.

By Theorem 2 and Lemma 1, $lfp(T_{f\Pi^I}(\emptyset,\neg I^-))$ is a subset of $I$.

By Theorem 12, $I=I_k$, where $I_0$ be the least model of $\Pi_0$, and for each $1\leq i\leq k$, $I_i$ is the least model of $\Pi_i(I_{i-1})\cup I_{i-1}$. Let $I_{-1}=\emptyset$. We show by induction that for $-1\leq i\leq k$, $I_i\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$. When $i=-1$, $I_{-1}\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$. As induction hypothesis, assume that for any $i$ with $0\leq i\leq k$, $I_{i-1}\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$. We next prove $I_i\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$.

Note that $I_i$ is the least model of $\Pi_i(I_{i-1})\cup I_{i-1}$ and by the induction hypothesis, $I_{i-1}\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$. Let $\Gamma=\Pi_i(I_{i-1})\cup I_{i-1}$. Since $\Gamma$ is a positive dl-program, the least model $I_i$ can be computed from $\Gamma$ by applying the van Emden–Kowalski one-step provability operator $T_\Gamma(S)$ via the sequence $\langle T^j_\Gamma(\emptyset)\rangle_{j=0}^{\infty}$, where $T^0_\Gamma(\emptyset)=\emptyset$ and for $j\geq 0$ $T^{j+1}_\Gamma(\emptyset)=T_\Gamma(T^j_\Gamma(\emptyset))$. That is, $I_i$ is equal to the fixpoint $lfp(T_\Gamma(\emptyset))$. Therefore, to prove $I_i\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$, we prove by induction that for each $j\geq 0$, $T^j_\Gamma(\emptyset)\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$. It is obviously true for $j=0$. As induction hypothesis, assume that for $0\leq j<s$, $T^j_\Gamma(\emptyset)\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$. We next show $T^s_\Gamma(\emptyset)\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$.

$T^s_\Gamma(\emptyset)=T_\Gamma(T^{s-1}_\Gamma(\emptyset))=\{head(r)\mid r\in\Gamma$ such that $body(r)$ is satisfied by $T^{s-1}_\Gamma(\emptyset)\}$. Note that $\Gamma=\Pi_i(I_{i-1})\cup I_{i-1}$, $I_{i-1}\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$ and $T^{s-1}_\Gamma(\emptyset)\subseteq lfp(T_{f\Pi_L^I}(\emptyset,\neg I^-))$. Let $r$ be a rule in $\Pi_i(I_{i-1})$ such that $body(r)$ is satisfied by $T^{s-1}_\Gamma(\emptyset)$. $\Pi_i$ must have a rule $r'$, where $head(r)=head(r')$ and $body(r)$ is obtained from $body(r')$ by deleting all negative literals and all nonmonotonic dl-atoms. For simplicity let $body(r')=body(r)\wedge A\wedge\neg B$, where $A$ is a ground nonmonotonic dl-atom and $B$ is either a ground atom or a ground dl-atom. By the definition of $\Pi_i(I_{i-1})$, $A$ is satisfied by $I_{i-1}$ and $B$ is not satisfied by $I_{i-1}$. Since $\Pi$ is stratified, the satisfaction of $A$ and $B$ only depends on the satisfaction of their input atoms in $\bigcup_{0\leq j<i}\mathcal{HB}_{\Pi_j}$,

independently of any atoms in $\mathcal{HB}_\Pi \setminus \bigcup_{0\le j<i} \mathcal{HB}_{\Pi_j}$. This means that $A$ is entailed by $I_{i-1} \cup \neg I^-_{i-1}$ and $B$ is not entailed by $I_{i-1} \cup \neg I^-_{i-1}$; i.e. $I_{i-1} \cup \neg I^-_{i-1} \models A$ and $I_{i-1} \cup \neg I^-_{i-1} \not\models B$. Then, for any interpretation $J$ with $I_{i-1} \subseteq J$ and $I^-_{i-1} \subseteq J^-$, $A$ (resp. $B$) is satisfied (resp. not satisfied) by $J$. Since $I_{i-1} \subseteq I$ and $I^-_{i-1} \subseteq I^-$, $A$ (resp. $B$) is satisfied (resp. not satisfied) by $I$. Since $body(r)$ contains no negative literals or nonmonotonic dl-atoms and $T_\Gamma^{s-1}(\emptyset) \subseteq I_i \subseteq I$, that $body(r)$ is satisfied by $T_\Gamma^{s-1}(\emptyset)$ implies $body(r)$ is satisfied by $I$. This shows that $body(r')$ is satisfied by $I$. Thus, the rule $r'$ is in $f\Pi^I_L$.

For the above rule $r'$ with $body(r') = body(r) \wedge A \wedge \neg B$, we next prove $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) \cup \neg I^- \models body(r')$. Since $body(r)$ contains no negative literals or nonmonotonic dl-atoms, that $body(r)$ is satisfied by $T_\Gamma^{s-1}(\emptyset)$ implies $body(r)$ is satisfied by all interpretations $J$ with $T_\Gamma^{s-1}(\emptyset) \subseteq J$. Since $T_\Gamma^{s-1}(\emptyset) \subseteq lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \subseteq I$, $body(r)$ is satisfied by all interpretations that satisfy $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) \cup \neg I^-$. This means $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) \cup \neg I^- \models body(r)$. Moreover, as shown above, for any interpretation $J$ with $I_{i-1} \subseteq J$ and $I^-_{i-1} \subseteq J^-$, $A$ (resp. $B$) is satisfied (resp. not satisfied) by $J$. Since $I_{i-1} \subseteq lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-))$ and $I^-_{i-1} \subseteq I^-$, for any interpretation $J$ that satisfies $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) \cup \neg I^-$, $A$ (resp. $B$) is satisfied (resp. not satisfied) by $J$. This means $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) \cup \neg I^- \models A$ and $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) \cup \neg I^- \models \neg B$. As a result, $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) \cup \neg I^- \models body(r')$. By the definition of the fixpoint $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-))$, $head(r') \in lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-))$ and thus $head(r) \in lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-))$. This shows $T^s_\Gamma(\emptyset) \subseteq lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-))$. Therefore, $lfp(T_\Gamma(\emptyset)) \subseteq lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-))$ and thus $I_i \subseteq lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-))$.

To conclude, $I = I_k \subseteq lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) \subseteq I$; i.e. $lfp(T_{f\Pi^I_L}(\emptyset, \neg I^-)) = I$. Hence, $I$ is a well-justified FLP answer set. □

**Proof of Theorem 14.**

1. Membership

We first prove the $\Sigma^p_2$-membership of deciding the existence of FLP answer sets. Given a propositional logic program $\Pi$, we guess an interpretation $I$ and first verify that $I$ is a model of $\Pi$, and then compute the FLP-reduct $f\Pi^I$. These two steps can be done in polynomial time. The main part of determining if $I$ is an FLP answer set of $\Pi$ is to determine if $I$ is a minimal model of $f\Pi^I$. Note that $I$ is a minimal model of $f\Pi^I$ if and only if $f\Pi^I \cup \neg I^- \cup \neg \bigwedge_{A\in I} A$ is unsatisfiable. This is co-NP-complete because it is NP-complete to determine if a propositional theory is satisfiable. Thus with a call to an NP oracle, we can verify whether $I$ is an FLP answer set of $\Pi$ in polynomial time.

Next we prove the $\Sigma^p_2$-membership of deciding the existence of well-justified FLP answer sets. Given a propositional logic program $\Pi$, we guess an interpretation $I$ and can verify that $I$ is a model of $\Pi$ and compute the FLP-reduct $f\Pi^I$ in polynomial time. It consists of two major parts to determine if $I$ is a well-justified FLP answer set of $\Pi$: (1) compute the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$; and (2) determine whether $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models \bigwedge_{A\in I} A$. Let $\Pi$ consist of $M$ rules. To reach the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$, we have computations of the form $T^i_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models body(r)$ for at most $M^2$ times. Note that it is co-NP-complete to compute $T^i_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models body(r)$. Thus part (1) can be done in polynomial time with the help of an NP oracle. Part (2) can be computed with one call to an NP oracle. Consequently, we can verify whether $I$ is a well-justified FLP answer set of $\Pi$ in polynomial time with the help of an NP oracle.

2. Hardness

Next we prove the $\Sigma^p_2$-hardness of deciding the existence of ordinary FLP resp. well-justified FLP answer sets. A positive disjunctive logic program $P$ consists of a finite set of rules of the form $A_1 \vee \cdots \vee A_l \leftarrow B_1 \wedge \cdots \wedge B_m$, where $l > 0$, $m \ge 0$, and each $A_i$ and $B_i$ is a ground atom. An interpretation $I$ is a standard answer set of $P$ if and only if $I$ is a minimal model of $P$ [34]. As shown by Eiter and Gottlob [23], for a given ground atom $A$ it is $\Sigma^p_2$-hard to decide whether a given $P$ has a standard answer set (i.e. a minimal model) in which $A$ is true.

Let $\Pi = P' \cup \{A \leftarrow \neg A\}$ be a propositional logic program, where $P'$ is $P$ with each rule $H \leftarrow B$ replaced by a material implication $B \supset H$. Note that an interpretation $I$ is a minimal model of $P$ if and only if $I$ is a minimal model of $P'$. Since all rule bodies in $P'$ are empty, by Theorem 7, $I$ is a minimal model (standard answer set) of $P$ if and only if $I$ is a well-justified FLP answer set of $P'$ if and only if $I$ is an FLP answer set of $P'$.

Assume that $I$ is a minimal model (standard answer set) of $P$ in which $A$ is true. Then, the FLP-reduct $f\Pi^I$ of $\Pi$ w.r.t. $I$ is the same as $P'$. By Theorem 7, $I$ is a well-justified FLP answer set of $f\Pi^I$, i.e., for each $E \in I$, $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models E$. This means $I$ is also a well-justified FLP answer set of $\Pi$, and by Corollary 2 also an FLP answer set of $\Pi$.

Conversely, assume that $I$ is an FLP or a well-justified FLP answer set of $\Pi$. Due to the rule $A \leftarrow \neg A$ in $\Pi$, $A$ must be in $I$ and thus $f\Pi^I = P'$. By Theorem 4, $I$ is a minimal model of $f\Pi^I$. Thus $I$ is a minimal model (standard answer set) of $P$ in which $A$ is true.

The above proof shows that deciding whether a positive disjunctive logic program $P$ has a standard answer set in which a given ground atom $A$ is true can be reduced to deciding the existence of ordinary FLP resp. well-justified FLP answer sets of a propositional logic program $\Pi$. Since $\Pi$ can be constructed from $P$ in polynomial time, the $\Sigma^p_2$-hardness of deciding the existence of ordinary FLP resp. well-justified FLP answer sets of a propositional logic program immediately follows. We thus conclude the proof of Theorem 14. □

To prove Theorem 15, we introduce the following two lemmas, which show that cautious (resp. brave) reasoning for propositional logic programs under the FLP or the well-justified FLP answer set semantics can be reduced to deciding the non-existence (resp. existence) of FLP or well-justified FLP answer sets, and vice versa.

**Lemma 5.** *Let $\Pi$ be a propositional logic program and $l$ an atom in $\mathcal{HB}_\Pi$. Let $\Pi_1 = \Pi \cup \{p \leftarrow l \wedge \neg p\}$ (resp. $\Pi_1 = \Pi \cup \{p \leftarrow \neg l \wedge \neg p\}$), where $p$ is a ground atom of a 0-ary predicate not occurring in $\Pi$ (i.e., $\mathcal{HB}_{\Pi_1} = \mathcal{HB}_\Pi \cup \{p\} \neq \mathcal{HB}_\Pi$). Then, $l$ belongs to every (resp. some) FLP or well-justified FLP answer set of $\Pi$ if and only if $\Pi_1$ has no (resp. an) FLP or well-justified FLP answer set.*

**Proof.** We first prove that cautious reasoning can be reduced to deciding the non-existence of FLP or well-justified FLP answer sets.

($\Longrightarrow$) Assume that $l$ belongs to every FLP or well-justified FLP answer set of $\Pi$, and towards a contradiction that $\Pi_1$ has an FLP or a well-justified FLP answer set $I$. Observe that $f\Pi^I = f\Pi_1^I$: otherwise the rule $p \leftarrow l \wedge \neg p$ would be in $f\Pi_1^I$ which implies $p \notin I$ and $l \in I$; however then $I$ would not be a model of this rule and thus also not of $f\Pi_1^I$, contradicting our assumption that $I$ is an FLP or a well-justified FLP answer set of $\Pi_1$. Given that $f\Pi^I = f\Pi_1^I$ however, we conclude that $l \notin I$ (either $p \in I$ or $l \notin I$ has to hold, but $p \in I$ is not founded since there is no rule with head $p$ in $f\Pi^I = f\Pi_1^I$), and that $I$ is also an FLP or well-justified FLP answer set of $\Pi$. Since $l \notin I$, we reach a contradiction that $l$ belongs to every FLP or well-justified FLP answer set of $\Pi$.

($\Longleftarrow$) Assume that $\Pi_1$ has no FLP or well-justified FLP answer set, and towards a contradiction that $\Pi$ has an FLP or well-justified FLP answer set $I$ with $l \notin I$. Then $I$ is a model of $\Pi_1$ and $f\Pi^I = f\Pi_1^I$. So $I$ is also an FLP or well-justified FLP answer set of $\Pi_1$, which contradicts that $\Pi_1$ has no FLP or well-justified FLP answer set.

We next prove that brave reasoning can be reduced to deciding the existence of FLP or well-justified FLP answer sets.

($\Longrightarrow$) Let $I$ be an FLP or well-justified FLP answer set of $\Pi$ with $l \in I$. Then $I$ is a model of $\Pi_1$ and $f\Pi^I = f\Pi_1^I$. So $I$ is also an FLP or well-justified FLP answer set of $\Pi_1$.

($\Longleftarrow$) Let $\Pi_1$ have an FLP or well-justified FLP answer set $I$. Then $p \notin I$ and $l \in I$ (by the rule $p \leftarrow \neg l \wedge \neg p$). Since $f\Pi^I = f\Pi_1^I$, $I$ is also an FLP or well-justified FLP answer set of $\Pi$ with $l \in I$. □

**Lemma 6.** *Let $\Pi$ be a propositional logic program, $\Pi_1 = \Pi \cup \{p \leftarrow p\}$ and $\Pi_2 = \Pi \cup \{p\}$, where $p$ is a ground atom of a 0-ary predicate not occurring in $\Pi$ (i.e., $\mathcal{HB}_{\Pi_1} = \mathcal{HB}_{\Pi_2} = \mathcal{HB}_\Pi \cup \{p\} \neq \mathcal{HB}_\Pi$). Then, $\Pi$ has no (resp. an) FLP or well-justified FLP answer set if and only if $p$ belongs to every (resp. some) FLP or well-justified FLP answer set of $\Pi_1$ (resp. $\Pi_2$).*

**Proof.** First note that if a logic program $\Pi$ has no answer set, which means $\Pi$ is inconsistent, then everything is cautiously true in $\Pi$ under the answer set semantics, i.e., any $l \in \mathcal{HB}_\Pi$ trivially belongs to every answer set of $\Pi$.

Since $p$ is a fresh atom not occurring in $\Pi$, $\Pi$ and $\Pi_1$ by construction have the same FLP and well-justified FLP answer sets, none of which contains $p$ (which would not be founded). Therefore, if $\Pi$ has no FLP or well-justified FLP answer set, then $\Pi_1$ has no FLP or well-justified FLP answer set. Thus, $p$ trivially belongs to every FLP or well-justified FLP answer set of $\Pi_1$. Conversely, if $p$ belongs to every FLP or well-justified FLP answer set of $\Pi_1$, then $\Pi_1$ must be inconsistent without FLP or well-justified FLP answer sets. Hence, $\Pi$ must have no FLP or well-justified FLP answer set.

Since $p$ is a fresh atom not occurring in $\Pi$, by construction $\Pi$ has an FLP or well-justified FLP answer set $I$ if and only if $\Pi_2$ has an FLP or well-justified FLP answer set $I \cup \{p\}$. If $\Pi$ has an FLP or well-justified FLP answer set $I$, then $p$ belongs to the FLP or well-justified FLP answer set $I \cup \{p\}$ of $\Pi_2$. Conversely, if $p$ belongs to an FLP or well-justified FLP answer set $I$ of $\Pi_2$, then $\Pi$ has an FLP or well-justified FLP answer set $I \setminus \{p\}$. □

**Proof of Theorem 15.** Lemma 5 says that for propositional logic programs, the problem of deciding whether a ground atom belongs to every (resp. some) FLP or well-justified FLP answer set can be reduced to deciding whether there exists no (resp. an) FLP or well-justified FLP answer set, while Lemma 6 states the converse reduction. This shows that cautious (resp. brave) reasoning for propositional logic programs falls in the same complexity class as the non-existence (resp. existence) of FLP or well-justified FLP answer sets. Since it is $\Sigma_2^p$-complete to determine if a propositional logic program has an FLP answer set or a well-justified FLP answer set (Theorem 14), deciding whether a ground atom is in every (resp. some) FLP or well-justified FLP answer set is complete for co-$\Sigma_2^p$ (resp. $\Sigma_2^p$). □

**Proof of Theorem 16.** The proof is similar to the proof of Theorems 14 and 15, so we only give a sketch for the $\Sigma_2^p$-completeness of answer set existence. For the membership proof, since deciding whether a propositional formula is satisfiable is in NP, deciding whether a propositional formula with polynomially computable aggregates is satisfiable is also in NP. Then, by the same argument as the proof of Theorem 14, we can conclude that deciding whether a propositional logic program with polynomially computable aggregates has an FLP answer set or a well-justified FLP answer set is in $\Sigma_2^p$. For the hardness proof, since propositional logic programs without aggregates are a special case of propositional logic programs with aggregates, by Theorem 14 it is $\Sigma_2^p$-hard to decide whether a propositional logic program with aggregates has an FLP answer set or a well-justified FLP answer set. □

**Proof of Theorem 17.** For the membership proof, since ground normal logic programs with aggregates are special propositional logic programs with aggregates, by Theorem 16 deciding whether a ground normal program with polynomially computable aggregates has a well-justified FLP answer set is in $\Sigma_2^p$.

Since a ground Horn logic program is a special ground normal logic program, to establish the theorem it remains to prove the $\Sigma_2^p$-hardness of deciding the existence of some well-justified FLP answer set for a given ground Horn logic program with polynomially computable aggregates. We achieve this by a reduction of deciding the validity of a quantified Boolean formula

$$\Phi = \exists x_1 \cdots \exists x_n \, \forall y_1 \cdots \forall y_m E \quad n, m \geq 1$$

where $E$ is a propositional formula made of ground atoms $x_1, \ldots, x_n, y_1, \ldots, y_m$. For each truth assignment $\upsilon$ to $x_1, \ldots, x_n$, $\Phi$ is true if $(\bigwedge_{\upsilon(x_i)=true} x_i) \wedge (\bigwedge_{\upsilon(x_i)=false} \neg x_i) \models E$. Also, $\Phi$ is valid if there is *such* an assignment $\upsilon$ such that $\Phi$ is true. It has been shown that deciding the validity of a quantified Boolean formula of the above type is $\Sigma_2^p$-hard [61].

Note that for any ground formula $F$, we can construct an aggregate atom $\mathrm{SUM}\langle(\{1\}, X) : F\rangle = 1$ which is logically equivalent to $F$; i.e., since the aggregate variable $X$ does not appear in $F$, any interpretation $I$ satisfies $\mathrm{SUM}\langle(\{1\}, X) : F\rangle = 1$ if and only if $I$ satisfies $F$. Therefore, we can use $F$ and $\mathrm{SUM}\langle(\{1\}, X) : F\rangle = 1$ exchangeably.

Let $x'_1, \ldots, x'_n, f, f'$ be new ground atoms with a zero-arity predicate. We define a ground Horn logic program $\Pi$ with aggregate, which consists of the following rules:

$$x'_i \leftarrow \mathrm{SUM}\langle(\{1\}, X) : \neg x_i\rangle = 1 \qquad \text{for each } 1 \leq i \leq n \tag{1}$$

$$x_i \leftarrow \mathrm{SUM}\langle(\{1\}, X) : \neg x'_i\rangle = 1 \qquad \text{for each } 1 \leq i \leq n \tag{2}$$

$$f \leftarrow x_i \wedge x'_i \wedge \mathrm{SUM}\langle(\{1\}, X) : \neg f\rangle = 1 \qquad \text{for each } 1 \leq i \leq n \tag{3}$$

$$f' \leftarrow \mathrm{SUM}\langle(\{1\}, X) : \neg y_j\rangle = 1 \wedge \mathrm{SUM}\langle(\{1\}, X) : \neg f'\rangle = 1 \qquad \text{for each } 1 \leq j \leq m \tag{4}$$

$$y_j \leftarrow \mathrm{SUM}\langle(\{1\}, X) : E\rangle = 1 \qquad \text{for each } 1 \leq j \leq m \tag{5}$$

Intuitively, for each $1 \leq i \leq n$, $x'_i$ corresponds to $\neg x_i$, which is denoted by the logically equivalent aggregate atom $\mathrm{SUM}\langle(\{1\}, X) : \neg x_i\rangle = 1$. Since all of the ground atoms $x_i$, $x'_i$, $y_j$, $f$, $f'$ are with a zero-arity predicate, the Herbrand base of $\Pi$ is $\mathcal{HB}_\Pi = \{x_1, \ldots, x_n, y_1, \ldots, y_m, x'_1, \ldots, x'_n, f, f'\}$.

Note that all aggregate atoms in $\Pi$ are computable in polynomial time.

Let $I$ be a well-justified FLP answer set of $\Pi$. Observe that neither $f$ nor $f'$, which only occur in rules (3) and (4), can be founded. Therefore, $f \notin I$ and $f' \notin I$. Moreover, rules (3) and (4) intuitively act as constraints, whose body must not be satisfied for $I$ to be well-justified. Therefore, $y_j$ is in $I$ (by rules (4)) for every $1 \leq j \leq m$, and for every $i = 1, \ldots, n$, atoms $x_i$ and $x'_i$ cannot be jointly in $I$ (by rules (3)). Additionally, either $x_i \in I$ or $x'_i \in I$ holds (by rules (1) and (2), because if $I \cap \{x_i, x'_i\} = \emptyset$ for some index $i$, then the body of the respective rules (1) and (2) is satisfied but not their head). That is, $x_i \in I$ ($x'_i \in I$) if and only if $x'_i \notin I$ (resp. $x_i \notin I$).

Next we show that $\Pi$ has a well-justified FLP answer set if and only if the quantified Boolean formula $\Phi$ is valid.

($\Longrightarrow$) Let $I$ be a well-justified FLP answer set of $\Pi$. We define the truth assignment $\upsilon$ to the atoms $x_1, \ldots, x_n$ as follows:

$$\upsilon(x_i) = \begin{cases} true & \text{if } x_i \in I \\ false & \text{if } x'_i \in I \end{cases}$$

By $I\,|_\upsilon$ we denote $\{x_i \mid x_i \in I\}$ and by $\neg I^-\,|_\upsilon$ we denote $\{\neg x_i \mid x_i \notin I\}$. Then under the assignment $\upsilon$, $\Phi$ is true if $I\,|_\upsilon \cup \neg I^-\,|_\upsilon \models E$.

Since $\{y_1, \ldots, y_m\} \subseteq I$ and rules (5) are the only rules in $\Pi$ whose heads contain $y_1, \ldots, y_m$, by the level mapping of the well-justified FLP answer set semantics the FLP reduct $f\Pi^I$ must contain rules (5) and the entailment $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models \mathrm{SUM}\langle(\{1\}, X) : E\rangle = 1$ must hold; for otherwise $y_1, \ldots, y_m$ would have no justification. Then $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models E$. Since every rule head in $\Pi$ is a ground atom, the least fixpoint itself is an interpretation, in particular $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = I$ and thus $I \cup \neg I^- \models E$. Note that $y_1, \ldots, y_m$ can be justified only by the rules (5). Due to $\mathrm{SUM}\langle(\{1\}, X) : E\rangle = 1$ in their bodies, this only is the case if $I \setminus \{y_1, \ldots, y_m\} \cup \neg I^- \models E$. Moreover, since the atoms $x'_1, \ldots, x'_n, f, f'$ do not occur in $E$, $I \setminus \{y_1, \ldots, y_m, x'_1, \ldots, x'_n\} \cup \neg I^- \setminus \{\neg x'_1, \ldots, \neg x'_n, \neg f, \neg f'\} \models E$. This means $I\,|_\upsilon \cup \neg I^-\,|_\upsilon \models E$ and thus $\Phi$ is true under the above truth assignment $\upsilon$. Hence $\Phi$ is valid.

($\Longleftarrow$) Assume that $\Phi$ is valid, i.e. there exists a truth assignment $\upsilon$ to the atoms $x_1, \ldots, x_n$ such that $\Phi$ is true. Let $I$ be the following interpretation:

$$I = \left\{x_i \;\middle|\; \upsilon(x_i) = true, 1 \leq i \leq n\right\} \cup \left\{x'_i \;\middle|\; \upsilon(x_i) = false, 1 \leq i \leq n\right\} \cup \{y_1, \ldots, y_m\}.$$

Obviously, $I$ is a model of $\Pi$. Since all rule heads in $\Pi$ are ground atoms, to show that $I$ is a well-justified FLP answer set, it suffices to show $lfp(T_{f\Pi^I}(\emptyset, negI^-)) = I$.

Clearly, no rules of (3) and (4) are in the FLP reduct $f\Pi^I$. Since $\Phi$ is true under the assignment $\upsilon$, $I|_\upsilon \cup \neg I^-|_\upsilon \models E$ and thus $I$ satisfies $E$. Then, $I$ satisfies $\mathrm{SUM}\langle(\{1\}, X) : E\rangle = 1$. So all rules of (5) are in $f\Pi^I$. For $1 \le i \le n$, $x_i \in I$ if and only if $x_i \leftarrow \mathrm{SUM}\langle(\{1\}, X) : \neg x_i'\rangle = 1$ is in $f\Pi^I$, and $x_i' \in I$ if and only if $x_i' \leftarrow \mathrm{SUM}\langle(\{1\}, X) : \neg x_i\rangle = 1$ is in $f\Pi^I$. As a result, $f\Pi^I$ consists of the following rules:

$$x_i' \leftarrow \mathrm{SUM}\big\langle\big(\{1\}, X\big) : \neg x_i\big\rangle = 1 \qquad \text{if } x_i \in I^-, \text{ for each } 1 \le i \le n \tag{1'}$$

$$x_i \leftarrow \mathrm{SUM}\big\langle\big(\{1\}, X\big) : \neg x_i'\big\rangle = 1 \qquad \text{if } x_i' \in I^-, \text{ for each } 1 \le i \le n \tag{2'}$$

$$y_j \leftarrow \mathrm{SUM}\big\langle\big(\{1\}, X\big) : E\big\rangle = 1 \qquad \text{for each } 1 \le j \le m \tag{5'}$$

Note that the heads of all rules of (1′) and (2′) constitute $I \setminus \{y_1, \ldots, y_m\}$.

Next we build the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$. To start, let $T^0_{f\Pi^I}(\emptyset, \neg I^-) = \emptyset$. Since the bodies of all rules of (1′) and (2′) are true in $T^0_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^-$, $T^1_{f\Pi^I}(\emptyset, \neg I^-) = I \setminus \{y_1, \ldots, y_m\}$. Since $I|_\upsilon \cup \neg I^-|_\upsilon \models E$, $(I \setminus \{y_1, \ldots, y_m\})|_\upsilon \cup \neg I^-|_\upsilon \models E$ and thus $T^1_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models E$. Then, $T^1_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models \mathrm{SUM}\langle(\{1\}, X) : E\rangle = 1$. So $T^2_{f\Pi^I}(\emptyset, \neg I^-) = T^1_{f\Pi^I}(\emptyset, \neg I^-) \cup \{y_1, \ldots, y_m\} = I$. The fixpoint is $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = T^2_{f\Pi^I}(\emptyset, \neg I^-)$, so $I$ is a well-justified FLP answer set of $\Pi$.

The above proof shows that deciding the validity of a quantified Boolean formula $\Phi$ can be reduced to deciding the existence of well-justified FLP answer sets of a ground Horn logic program $\Pi$ with polynomially computable aggregates. Since $\Pi$ can be constructed from $\Phi$ in polynomial time, the $\Sigma_2^p$-hardness of deciding the existence of well-justified FLP answer sets of a ground Horn logic program with aggregates immediately follows from the $\Sigma_2^p$-hardness of deciding the validity of a quantified Boolean formula. This concludes the proof of Theorem 17. □

For the proof of Theorem 18, we first recall the following lemma from [24, Lemma E.5].

**Lemma 7.** *Let $\Pi$ be a dl-program relative to a DL knowledge base $L$. The number of ground dl-atoms in ground($\Pi$) is polynomial, and every such ground dl-atom $A = DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{c})$ has in general exponentially many different concrete inputs $I_p$ (that is, interpretations $I_p$ of its input predicate symbols $p = p_1, \ldots, p_m$), but each of these concrete inputs $I_p$ has a polynomial size. Furthermore, if $\Pi$ is positive, then during the computation of the least model of $\Pi$ by fixpoint iteration, the input of any ground dl-atom $A$ in ground($\Pi$) can increase only polynomially many times, and it thus needs to be evaluated polynomially often.*

For clarity, we prove Theorem 18 by dividing it into three independent cases and proving them separately. Case 1: $L$ belongs to $\mathcal{SHIF}(\mathbf{D})$; case 2: $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$; and case 3: $L$ belongs to $\mathcal{SROIQ}(\mathbf{D})$. We first prove two lemmas. In the sequel, we denote $\mathcal{HB}^*_\Pi = \mathcal{HB}_\Pi \cup \{\neg A \mid A \in \mathcal{HB}_\Pi\}$. A subset of $\mathcal{HB}^*_\Pi$ is said to be consistent if it contains no contradictory literals $A$ and $\neg A$.

**Lemma 8.** *Let $\Pi$ be a dl-program relative to a DL knowledge base $L$, $A = DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{c})$ a ground dl-atom in ground($\Pi$), and $S$ a consistent subset of $\mathcal{HB}^*_\Pi$. Computing $S \models A$ or $S \models \neg A$ is in* EXP *when $L$ belongs to $\mathcal{SHIF}(\mathbf{D})$.*

**Proof.** By definition, an interpretation $I$ of $\Pi$ satisfies the dl-atom $A$ if $L \cup \bigcup_{i=1}^m A_i(I) \models Q(\mathbf{c})$, where each $A_i(I)$ is a set of concept membership axioms, role membership axioms, equality/inequality axioms, or their negations, which are obtained from the input predicate $p_i$ in terms of $I$. By Lemma 7, any interpretation of the input predicates of $A$ has a polynomial size, so $\bigcup_{i=1}^m A_i(I)$ has a polynomial size and the computation of $\bigcup_{i=1}^m A_i(I)$ is feasible in exponential time. The computation of $L \cup \bigcup_{i=1}^m A_i(I) \models Q(\mathbf{c})$ can be reduced to computing the unsatisfiability of the DL knowledge base $L \cup \bigcup_{i=1}^m A_i(I) \cup \{\neg Q(\mathbf{c})\}$. Since deciding whether a DL knowledge base in $\mathcal{SHIF}(\mathbf{D})$ is satisfiable is complete for EXP [62,39], the computation of $L \cup \bigcup_{i=1}^m A_i(I) \models Q(\mathbf{c})$ is feasible in exponential time. As a result, deciding whether an interpretation $I$ satisfies a dl-atom $A$ (resp. $\neg A$) is in EXP.

Computing $S \models A$ (resp. $S \models \neg A$) is to check that every model $I$ of $S$ satisfies $A$ (resp. $\neg A$). Checking if an interpretation $I$ is a model of $S$ is in EXP. By Lemma 7, $A$ has exponentially many different concrete inputs/interpretations. So $S \models A$ or $S \models \neg A$ can be computed by calling a $\mathcal{SHIF}(\mathbf{D})$ reasoner at most exponential times. In each call one interpretation is checked to see if it satisfies $A$ (resp. $\neg A$). Consequently, computing $S \models A$ or $S \models \neg A$ is in EXP when $L$ belongs to $\mathcal{SHIF}(\mathbf{D})$. □

**Lemma 9.** *Let $\Pi$ be a dl-program relative to a DL knowledge base $L$, $A = DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{c})$ a ground dl-atom in ground($\Pi$), and $S$ a consistent subset of $\mathcal{HB}^*_\Pi$. Computing $S \models A$ (resp. $S \models \neg A$) is in co-*NEXP *(resp.* NEXP*) when $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$, and in co-*N2EXP *(resp.* N2EXP*) when $L$ belongs to $\mathcal{SROIQ}(\mathbf{D})$.*

**Proof.** As shown for Lemma 8, checking whether an interpretation $I$ of $\Pi$ satisfies a dl-atom $A = DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{c})$ amounts to checking the unsatisfiability of the DL knowledge base $L \cup \bigcup_{i=1}^m A_i(I) \cup \{\neg Q(\mathbf{c})\}$. Recall that deciding whether a DL knowledge base in $\mathcal{SHOIN}(\mathbf{D})$ (resp. $\mathcal{SROIQ}(\mathbf{D})$) is satisfiable is complete for NEXP (resp. N2EXP) [62,41].

To compute $S \models A$ is to check that every model $I$ of $S$ satisfies the dl-atom $A$. Its complementary task, i.e., to compute $S \not\models A$, is to check that there exists a model $I$ of $S$ that does not satisfy $A$. The latter can be done by guessing an interpretation $I$ for $\Pi$ together with an interpretation $J$ for the DL knowledge base $L \cup \bigcup_{i=1}^{m} A_i(I) \cup \{\neg Q(\mathbf{c})\}$ and then verifying that (1) $I$ is a model of $S$, and (2) $J$ satisfies $L \cup \bigcup_{i=1}^{m} A_i(I) \cup \{\neg Q(\mathbf{c})\}$.[10] The guess of $I$ and $J$ can be done in exponential (resp. double exponential) time when $L$ is in $\mathcal{SHOIN}(\mathbf{D})$ (resp. $\mathcal{SROIQ}(\mathbf{D})$). Verifying that $I$ is a model of $S$ can be done in exponential time. Since deciding whether a DL knowledge base in $\mathcal{SHOIN}(\mathbf{D})$ (resp. $\mathcal{SROIQ}(\mathbf{D})$) is satisfiable is complete for NEXP (resp. N2EXP), verifying that $J$ satisfies $L \cup \bigcup_{i=1}^{m} A_i(I) \cup \{\neg Q(\mathbf{c})\}$ can be done in exponential (resp. double exponential) time. This shows that the computation of $S \not\models A$ can be done in NEXP (resp. N2EXP) for $\mathcal{SHOIN}(\mathbf{D})$ (resp. $\mathcal{SROIQ}(\mathbf{D})$). As a result, computing $S \models A$ is in co-NEXP when $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$, and in co-N2EXP when $L$ belongs to $\mathcal{SROIQ}(\mathbf{D})$.

To compute $S \models \neg A$ is to check that every model $I$ of $S$ does not satisfy the dl-atom $A$. That is, for every model $I$ of $S$, $L \cup \bigcup_{i=1}^{m} A_i(I) \not\models Q(\mathbf{c})$ or equivalently, $L \cup \bigcup_{i=1}^{m} A_i(I) \cup \{\neg Q(\mathbf{c})\}$ is satisfiable. Checking this satisfiability is in NEXP (resp. N2EXP) for $\mathcal{SHOIN}(\mathbf{D})$ (resp. $\mathcal{SROIQ}(\mathbf{D})$). There may be at most exponentially many such models for $S$ (Lemma 7). Therefore, computing $S \models \neg A$ is in NEXP when $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$, and in N2EXP when $L$ belongs to $\mathcal{SROIQ}(\mathbf{D})$. □

**Proof of Theorem 18.** Case 1: deciding whether $\Pi$ has a well-justified FLP answer set is NEXP-complete when $L$ belongs to $\mathcal{SHIF}(\mathbf{D})$.

We first guess an interpretation $I$ and show that we can verify in EXP that $I$ is a well-justified FLP answer set of $\Pi$. By Lemma 8, for each dl-atom $A$ appearing in a rule body $body(r)$ of $ground(\Pi)$, computing $I \cup \neg I^- \models A$ and $I \cup \neg I^- \models \neg A$ is in EXP and thus checking if $I$ satisfies $body(r)$ is in EXP. $ground(\Pi)$ may have exponentially many rules, so checking whether $I$ is a model of $\Pi$ and computing the FLP reduct $f\Pi^I$ is in EXP.

To verify that $I$ is a well-justified FLP answer set of $\Pi$, we (1) build the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$, and (2) check $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) \cup \neg I^- \models \bigwedge_{A \in I} A$. Let $ground(\Pi)$ consist of $M$ rules. To reach the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$, we have computations of the form $T^i_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models body(r)$ for at most $M^2$ times. By Lemma 8, it is EXP to compute $T^i_{f\Pi^I}(\emptyset, \neg I^-) \cup \neg I^- \models body(r)$. Thus part (1) is feasible in exponential time. Part (2) can also be done in exponential time. Consequently, we can verify whether $I$ is a well-justified FLP answer set of $\Pi$ in exponential time. Therefore, deciding whether $\Pi$ has a well-justified FLP answer set can be done in NEXP.

Recall that for a normal logic program, the well-justified FLP answer set semantics coincides with the standard answer set semantics. Since deciding whether a non-ground normal logic program has an answer set under the standard answer set semantics is NEXP-complete [11], it is NEXP-hard to determine whether $\Pi$ has a well-justified FLP answer set.

To conclude, deciding whether $\Pi$ has a well-justified FLP answer set is NEXP-complete when $L$ belongs to $\mathcal{SHIF}(\mathbf{D})$. □

In the above proof of the hardness of answer set existence for a dl-program relative to a $\mathcal{SHIF}(\mathbf{D})$ knowledge base, we used the existing hardness result for a non-ground normal logic program under the standard answer set semantics. It seems that there is no such existing hardness result available for a dl-program relative to a $\mathcal{SHOIN}(\mathbf{D})$ or $\mathcal{SROIQ}(\mathbf{D})$ knowledge base. Observe that Lemmas 5 and 6 apply to dl-programs relative to any DL knowledge bases as well (with the same proof); i.e., cautious (resp. brave) reasoning for dl-programs under the well-justified FLP answer set semantics can be reduced to deciding the non-existence (resp. existence) of well-justified FLP answer sets, and vice versa. Thus, cautious (resp. brave) reasoning for dl-programs falls in the same complexity class as the non-existence (resp. existence) of well-justified FLP answer sets. Therefore, to prove Theorem 18 for the case of $\mathcal{SHOIN}(\mathbf{D})$, we first introduce the following hardness result.

**Lemma 10.** *Let $\Pi$ be a dl-program relative to a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base $L$ and $l$ a ground atom in $\mathcal{HB}_\Pi$. Deciding whether $l$ is in some well-justified FLP answer set of $\Pi$ is* $\mathrm{P}^{\mathrm{NEXP}}$*-hard.*

**Proof.** Let $M$ be a polynomial-time bounded deterministic Turing machine with access to a NEXP oracle. The $\mathrm{P}^{\mathrm{NEXP}}$-hardness is proved by a reduction of $M$ to brave reasoning for a stratified dl-program $P$ relative to a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base, where dl-atoms in $P$ are used to decide NEXP oracle calls made by $M$. The reduction is just the same as the one presented in [24, Theorem 7.2] except that we do not need to introduce the rule $\neg b^l_{2l-2}(0) \leftarrow$ to the stratified dl-program $P$, where $\neg b^l_{2l-2}(0)$ is a "classical negated atom". For simplicity, we do not reproduce the reduction here. As a result, $M$ accepts an input $v$ if and only if a ground atom $l$ belongs to the unique strong answer set of $P$. By Theorem 13, $M$ accepts an input $v$ if and only if a ground atom $l$ belongs to the unique well-justified FLP answer set of $P$. Therefore, deciding whether $l$ is in some well-justified FLP answer set of $\Pi$ is $\mathrm{P}^{\mathrm{NEXP}}$-hard. □

[10] When $J$ satisfies $L \cup \bigcup_{i=1}^{m} A_i(I) \cup \{\neg Q(\mathbf{c})\}$, $L \cup \bigcup_{i=1}^{m} A_i(I) \cup \{\neg Q(\mathbf{c})\}$ is satisfiable. In this case, $L \cup \bigcup_{i=1}^{m} A_i(I) \not\models Q(\mathbf{c})$ and thus $I$ does not satisfy the dl-atom $A$.

**Proof of Theorem 18.** Case 2: deciding whether $\Pi$ has a well-justified FLP answer set is $\mathrm{P}^{\mathrm{NEXP}}$-complete when $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$.

By Lemma 7, the number of ground dl-atoms in $ground(\Pi)$ is polynomial. Let $\mathcal{HB}^{DL}_{\Pi} = \{A \in \mathcal{HB}_{\Pi} \mid A \text{ is an input atom of a dl-atom in } ground(\Pi)\}$. The size of $\mathcal{HB}^{DL}_{\Pi}$ is also polynomial. Let $I_p \subseteq \mathcal{HB}^{DL}_{\Pi}$ be an interpretation of input atoms of all dl-atoms in $ground(\Pi)$. We call $I_p$ an *input interpretation* of dl-atoms. Let $I_p^- = \mathcal{HB}^{DL}_{\Pi} \setminus I_p$.

We first guess an input interpretation $I_p$ together with a chain $I_p^0 = \emptyset \subset I_p^1 \subset \cdots \subset I_p^k = I_p$. Since the size of $I_p$ is polynomial, $k$ is a polynomial. For each dl-atom $A = DL[S_1 op_1 p_1, \ldots, S_m op_m p_m; Q](\mathbf{c})$ in $ground(\Pi)$ and each $i \in \{0, \ldots, k\}$, we check $I_p^i \cup \neg I_p^- \models A$ and $I_p^i \cup \neg I_p^- \models \neg A$ by calling a NEXP oracle (see Lemma 9). Hence the evaluation of all dl-atoms in $ground(\Pi)$ can be done in polynomial time with the help of a NEXP oracle.

Given the above guess and the recorded results of the above checks $I_p^i \cup \neg I_p^- \models A$ and $I_p^i \cup \neg I_p^- \models \neg A$ for all dl-atoms $A$ in $ground(\Pi)$, we next call a NEXP oracle to check if there is a Herbrand model $I$ of $\Pi$ that (1) is an extension of $I_p$, which (2) is obtained by a fixpoint computation compliant with the chain and (3) yielding $I$ as a least fixpoint. More specifically, we guess $I$ and check (1) $I_p = I \cap \mathcal{HB}^{DL}_{\Pi}$; (2) whether in the sequence $\langle T^i_{\Pi}(\emptyset, \neg I^-)\rangle_{i=0}^{\infty}$ for the computation of the fixpoint $lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$, the input interpretations increase following the chain $I_p^0 = \emptyset \subset I_p^1 \subset \cdots \subset I_p^k = I_p$; and (3) whether $lfp(T_{f\Pi^I}(\emptyset, \neg I^-)) = I$, i.e., whether $I$ is a well-justified FLP answer set of $\Pi$. Since this essentially amounts to evaluating an ordinary normal logic program it can be done in polynomial time with the help of a NEXP oracle.

The above proof shows that given a guess of input interpretations for all ground dl-atoms, deciding whether $\Pi$ has a well-justified FLP answer set complying with the guess is in $\mathrm{P}^{\mathrm{NEXP}}$ when $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$. Consequently, deciding whether $\Pi$ has a well-justified FLP answer set is in $\mathrm{NP}^{\mathrm{NEXP}}$ when $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$. Recalling that $\mathrm{NP}^{\mathrm{NEXP}} = \mathrm{P}^{\mathrm{NEXP}}$ [37], the result follows.

By Lemma 10, brave reasoning for dl-programs relative to $\mathcal{SHOIN}(\mathbf{D})$ knowledge bases is $\mathrm{P}^{\mathrm{NEXP}}$-hard. Since brave reasoning for dl-programs falls in the same complexity class as existence of well-justified FLP answer sets, deciding whether a dl-program has a well-justified FLP answer set is also $\mathrm{P}^{\mathrm{NEXP}}$-hard when $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$. As a result, deciding whether a dl-program relative to a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base has a well-justified FLP answer set is $\mathrm{P}^{\mathrm{NEXP}}$-complete. □

To prove Theorem 18 for the case of $\mathcal{SROIQ}(\mathbf{D})$, we recall the concept of a domino system. A domino system is a triple $\mathcal{D} = (D, H, V)$, where $D = \{1, \ldots, p\}$ is a finite set of tiles and $H, V \subseteq D \times D$ are horizontal and vertical matching relations. For a positive integer $m$ and a word $w = w_1 \ldots w_n$ over $D$ of length $n \leq m$, we say $\mathcal{D}$ admits the tiling of $m \times m$ with initial condition $w$ iff there exists a mapping $\tau : \{1, \ldots, m\} \times \{1, \ldots, m\} \to D$ such that for $1 < i \leq m$ and $1 \leq j \leq m$, $\langle \tau(i-1, j), \tau(i, j)\rangle \in H$, for $1 \leq i \leq m$ and $1 < j \leq m$, $\langle \tau(i, j-1), \tau(i, j)\rangle \in V$, and for $1 \leq i \leq n$, $\tau(i, 1) = w_i$.

We also borrow from [41, Theorem 5] the following polynomial-time reduction of the tilability of domino systems to the satisfiability of DL knowledge bases in $\mathcal{SROIQ}(\mathbf{D})$.

**Lemma 11.** *For a domino system $\mathcal{D} = (D, H, V)$ with $D = \{1, \ldots, p\}$ and initial condition $w = w_1 \ldots w_n$, there exist $\mathcal{SROIQ}(\mathbf{D})$ knowledge bases $L_g$, $L_c$ and $L_w$, where $L_g$ consists of axioms (3)–(33) in [41], $L_c$ consists of the following axioms (1)–(5) and $L_w$ consists of the axioms (6)–(9)*

$$\top \sqsubseteq D_1 \sqcup \cdots \sqcup D_p \tag{1}$$
$$D_i \sqcap D_j \sqsubseteq \bot \qquad 1 \leq i < j \leq p \tag{2}$$
$$D_i \sqsubseteq \forall r.D_i \qquad 1 \leq i \leq p \tag{3}$$
$$D_i \sqcap \exists \upsilon.D_j \sqsubseteq \bot \qquad < i, j > \notin V \tag{4}$$
$$D_i \sqcap \exists h.D_j \sqsubseteq \bot \qquad < i, j > \notin H \tag{5}$$
$$O \sqsubseteq I_1 \tag{6}$$
$$I_k \sqsubseteq \forall r.I_k \qquad 1 \leq k \leq n \tag{7}$$
$$I_k \sqsubseteq \forall h.I_{k+1} \qquad 1 \leq k < n \tag{8}$$
$$I_k \sqsubseteq D_{w_k} \qquad 1 \leq k \leq n \tag{9}$$

*such that $\mathcal{D}$ admits the tiling of $2^{2^n} \times 2^{2^n}$ with initial condition $w$ iff the concept $O$ is satisfiable w.r.t. the DL knowledge base $L_g \cup L_c \cup L_w$.*

The next result follows from a reduction from simple Turing machines to domino systems [6, Theorem 6.12].

**Lemma 12.** *Let $M$ be a nondeterministic Turing machine with time-(and thus space-) bound $2^{2^n}$, deciding an* N2EXP*-complete language $L(M)$ over the alphabet $\Sigma = \{0, 1, \text{“ ”}\}$. There exists a domino system $\mathcal{D} = (D, H, V)$ and a linear-time reduction trans that takes any input $b \in \Sigma^*$ to a word $w \in D^*$ with $|b| = n = |w|$ such that $M$ accepts $b$ if and only if $\mathcal{D}$ admits the tiling of $2^{2^n} \times 2^{2^n}$ with initial condition $w$.*

Since brave reasoning for dl-programs falls in the same complexity class as the existence of well-justified FLP answer sets, to prove Theorem 18 for the case of $\mathcal{SROIQ}(\mathbf{D})$, we first prove the following hardness result.

**Lemma 13.** *Let $\Pi$ be a dl-program relative to a $\mathcal{SROIQ}(\mathbf{D})$ knowledge base $L$ and $l$ a ground atom in $\mathcal{HB}_\Pi$. Deciding whether $l$ is in some well-justified FLP answer set of $\Pi$ is $\mathrm{P}^{\mathrm{N2EXP}}$-hard.*

**Proof.** Recall that a stratified dl-program has a unique strong answer set, which is also the unique well-justified FLP answer set. The $\mathrm{P}^{\mathrm{N2EXP}}$-hardness is proved by a generic reduction of a Turing machine $M$ with access to an N2EXP oracle to brave reasoning for a stratified dl-program $\Pi$ relative to a $\mathcal{SROIQ}(\mathbf{D})$ knowledge base $L$ under the strong answer set semantics, exploiting the N2EXP-hardness proof for $\mathcal{SROIQ}(\mathbf{D})$ [41]. The key is to use dl-atoms in $\Pi$ to decide the results of N2EXP oracle calls made by $M$. The reduction is a slight modification of the reduction presented in [24, Theorem 7.2].

More concretely, let $M$ be a polynomial-time bounded deterministic Turing machine with access to an N2EXP oracle, and let $v$ be an input for $M$. Since every oracle call can simulate the computation of $M$ on $v$ before that call, once the results of all previous oracle calls are known, we can assume that the input of every oracle call is given by $v$ and the results of all previous oracle calls. Since the computation of $M$ after all oracle calls can be simulated within an additional oracle call, we can assume that the result of the last oracle call is the result of the computation of $M$ on $v$. Finally, since any input to an oracle call can be enlarged by "*dummy*" bits, we can assume that the inputs to all oracle calls have the same length $n = 2 \times (t + l)$, where $t$ is the size of $v$, and $l = f(t)$ is the number of all oracle calls: We assume that the input to the $m+1$-th oracle call ($m \in \{0, \ldots, l-1\}$) has the form

$$v_t 1 v_{t-1} 1 \cdots v_1 1 c_0 1 c_1 1 \cdots c_{m-1} 1 c_m 0 \cdots c_{l-1} 0$$

where $v_t, v_{t-1}, \ldots, v_1$ are the symbols of $v$ in reverse order, which are all marked as valid by a subsequent "1", $c_0, c_1, \ldots, c_{m-1}$ are the results of the previous $m$ oracle calls, which are all marked as valid by a subsequent "1", and $c_m, \ldots, c_{l-1}$ are "*dummy*" bits, which are all marked as invalid by a subsequent "0".

By Lemma 12, for an N2EXP oracle $M'$, there exists a domino system $\mathcal{D} = (D, H, V)$ and a linear-time reduction *trans* that takes any input $b \in \Sigma^*$ with $|b| = n$ to a word $w = w_1 \cdots w_n \in D^*$ such that $M'$ accepts $b$ if and only if $\mathcal{D}$ admits the tiling of $2^{2^n} \times 2^{2^n}$ with initial condition $w$. By Lemma 11, there exist $\mathcal{SROIQ}(\mathbf{D})$ knowledge bases $L_g$, $L_c$, $L_w$ such that $\mathcal{D}$ admits the tiling of $2^{2^n} \times 2^{2^n}$ with initial condition $w$ if and only if the concept $O$ is satisfiable w.r.t. the knowledge base $L_g \cup L_c \cup L_w$.

Let the stratified dl-program $\Pi$ relative to a $\mathcal{SROIQ}(\mathbf{D})$ knowledge base $L$ be defined as follows:

$$L = L_g \cup L_c \cup L'_w$$
$$\Pi = \bigcup_{j=0}^{l} \Pi^j$$

where $L'_w$ consists of the following axioms

$$O \sqsubseteq I_1 \tag{10}$$
$$I_k \sqsubseteq \forall r.I_k \qquad 1 \le k \le n \tag{11}$$
$$I_k \sqsubseteq \forall h.I_{k+1} \qquad 1 \le k < n \tag{12}$$
$$I_k \sqsubseteq D_j \sqcup \exists s.(\{a_{k,j}\} \sqcap \neg A) \qquad 1 \le k \le n, 1 \le j \le p \tag{13}$$

$p$ is the number of tiles of the domino system, $n$ is the size of the inputs to all oracle calls, $A$, $s$, $a_{k,j}$ are fresh concept, role, individuals that do not occur in $L_g$ and $L_c$. $L'_w$ consists of $(2 \times n + p \times n)$ axioms, which is polynomial. Intuitively, $I_k \sqsubseteq D_j$ means that the tile in the $k$-th position of the initial condition is $j$. The concept $\exists s.(\{a_{k,j}\} \sqcap \neg A)$ acts as a "*switch*" because when $A(a_{k,j})$ is true, $(\{a_{k,j}\} \sqcap \neg A) \equiv \bot$ and $\exists s.(\{a_{k,j}\} \sqcap \neg A) \equiv \bot$ and thus $I_k \sqsubseteq D_j$. The set $\{A(a_{k,w_k}) \mid 1 \le k \le n\}$ of concept membership axioms expresses that the initial condition of the domino system is $w = w_1 \cdots w_n$.

For every $j \in \{0, \ldots, l\}$, $\Pi^j = \Pi^j_v \cup \Pi^j_q \cup \Pi^j_{w \leftarrow b} \cup \Pi^j_{s \leftarrow w}$. Informally, every set of dl-rules $\Pi^j$ generates the input of the $j+1$-th oracle call, which includes the results of the first $j$ oracle calls. Here $\Pi^l$ prepares the input of a "dummy" (non-happening) $l+1$-th oracle call which contains the result of the $l$-th (i.e., the last) oracle call. More concretely, the bitstring $a_{-2t} \cdots a_{2l-1}$ is the input of the $j+1$-th oracle call if and only if $b^j_{-2t}(a_{-2t}), \ldots, b^j_{2l-1}(a_{2l-1})$ are in the strong answer set of $\Pi$. The components $\Pi^j_v$, $\Pi^j_q$, $\Pi^j_{w \leftarrow b}$ and $\Pi^j_{s \leftarrow w}$, with $j \in \{0, \ldots, l\}$, are defined as follows:

(1) $\Pi^0_v$ writes $v$ into the input of the first oracle call, and for each $j \in \{1, \ldots, l\}$, $\Pi^j_v$ copies $v$ into the input of the $j+1$-th oracle call, i.e.

$$\Pi^0_v = \left\{ b^0_{-2i}(v_i) \;\middle|\; i \in \{1, \ldots, t\} \right\} \cup \left\{ b^0_{-2i+1}(1) \;\middle|\; i \in \{1, \ldots, t\} \right\}$$
$$\Pi^j_v = \left\{ b^j_{-i}(x) \leftarrow b^{j-1}_{-i}(x) \;\middle|\; i \in \{1, \ldots, 2t\} \right\}$$

(2) $\Pi_q^0$ initializes the rest of the input of the first oracle call with "*dummy*" bits, and every $\Pi_q^j$ with $j \in \{1, \ldots, l\}$ writes the result of the $j$-th oracle call into the input of the $j+1$-th oracle call and carries over all the other result and dummy bits from the input of the $j$-th oracle call, i.e.

$$\Pi_q^0 = \left\{ b_i^0(0) \;\middle|\; i \in \{0, \ldots, 2l-1\} \right\}$$
$$\Pi_q^j = \left\{ b_i^j(x) \leftarrow b_i^{j-1}(x) \;\middle|\; i \in \{0, \ldots, 2l-1\}, i \notin \{2j-2, 2j-1\} \right\}$$
$$\cup \left\{ b_{2j-2}^j(0) \leftarrow DL\left[A \uplus A^{j-1}; O \sqsubseteq \bot\right](); b_{2j-2}^j(1) \leftarrow \neg b_{2j-2}^j(0); b_{2j-1}^j(1) \right\}$$

Note that for a DL knowledge base $\mathcal{S}$, $\mathcal{S} \models O \sqsubseteq \bot$ if and only if $O$ is unsatisfiable w.r.t. $\mathcal{S}$.

(3) Every $\Pi_{w \leftarrow b}^j$ with $j \in \{0, \ldots, l\}$ realizes the above-mentioned linear-time reduction *trans*, which transforms any input $b^j$ of the Turing machine $M$ into an initial condition $w^j$ of the same length of $M's$ domino system $\mathcal{D}$.

(4) Every $\Pi_{s \leftarrow w}^j$ with $j \in \{0, \ldots, l\}$ transforms the initial condition $w^j$ of $\mathcal{D}$ into an input of the $j+1$-th dl-atom via the predicate $A^j$, i.e.

$$\Pi_{s \leftarrow w}^j = \left\{ A^j(a_{i,d}) \leftarrow w_i^j(d) \;\middle|\; i \in \{1, \ldots, n\}, d \in D \right\}$$

Observe that $M$ accepts $v$ if and only if the last oracle call returns "yes". The latter is equivalent to $b_{2l-2}^l(1)$ being derived from $\Pi$ and thus $b_{2l-2}^l(0)$ being not derived from $\Pi$. So $M$ accepts $v$ if and only if $b_{2l-2}^l(1)$ belongs to the strong answer set of $\Pi$ if and only if $b_{2l-2}^l(1)$ belongs to the well-justified FLP answer set of $\Pi$.

To conclude, it is $\mathrm{P}^{\mathrm{N2EXP}}$-hard to decide whether a given ground atom $l \in \mathcal{HB}_\Pi$ is in some well-justified FLP answer set of a dl-program $\Pi$ relative to a $\mathcal{SROIQ}(\mathbf{D})$ knowledge base $L$. □

**Proof of Theorem 18.** Case 3: deciding whether $\Pi$ has a well-justified FLP answer set is $\mathrm{P}^{\mathrm{N2EXP}}$-complete when $L$ belongs to $\mathcal{SROIQ}(\mathbf{D})$.

The proof of membership is analogous to the above proof of membership for the case of $\mathcal{SHOIN}(\mathbf{D})$.

By Lemma 13, brave reasoning for dl-programs relative to $\mathcal{SROIQ}(\mathbf{D})$ knowledge bases is $\mathrm{P}^{\mathrm{N2EXP}}$-hard. Since brave reasoning for dl-programs falls in the same complexity class as the existence of well-justified FLP answer sets, deciding whether a dl-program has a well-justified FLP answer set is also $\mathrm{P}^{\mathrm{N2EXP}}$-hard when $L$ belongs to $\mathcal{SROIQ}(\mathbf{D})$. Consequently, deciding whether a dl-program relative to a $\mathcal{SROIQ}(\mathbf{D})$ knowledge base has a well-justified FLP answer set is $\mathrm{P}^{\mathrm{N2EXP}}$-complete. □

**Proof of Theorem 19.** Recall that cautious (resp. brave) reasoning for dl-programs falls in the same complexity class as the non-existence (resp. existence) of well-justified FLP answer sets. By Theorem 18, it immediately follows that deciding whether a ground atom is in every (resp. some) well-justified FLP answer set is co-NEXP-complete (resp. NEXP-complete) when $L$ belongs to $\mathcal{SHIF}(\mathbf{D})$, $\mathrm{P}^{\mathrm{NEXP}}$-complete (resp. $\mathrm{P}^{\mathrm{NEXP}}$-complete) when $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$, and $\mathrm{P}^{\mathrm{N2EXP}}$-complete (resp. $\mathrm{P}^{\mathrm{N2EXP}}$-complete) when $L$ belongs to $\mathcal{SROIQ}(\mathbf{D})$. □

To prove Theorem 20, we introduce the following lemma. Given a dl-program $\Pi$ relative to a DL knowledge base $L$, for every subset $S$ of $\mathcal{HB}_\Pi$, let $S^{\leq 2}$ denote its restriction to unary and binary predicates. Moreover, we associate with $\Pi$ its dl-satisfaction table $T(\Pi, L)$ given by all tuples $\langle I, A, v \rangle$ such that $I$ is a subset of $\mathcal{HB}_\Pi^{\leq 2}$, $A$ is a ground dl-atom from $ground(\Pi)$, and $v = 1$ if $I$ satisfies $A$ under $L$, while $v = 0$ otherwise.

**Lemma 14.** *Given a ground dl-program $\Pi$ relative to a DL knowledge base $L$, its dl-satisfaction table $T(\Pi, L)$, and a Herbrand interpretation $I$, deciding whether $I$ is an FLP answer set of $\Pi$ relative to $L$ is in* EXP.

**Proof.** We first compute $f\Pi^I$, which can be done in polynomial time. For this purpose the projection $I^{\leq 2}$ is generated, and then $I \models body(r)$ is checked by deciding $A \in I$ for ordinary body atoms $A$, respectively by looking up $\langle I^{\leq 2}, A, 1 \rangle \in T(\Pi, L)$ for dl-atoms $A$. The correctness of the latter is an immediate consequence of the fact the only unary and binary predicates occur in the input list of any dl-atom.

Clearly, checking $head(r) \in I$ for every $r \in f\Pi^I$ can also be done in polynomial time, and it (if it succeeds) additionally verifies that $I$ is a model of $\Pi$ relative to $L$ (otherwise $I$ is also not an FLP answer set).

Second, we need to check for minimality, that is we need to verify $J \not\models f\Pi^I$ for every $J \subset I$. We do so by an exponential number of tests of answer set existence for ground ordinary Horn programs with constraints. The size of each of the programs is bounded by the size $f\Pi^I$ plus a single constraint. Hence, answer set existence can be checked in polynomial time for each of these programs. Every program $P(I, I')$ corresponds to a subset $I'$ of $I^{\leq 2}$ by the following construction: $P(I, I')$ is obtained from $f\Pi^I$ by

(*i*) removing all literals $\neg A$ from rule bodies where $A$ is an ordinary atom;
(*ii*) replacing every unary or binary ordinary atom $A$ with 1 if $A \in I'$, and with 0 otherwise;

(*iii*) replacing every dl-atom $A$ with $v$, where $\langle I', A, v\rangle \in T(\Pi, L)$;
(*iv*) removing then all rules $r$ such that $head(r) = 1$, or $body(r)$ contains 0 or $\neg 1$, and removing 0 from the heads, respectively $\neg 0$ and 1 from the bodies of the remaining rules; and
(*v*) if $I' = I^{\leq 2}$, then adding the constraint $\leftarrow A_1, \ldots, A_m$, where $I \setminus I^{\leq 2} = \{A_1, \ldots, A_m\}$.

The following property establishes that minimality checking can be done by checking answer set existence for all programs $P(I, I')$ such that $I' \subseteq I^{\leq 2}$: there exists some $J \subset I$ such that $J \models f\Pi^I$ if and only if $P(I, I')$, where $I' = J^{\leq 2}$, admits an answer set. Observe that the transformations in items (*i*)–(*iv*) are equivalence preserving for $J \subseteq I$ and that the constraint in item (*v*) ensures $J \subset I$ (unless this already holds due to $I' = J^{\leq 2} \subset I^{\leq 2}$). Given these observations, the proof of the above property is simple and left to the user.

We have thus shown that deciding whether $I$ is an FLP answer set of $\Pi$ relative to $L$ is in EXP provided that its dl-satisfaction table $T(\Pi, L)$ is given. □

**Proof of Theorem 20.** In case of $\mathcal{SHIF}(\mathbf{D})$ determining the value $v$ for a tuple $\langle I', A, v\rangle$ of $T(\Pi, L)$ is possible in deterministic exponential time. Therefore, the whole table $T(\Pi, L)$ can be computed by performing exponentially many exponential time computations; hence, $T(\Pi, L)$ is computable in deterministic exponential time. Consequently, guessing an interpretation $I$ and deciding whether it is an FLP answer set (computing $T(\Pi, L)$ first and applying Lemma 14) is feasible in nondeterministic exponential time.

When $L$ belongs to $\mathcal{SHOIN}(\mathbf{D})$, computing $T(\Pi, L)$ is not feasible in deterministic exponential time. However, given that the number $n_0$ of tuples where $v = 0$ is known, one can proceed as before in nondeterministic exponential time. Establishing $n_0$ requires a polynomial number (in the size of $\Pi$) of decision problems to be solved, where each problem is in NEXP (and depends on the previous result). Thus, $\mathrm{P}^{\mathrm{NEXP}}$ membership of the problem can be established as follows.

We first compute $n_0$ in binary search by deciding problems of the form: given $k$ and $\Pi$, are there at least $k$ tuples in $T(\Pi, L)$ such that $v = 0$. Since the number of tuples in $T(\Pi, L)$ is exponential in the size of $\Pi$, the required size of $k$ in binary representation is polynomial in the size of $\Pi$. Moreover, given a ground dl-atom $A$, an interpretation $I \subseteq \mathcal{HB}_\Pi^{\leq 2}$, and an exponential size witness candidate $w$ for $I \not\models A$ (recall that $I \not\models A$ is in NEXP and think of a potential computation path of a corresponding nondeterministic Turing machine computation), checking that $w$ indeed witnesses $I \not\models A$ is in EXP. Therefore, the sub-problems used in our binary search, i.e., given $k$ and $\Pi$, deciding whether there are at least $k$ tuples in $T(\Pi, L)$ such that $v = 0$, are in NEXP. By polynomially many calls to a NEXP oracle, we thus can establish the exact number $n_0$ of tuples in $T(\Pi, L)$ such that $v = 0$ in $\mathrm{P}^{\mathrm{NEXP}}$.

Once $n_0$ is known, we can use one more call to the oracle to decide FLP answer set existence, by guessing $I$ together with $n_0$ tuples $t_1, \ldots, t_{n_0}$ of $T(\Pi, L)$ where $v = 0$ and corresponding witness candidates $w_1, \ldots, w_{n_0}$ for $I \not\models A$. The oracle then first checks in exponential time for each $w_i$, that it correctly witnesses $I \not\models A$ and then proceeds as in Lemma 14, given that $T(\Pi, L)$ can now be constructed in exponential time since all other entries are known to have $v = 1$. This proves $\mathrm{P}^{\mathrm{NEXP}}$-membership in case of $\mathcal{SHOIN}(\mathbf{D})$.

The membership proof for $\mathcal{SROIQ}(\mathbf{D})$ is analogous, using an N2EXP oracle instead of the NEXP oracle.

Matching lower bounds, i.e, hardness for NEXP, $\mathrm{P}^{\mathrm{NEXP}}$, and $\mathrm{P}^{\mathrm{N2EXP}}$, respectively, follows from the corresponding reductions for well-justified FLP answer set semantics. It is sufficient to observe that the programs constructed make use of monotonic dl-atoms only; hence, its well-justified FLP answer sets coincide with its FLP answer sets. □

**Proof of Theorem 21.** The termination property of Algorithm 1 follows from the assumption that all complex atoms occurring in a logic program are decidable.

We note that the projections of all compatible sets of a normal logic program $\Pi$ with complex atoms include all FLP answer sets of $\Pi$. Then by Corollary 2 they also include all well-justified FLP answer sets. Indeed, given an interpretation $I$ of $\Pi$, let $\hat{I}$ denote its extension to $\hat{\Pi}$ in which, for each complex atom $A$, (i) $E_A$ is true if and only if $I$ satisfies $A$ and (ii) $E'_A$ is opposite to $E_A$. Then the reduct $f\hat{\Pi}^{\hat{I}}$ consists of all rules in $\widehat{f\Pi^I}$ plus for every complex atom $A$ the rule $E_A \leftarrow \neg E'_A$ if $I$ satisfies $A$ and the rule $E'_A \leftarrow \neg E_A$ otherwise. Hence if $I' \subseteq I$ is a model of $f\Pi^I$, then $f\hat{\Pi}^{\hat{I}}$ has a model $\hat{J}$ which on all $E_A$ and $E'_A$ coincides with $\hat{I}$ and whose projection $J$ on $\Pi$ coincides with $I'$; if $I$ is minimal (i.e., an FLP answer set of $\Pi$), then also the corresponding $J$ is minimal, and thus an answer set of $\hat{\Pi}$.

This means that for every well-justified FLP answer set $I$ of $\Pi$, there must be a compatible set $\hat{I}$ whose projection on $\Pi$ is $I$ such that $I = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$. Obviously, this well-justified answer set will be identified in the checking step of Algorithm 1. This shows the completeness of Algorithm 1.

For each output $I$ of Algorithm 1, $I$ must be the projection of some compatible set $\hat{I}$ such that $I = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$. As shown in [20], $I$ is an FLP answer set of $\Pi$. Since $I = lfp(T_{f\Pi^I}(\emptyset, \neg I^-))$, $I$ is also a well-justified FLP answer set of $\Pi$. This shows the soundness of Algorithm 1. □

## References

[1] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Professional Book Center, Edinburgh, Scotland, UK, 2005, pp. 364–369.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P.F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, 2nd edition, Cambridge University Press, 2010.
[3] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving with Answer Sets, Cambridge University Press, 2003.
[4] M. Bartholomew, J. Lee, Y. Meng, First-order extension of the FLP stable model semantics via modified circumscription, in: 22nd International Joint Conference on Artificial Intelligence (IJCAI), 2011, pp. 724–730.
[5] T.J.M. Bench-Capon, P.E. Dunne, Argumentation in artificial intelligence, Artif. Intell. 171 (10–15) (2007) 619–641.
[6] E. Börger, E. Grädel, Y. Gurevich, The Classical Decision Problem, Springer, 2001.
[7] G. Brewka, T. Eiter, Equilibria in heterogeneous nonmonotonic multi-context systems, in: AAAI, 2007, pp. 385–390.
[8] G. Brewka, T. Eiter, M. Fink, Nonmonotonic multi-context systems: a flexible approach for integrating heterogeneous knowledge sources, in: M. Balduccini, T.C. Son (Eds.), Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning, in: Lect. Notes Comput. Sci., vol. 6565, Springer, 2011, pp. 233–258.
[9] G. Brewka, T. Eiter, M. Truszczyński, Answer set programming at a glance, Commun. ACM 54 (12) (2011) 92–103.
[10] D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: the DL-Lite family, J. Autom. Reason. 39 (3) (2007) 385–429.
[11] E. Dantsin, T. Eiter, G. Gottlob, A. Voronkov, Complexity and expressive power of logic programming, ACM Comput. Surv. 33 (3) (2001) 374–425.
[12] M. Dao-Tran, T. Eiter, M. Fink, T. Krennwallner, Modular nonmonotonic logic programming revisited, in: 25th International Conference on Logic Programming (ICLP), Springer, 2009, pp. 145–159.
[13] M. Dao-Tran, T. Eiter, M. Fink, T. Krennwallner, Distributed nonmonotonic multi-context systems, in: F. Lin, U. Sattler (Eds.), 12th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, Ontario, Canada, May 9–13, 2010, AAAI Press, 2010, pp. 60–70.
[14] J. de Bruijn, T. Eiter, H. Tompits, Embedding approaches to combining rules and ontologies into autoepistemic logic, in: 11th International Conference on Principles of Knowledge Representation and Reasoning (KR), AAAI Press, 2008, pp. 485–495.
[15] J. de Bruijn, D. Pearce, A. Polleres, A. Valverde, A semantical framework for hybrid knowledge bases, Knowl. Inf. Syst. 25 (1) (2010) 81–104.
[16] M. Denecker, N. Pelov, M. Bruynooghe, Ultimate well-founded and stable semantics for logic programs with aggregates, in: 17th International Conference on Logic Programming (ICLP), Springer, 2001, pp. 212–226.
[17] P. Dung, P. Mancarella, F. Toni, Computing ideal sceptical argumentation, Artif. Intell. 171 (2007) 642–674.
[18] P.M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artif. Intell. 77 (2) (1995) 321–357.
[19] T. Eiter, M. Fink, G. Ianni, T. Krennwallner, P. Schüller, Pushing efficient evaluation of HEX programs by modular decomposition, in: J. Delgrande, W. Faber (Eds.), 11th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2011, Vancouver, British Columbia, Canada, May 16–19, in: Lect. Notes Comput. Sci., vol. 6645, 2011, pp. 93–106.
[20] T. Eiter, M. Fink, T. Krennwallner, C. Redl, Conflict-driven ASP solving with external sources, Theory Pract. Log. Program. 12 (4–5) (2012) 659–679.
[21] T. Eiter, M. Fink, T. Krennwallner, C. Redl, P. Schüller, Exploiting unfounded sets for HEX-program evaluation, in: 13th European Conference on Logics in Artificial Intelligence (JELIA), 2012, pp. 160–175.
[22] T. Eiter, M. Fink, P. Schüller, A. Weinzierl, Finding explanations of inconsistency in nonmonotonic multi-context systems, Tech. Rep. INFSYS RR-1843-12-09, INFSYS RR-1843-03-08, Inst. für Informationssysteme, TU Wien, preliminary version in: Proc. 12th International Conference on Knowledge Representation and Reasoning, KR 2010, AAAI Press, 2012, pp. 329–339.
[23] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: propositional case, Ann. Math. Artif. Intell. 15 (3–4) (1995) 289–323.
[24] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, H. Tompits, Combining answer set programming with description logics for the semantic web, Artif. Intell. 172 (12–13) (2008) 1495–1539.
[25] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, A uniform integration of higher-order reasoning and external evaluations in answer-set programming, in: 19th International Joint Conference on Artificial Intelligence (IJCAI), Professional Book Center, 2005, pp. 90–96.
[26] W. Faber, N. Leone, G. Pfeifer, Recursive aggregates in disjunctive logic programs: semantics and complexity, in: 9th European Conference on Logics in Artificial Intelligence (JELIA), Springer, 2004, pp. 200–212.
[27] W. Faber, G. Pfeifer, N. Leone, Semantics and complexity of recursive aggregates in answer set programming, Artif. Intell. 175 (1) (2011) 278–298.
[28] F. Fages, Consistency of Clark's completion and existence of stable models, J. Methods Log. Comput. Sci. 1 (1994) 51–60.
[29] P. Ferraris, Answer sets for propositional theories, in: 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), Springer, 2005, pp. 119–131.
[30] P. Ferraris, J. Lee, V. Lifschitz, Stable models and circumscription, Artif. Intell. 175 (1) (2011) 236–263.
[31] P. Ferraris, V. Lifschitz, Mathematical foundations of answer set programming, in: We Will Show Them!, vol. 1, 2005, pp. 615–664.
[32] M. Fitting, First-Order Logic and Automated Theorem Proving, 2nd edition, Springer, 1996.
[33] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: 5th International Conference on Logic Programming (ICLP), MIT Press, 1988, pp. 1070–1080.
[34] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Gener. Comput. 9 (1991) 365–385.
[35] C. Ghidini, F. Giunchiglia, Local models semantics, or contextual reasoning = locality + compatibility, Artif. Intell. 127 (2) (2001) 221–259.
[36] B.C. Grau, I. Horrocks, B. Motik, B. Parsia, P.F. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, J. Web Semant. 6 (4) (2008) 309–322.
[37] L.A. Hemachandra, The strong exponential hierarchy collapses, J. Comput. Syst. Sci. 39 (3) (1989) 299–322.
[38] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible SROIQ, in: 10th International Conference on Principles of Knowledge Representation and Reasoning (KR), 2006, pp. 57–67.
[39] I. Horrocks, P.F. Patel-Schneider, Reducing OWL entailment to description logic satisfiability, in: International Semantic Web Conference (ISWC), 2003, pp. 17–29.
[40] I. Horrocks, P.F. Patel-Schneider, F. van Harmelen, From SHIQ and RDF to OWL: the making of a web ontology language, J. Web Semant. 1 (1) (2003) 7–26.
[41] Y. Kazakov, RIQ and SROIQ are harder than SHOIQ*, in: 11th International Conference on Principles of Knowledge Representation and Reasoning (KR), 2008, pp. 274–284.
[42] J. Lee, R. Palla, Integrating rules and ontologies in the first-order stable model semantics (preliminary report), in: AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning, 2011.
[43] V. Lifschitz, Answer set programming and plan generation, Artif. Intell. 138 (1–2) (2002) 39–54.
[44] V. Lifschitz, Thirteen definitions of a stable model, in: Fields of Logic and Computation, 2010, pp. 488–503.
[45] L. Liu, E. Pontelli, T. Son, M. Truszczyński, Logic programs with abstract constraint atoms: the role of computations, Artif. Intell. 174 (3–4) (2010) 295–315.
[46] T. Lukasiewicz, A novel combination of answer set programming with description logics for the semantic web, IEEE Trans. Knowl. Data Eng. 22 (11) (2010) 1577–1592.
[47] V.W. Marek, M. Truszczyński, Stable models and an alternative logic programming paradigm, in: The Logic Programming Paradigm: a 25-Year Perspective, Springer, 1999, pp. 375–398.

[48] V.W. Marek, M. Truszczyński, Logic programs with abstract constraint atoms, in: 19th National Conference on Artificial Intelligence (AAAI), MIT Press, 2004, pp. 86–91.
[49] B. Motik, R. Rosati, Reconciling description logics and rules, J. ACM 57 (5) (2010).
[50] I. Niemela, Logic programs with stable model semantics as a constraint programming paradigm, Ann. Math. Artif. Intell. 25 (1999) 241–273.
[51] D. Pearce, A new logical characterisation of stable models and answer sets, in: Non-Monotonic Extensions of Logic Programming, 1996, pp. 57–70.
[52] D. Pearce, Equilibrium logic, Ann. Math. Artif. Intell. 47 (1–2) (2006) 3–41.
[53] W. Pelov, M. Denecker, M. Bruynooghe, Well-founded and stable semantics of logic programs with aggregates, Theory Pract. Log. Program. 7 (3) (2007) 301–353.
[54] Y.D. Shen, Well-supported semantics for description logic programs, in: 22nd International Joint Conference on Artificial Intelligence (IJCAI), 2011, pp. 1081–1086.
[55] Y.D. Shen, K.W. Wang, Extending logic programs with description logic expressions for the semantic web, in: International Semantic Web Conference (ISWC), 2011, pp. 633–648.
[56] Y.D. Shen, K.W. Wang, FLP semantics without circular justifications for general logic programs, in: 26th AAAI Conference on Artificial Intelligence (AAAI), AAAI Press, 2012, pp. 821–827.
[57] Y.D. Shen, J.H. You, A generalized Gelfond–Lifschitz transformation for logic programs with abstract constraints, in: 22nd AAAI Conference on Artificial Intelligence (AAAI), 2007, pp. 483–488.
[58] Y.D. Shen, J.H. You, A default approach to semantics of logic programs with constraint atoms, in: 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR), Springer, 2009, pp. 277–289.
[59] T.C. Son, E. Pontelli, A constructive semantic characterization of aggregates in answer set programming, Theory Pract. Log. Program. 7 (3) (2007) 355–375.
[60] T.C. Son, E. Pontelli, P.H. Tu, Answer sets for logic programs with arbitrary abstract constraint atoms, J. Artif. Intell. Res. 29 (2007) 353–389.
[61] L. Stockmeyer, A. Meyer, Word problems requiring exponential time, in: 5th ACM Symposium on the Theory of Computing, 1973, pp. 1–9.
[62] S. Tobies, Complexity results and practical algorithm for logics in knowledge representation, Ph.D. thesis, RWTH Aachen, Germany, 2001.
[63] M. Truszczyński, Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs, Artif. Intell. 174 (16–17) (2010) 1285–1306.
[64] M.H. van Emden, R.A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (4) (1976) 733–742.