Artificial Intelligence 277 (2019) 103165

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Determining inference semantics for disjunctive logic programs

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

[a] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[b] University of Chinese Academy of Sciences, Beijing 100049, China
[c] Institut für Logic and Computation, Technische Universität Wien, Favoritenstrasse 9-11, A-1040 Vienna, Austria

ARTICLE INFO

Article history:
Received 16 February 2018
Received in revised form 12 April 2019
Accepted 17 August 2019
Available online 23 August 2019

Keywords:
Answer set programming
Knowledge representation and reasoning
Nonmonotonic reasoning
Disjunctive logic programs

ABSTRACT

In a seminal paper, Gelfond and Lifschitz [34] introduced simple disjunctive logic programs, where in rule heads the disjunction operator "|" is used to express incomplete information, and defined the answer set semantics (called *GL-semantics* for short) based on a program transformation (called *GL-reduct*) and the minimal model requirement. Our observations reveal that the requirement of the GL-semantics, i.e., an answer set should be a minimal model of rules of the GL-reduct, may sometimes be too strong a condition and exclude some answer sets that would be reasonably acceptable. To address this, we present an alternative, more permissive answer set semantics, called the *determining inference (DI) semantics*. Specifically, we introduce a head selection function to formalize the operator | and define answer sets as follows: (i) Given an interpretation I and a selection function *sel*, we transform a disjunctive program Π into a normal program Π^I_{sel}, called a *disjunctive program reduct*; (ii) given a base answer set semantics $\mathcal{X}$ for normal programs, we define I to be a *candidate answer set* of Π w.r.t. $\mathcal{X}$ if I is an answer set of Π^I_{sel} under $\mathcal{X}$; and (iii) we define I to be an answer set of Π w.r.t. $\mathcal{X}$ if I is a minimal candidate answer set. The DI-semantics is general and applicable to extend any answer set semantics $\mathcal{X}$ for normal programs to disjunctive programs. By replacing $\mathcal{X}$ with the GL_{nlp}-semantics defined by Gelfond and Lifschitz [33], we induce a DI-semantics for simple disjunctive programs, and by replacing $\mathcal{X}$ with the well-justified semantics defined by Shen et al. [65], we further induce a DI-semantics for general disjunctive programs. We also establish a novel characterization of the GL-semantics in terms of a disjunctive program reduct, which reveals the essential difference of the DI-semantics from the GL-semantics and leads us to giving a satisfactory solution to the open problem presented by Hitzler and Seda [36] about characterizing split normal derivatives of a simple disjunctive program Π such that answer sets of the normal derivatives are answer sets of Π under the GL-semantics. Finally we give computational complexity results; in particular we show that in the propositional case deciding whether a simple disjunctive program Π has some DI-answer set is NP-complete. This is in contrast to the GL-semantics and equivalent formulations such as the FLP-semantics [24], where deciding whether Π has some answer set is Σ^p_2-complete, while brave and cautious reasoning are Σ^p_2- and Π^p_2-complete, respectively, for both GL- and DI-answer sets. For general disjunctive programs with compound formulas as building blocks,

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

https://doi.org/10.1016/j.artint.2019.103165

the complexity of brave and cautious reasoning increases under DI-semantics by one level of the polynomial hierarchy, which thus offers higher problem solving capacity.

1. Introduction

Answer set programming (ASP) is a major declarative programming paradigm in knowledge representation and reasoning [43,49,53], which is specifically oriented towards modeling and solving combinatorial search problems arising in many application areas of AI such as planning, reasoning about actions, diagnosis, and beyond [43,5,32,7,8,63]. The idea of ASP is to represent a problem by a logic program whose meaning is given by a set of intended models, called stable models or answer sets, which correspond to solutions to the given problem [43].

In the very beginning of the ASP evolution, only simple normal programs were considered [33], which consist of rules of the form

$$A \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n \tag{1}$$

where A and the B_i's and C_j's are atoms, and $\leftarrow$ is an if-then rule operator.[1] Rule (1) says that if the body condition $B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n$ holds, then the head A can be inferred. More specifically, the intuitive reading of this rule is that the head A can be inferred using this rule if (and only if) the body condition has been inferred by any other rules in the program [37].

In order to model incomplete information, Gelfond and Lifschitz [34] further introduced simple disjunctive programs consisting of rules with disjunctive heads of the form

$$A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n \tag{2}$$

where the A_i's, B_j's and C_l's are atoms, and $\mid$ is a disjunctive rule head operator.[2] Rule (2) says that if the body condition holds, then some A_i in the head can be inferred. These rules were further extended to more general forms containing aggregates [23,56,66,68,24,3], external source atoms [21,20], propositional formulas [54,26,69], or first-order formulas [6,27,65].

An answer set semantics of a logic program Π assigns to Π a collection of interpretations called *answer sets*, which can be built on the basis of rules of Π. As described in [30], the construction of such an answer set I is guided by the following two informal principles: (P1) I must satisfy the rules of Π; and (P2) it should adhere to the *rationality principle*, which says that one shall not believe anything one is not forced to believe. Note that the principle (P1) amounts to that I must be a model of Π, while the rationality principle is used for knowledge minimization in nonmonotonic reasoning. In the sequel, we refer to the two principles as the *Gelfond ASP principles*.

GL-semantics for simple disjunctive programs. Gelfond and Lifschitz [33] defined the seminal answer set semantics for a simple normal program Π by means of program transformation that was inspired by investigations in the use of autoepistemic logic [51] for defining the semantics of negation in logic programs [29]. The idea is to reduce Π, given an interpretation I, to another program Π^I that incorporates the assumption about negation with respect to I, and then to evaluate Π^I. Specifically, the *GL-reduct* Π^I is obtained from Π by removing first all rules (1) containing a $C_i \in I$ and then all $\neg C_i$ from the remaining rules. As no negation occurs in Π^I, the program has a canonical semantics given by the least Herbrand model, which can be constructed by fixpoint iteration over rules of Π^I; now if that model coincides with I, then I is 'stable' as it can be reproduced from the assumption, and is thus regarded as an answer set. The same concept is obtainable in many different ways [44], which provides evidence for a deeper interest and importance of the notion.

To define answer sets of a simple disjunctive program Π, Gelfond and Lifschitz [34] have generalized this construction by defining the GL-reduct Π^I of Π with respect to an interpretation I as for simple normal programs, and by requesting that I be a minimal model of Π^I. In this way, the program Π is again reduced to a program Π^I without negation, whose canonical models are its minimal models, which were first considered in the pioneering work by Minker [50]. Notably, the fixpoint-construction of the least Herbrand model does not readily generalize to minimal models, and more involving operators have been defined, cf. [46].

This *GL-semantics* for simple disjunctive programs is based on the two Gelfond ASP principles, where the rationality principle for knowledge minimization requires that every answer set I of Π be a minimal model of Π.[3]

[1] In [33], "," instead of "$\wedge$" was used for logical conjunction.

[2] In [34], "*not*" instead of "$\neg$" was used for default negation, "," instead of "$\wedge$" for logical conjunction, and "$\neg$" for what the authors referred to as classical negation. It appears that such classical negation is not the same as the negation in classical logic (we will explain the difference later in footnote 11), so in the literature the term *strong* negation is often used and expressed with the symbol "$\sim$".

[3] It is easy to show that if I is a minimal model of Π^I, then it must be a minimal model of Π.

Constructive view of the operator |. In rule (2), Gelfond and Lifschitz [34] deliberately used the operator | in order to indicate that the connection of alternatives in the rule head is not simply disjunction $\vee$ as in classical logic. The view is rather that one "commits" to an alternative A_i in a head $A_1 \mid \cdots \mid A_k$ for the belief state that represents the model one is willing to accept; in [30,32] "*or*" is used in place of | and called *epistemic disjunction*; this is supported by the fact that a common embedding of (ground) logic programs under answer sets into autoepistemic logic by Marek and Truszczynski [48] translates each alternative A_i into $A_i \wedge \mathsf{L}A_i$, where $\mathsf{L}A_i$ intuitively means that A_i is in the belief state of the reasoner. Alternative embeddings use so called *positive introspection axioms* $A_i \supset \mathsf{L}A_i$ [58] that put an alternative A_i in the belief state; for an extensive discussion of the subject, covering also non-ground programs, we refer to [15].

Intuitively, a rule head $A_1 \mid \cdots \mid A_k$ models incomplete information concerning the truth value of each alternative A_i, namely that the information available is insufficient to establish whether A_1 is true or A_2 is true $\cdots$ or A_k is true, but nonetheless sufficient to establish that at least one A_i is true. This means that the rule head offers different choices, i.e., it either infers A_1 or A_2 $\cdots$ or A_k. Therefore, a rule $A_1 \mid \cdots \mid A_k \leftarrow \mathit{Body}$ infers some A_i in the head if the body condition Body holds. As the choice of such an alternative A_i among $A_1, \cdots, A_k$ is arbitrary, the inference with such rules for the construction of a potential answer set is nondeterministic. It is such nondeterministic inferences via disjunctive rule heads that allow us to generate different candidate answer sets. In this sense, we may well refer to | as a *nondeterministic inference operator*. Notably, such an inference operator works at the meta-level, while the logical connective $\vee$ works at the object level and as such yields an objective formula; by a translation into a suitable logic, the inference operator may be expressed by using logical disjunction $\vee$ at its object level; the embeddings of disjunctive rules into autoepistemic logic mentioned above are examples.

The above constructive view of | as a nondeterministic inference operator allows us to clearly differentiate a rule head $A_1 \mid \cdots \mid A_k$ from a logical formula $A_1 \vee \cdots \vee A_k$ $(k > 1)$. The former infers nondeterministically one A_i among the alternatives, while the latter cannot establish the truth of any A_i and thus can infer neither A_1 nor A_2 $\cdots$ nor A_k.

Behavior of the operator | **in the GL-semantics.** Let Π be a simple disjunctive program and for any interpretation I, let $\Pi^I_\vee$ be the GL-reduct Π^I of Π with all occurrences of | replaced by $\vee$. As $\Pi^I_\vee$ and Π^I have the same models, I is an answer set of Π under the GL-semantics if and only if I is a minimal model of Π^I if and only if I is a minimal model of $\Pi^I_\vee$. That is, the operator | in rule heads plays in the GL-semantics the same role as the classical connective $\vee$; or, in other words, in the GL-semantics the operator | amounts to the connective $\vee$.

Consequently, given a rule

$$r: \quad A_1 \mid \cdots \mid A_k,$$

any other rule

$$r': \quad A_1 \mid \cdots \mid A_k \mid A_{k+1} \cdots \mid A_l \leftarrow \mathit{Body} \qquad (l > k)$$

is redundant and thus can be deleted because no answer set under the GL-semantics will contain some A_i with $k < i \leq l$. The reason is that in the GL-reduct Π^I the rule r reappears and – if r' is not removed – a rule

$$r'': \quad A_1 \mid \cdots \mid A_k \mid A_{k+1} \cdots \mid A_l \leftarrow \mathit{Body}^+$$

occurs, where Body^+ is the positive part of the body of r'. As every model of $A_1 \mid \cdots \mid A_k$ is a model of $A_1 \mid \cdots \mid A_k \mid A_{k+1} \cdots \mid A_l$, we can remove the rule r'' from Π^I without changing the models of Π^I; and as I is arbitrary, we can safely remove r' from Π without changing the answer sets of Π under the GL-semantics.

Two observations. For a simple disjunctive program Π, the GL-semantics defines an answer set I of Π to be a minimal model of rules of the GL-reduct Π^I, where the operator | in rule heads amounts to the classical connective $\vee$; so given a rule $A_1 \mid \cdots \mid A_k$, any other rule with a head $A_1 \mid \cdots \mid A_k \mid A_{k+1} \cdots \mid A_l$ $(l > k)$ is redundant and thus can be deleted. We observe that this effect does not seem to be fully desirable. Under the constructive view of the operator |, we expect the two rules to nondeterministically infer an alternative A_i, $1 \leq i \leq k$, from the first head, and an A_j, $1 \leq j \leq l$, from the second; removing the second rule may eliminate some candidate answer sets and thus miss some desired answer sets, as illustrated in the following example.

Example 1. Consider the following simple disjunctive program.

$$\Pi: \quad a \tag{1}$$
$$a \mid b \tag{2}$$
$$b \leftarrow \neg b \tag{3}$$

Rule (2) models incomplete information concerning the truth values of a and b, namely that it is insufficient to establish whether a is true or b is true, but nonetheless sufficient to establish that at least one of the two is true. That is, rule (2) presents two alternatives and infers either a or b. Rule (1) establishes the truth of a. So rules (1) and (2) together either

infer a (when rule (2) infers a) or $a \wedge b$ (when rule (2) infers b), which yields two potential answer sets for Π: $I_1 = \{a\}$ and $I_2 = \{a, b\}$. Rule (3) is a constraint stating that there is no answer set that does not contain b[4]; this excludes I_1. As a result, applying the three rules yields the only candidate answer set I_2 for Π. As I_2 is minimal in the sense that no proper subset J of I_2 is a candidate answer set for Π, we expect I_2 to be an answer set of Π. Note that this construction of answer sets is based on the two Gelfond ASP principles, where the rationality principle for knowledge minimization means that every answer set of Π must be minimal among all candidate answer sets of Π.

However, this simple disjunctive program has no answer set under the GL-semantics. As the operator | in rule heads amounts to the classical connective $\vee$, rule (2) is redundant given rule (1) and can be removed; i.e., under the GL-semantics this simple disjunctive program Π can be simplified to a simple normal program Π' consisting of rules (1) and (3). Removing rule (2) eliminates the only candidate answer set $I_2 = \{a, b\}$ of Π, leading to Π having no answer set.

The next example further demonstrates the need to differentiate the operator | from the logical connective $\vee$.

Example 2. Consider the following more general disjunctive program.

$$\Pi: \quad a \supset b \mid b \supset a \tag{1}$$
$$a \tag{2}$$
$$b \leftarrow \neg b \tag{3}$$

Rule (1) expresses a class subsumption relation with uncertainty, i.e., either a is subsumed by b or the other way around. Rule (2) states that we already have a, and rule (3) is a constraint stating that there is no answer set that does not contain b. Note that one cannot replace | with $\vee$ in rule (1), as that would make the rule a tautology $(a \supset b) \vee (b \supset a)$.

Rule (1) presents two alternatives for answer set construction, i.e., $a \supset b$ or $b \supset a$. Suppose that we choose $a \supset b$; then rules (1) and (2) infer $(a \supset b) \wedge a$, which logically entails both a and b and thus yields a potential answer set $I = \{a, b\}$. I satisfies the constraint rule (3), so it is a candidate answer set for Π. As I is a minimal model of Π, it must be a minimal candidate answer set. Hence we expect I to be an answer set of Π.

However, Π has no answer set under all of the existing answer set semantics for disjunctive programs such as those defined in [45,55,69,6,24,27].[5]

The second observation is that under the GL-semantics, according to the rationality principle for knowledge minimization it is required that every answer set of Π must be a minimal model of Π; this requirement may be too strong in general and exclude some candidate answer sets that may be desired. We have the following example illustrating this observation.

Example 3. Consider the following simple disjunctive program.

$$\Pi: \quad a \mid b \tag{1}$$
$$b \leftarrow a \tag{2}$$
$$c \leftarrow a \tag{3}$$
$$c \leftarrow \neg c \tag{4}$$

Intuitively, rule (1) presents two alternatives for answer set construction, namely a or b, and rules (2) and (3) infer b and c, respectively if a has already been derived. We distinguish between the following two cases.

First, suppose that we choose a from rule (1); then by rules (2) and (3) we obtain a potential answer set $I_1 = \{a, b, c\}$. I_1 satisfies the constraint rule (3), so it is a candidate answer set for Π.

Alternatively, suppose that we choose b from rule (1). As a is not inferred from rule (1), rules (2) and (3) are not applicable; so rules (1), (2) and (3) together infer a potential answer set $I_2 = \{b\}$. As I_2 does not satisfy the constraint rule (3), it is not a candidate answer set for Π.

Consequently, $I_1 = \{a, b, c\}$ is a minimal candidate answer set and thus we expect it to be an answer set of Π.

However, I_1 is not an answer set under the GL-semantics because it is not a minimal model of Π ($J = \{b, c\}$ is a smaller model).

In addition to the above observations, there are more cases showing a similar behavior of the GL-semantics. For instance, if we replace in Example 1 rule (1) with $a \leftarrow b$, we may expect the same answer set $I = \{a, b\}$; however, this answer set is not obtained under the GL-semantics. Furthermore, the constraints $b \leftarrow \neg b$ and $c \leftarrow \neg c$ in Examples 1-3 may be replaced with proper rules and/or more involved constraints (a program of this characteristic will be considered in Example 5).

[4] A constraint that there is no answer set that does not contain b does not imply that there is an answer set that contains b; the program $\Pi = \{b \leftarrow \neg b\}$ is a typical example. Notably, $b \leftarrow \neg b$ cannot be replaced by b.

[5] Where $a \supset b$ in rule heads is rewritten to $\neg a \vee b$ if syntactically needed, e.g. in case of [45].

Our approach. An analysis of the above examples reveals that requiring an answer set to be a minimal model of rules of the GL-reduct Π^I may sometimes be too strong a condition, which may exclude some answer sets that would be reasonably acceptable. The strong condition in fact becomes apparent by a characterization of the answer sets of the GL-semantics in terms of a different program transformation (see Theorem 2) that incorporates nondeterministic inferences via the operator |. This motivates us to consider an alternative way to define answer sets for disjunctive programs, which is more permissive than the GL-semantics.

In line with the two Gelfond ASP principles (P1) and (P2) for constructing an answer set I of Π, we proceed as follows. For (P1), that I satisfies each rule of Π, we view | truly as a nondeterministic inference operator that returns from each rule $A_1 \mid \cdots \mid A_k \leftarrow Body$ of Π one A_i of the alternatives once $Body$ is satisfied. This would produce different candidate answer sets for Π (corresponding to different choices of the alternatives) satisfying the principle (P1). As for (P2), that we shall not believe anything we are not forced to believe, we admit as answer sets of Π only those candidate answer sets that are minimal among all of them. Such answer sets are not necessarily minimal models of Π.

Starting from this, we develop a generic approach for handling incomplete information expressed by disjunctive rule heads. In more detail, the main contributions of this paper are summarized as follows.

(1) We present a general answer set semantics for disjunctive programs, called *determining inference semantics* (*DI-semantics* for short), which interprets the operator | in rule heads differently from the classical connective $\vee$, and does not require that answer sets should be minimal models. Specifically, we introduce a head selection function *sel* to formalize the rule head operator |, i.e., for every interpretation I and every rule head $H_1 \mid \cdots \mid H_k$, $sel(H_1 \mid \cdots \mid H_k, I)$ nondeterministically selects one alternative H_i satisfied by I. Then we define answer sets as follows: (i) Given an interpretation I and a selection function *sel*, we transform a disjunctive program Π into a normal program Π^I_{sel}, called a *disjunctive program reduct*, such that for every rule $head(r) \leftarrow body(r)$ in Π, $sel(head(r), I) \leftarrow body(r)$ is in Π^I_{sel} if I satisfies $body(r)$; (ii) given a base answer set semantics $\mathcal{X}$ for normal programs, we define I to be a *candidate answer set* of Π w.r.t. $\mathcal{X}$ if I is an answer set of Π^I_{sel} under $\mathcal{X}$; and (iii) we define I to be an answer set of Π w.r.t. $\mathcal{X}$ if I is a minimal candidate answer set. Such answer sets are called *DI-answer sets*.

(2) By replacing the base semantics $\mathcal{X}$ in the above general semantics with the GL_{nlp}-semantics defined by Gelfond and Lifschitz [33], we induce a DI-semantics for simple disjunctive programs. We show that an answer set under the GL-semantics is an answer set under the DI-semantics, but not vice versa; the main reason behind is that the GL-semantics interprets the operator | in rule heads as the classical connective $\vee$ and further requires that answer sets must be minimal models; this may exclude some desired answer sets. To clearly see the essential difference of the DI-semantics from the GL-semantics, we also present a new characterization of the GL-semantics in terms of a disjunctive program reduct Π^I_{sel}. It is based on this characterization that we obtain a satisfactory solution to the open problem presented by Hitzler and Seda [36] about characterizing split normal derivatives of a simple disjunctive program Π.

(3) By replacing the base semantics $\mathcal{X}$ with the well-justified semantics defined by Shen et al. [65], we further induce a DI-semantics for general disjunctive programs consisting of rules of the form $H_1 \mid \cdots \mid H_k \leftarrow B$, where B and every H_i are arbitrary first-order formulas. This closes an important open task presented in [65] about extending the well-justified semantics from general normal programs with rules of the form $H_1 \leftarrow B$ to general disjunctive programs.

(4) In disjunctive programs, every rule head $H_1 \mid \cdots \mid H_k$ can be viewed as a set $\{H_1, \cdots, H_k\}$ of alternatives. Some other set related constructs are also available in the ASP literature, such as *choice constructs* [67,28,11] and *set introduction rules* [35]. It seems that choice constructs are most closely related to disjunctive rule heads because both of them are used in rule heads to express a set of alternatives. In this paper we clarify the difference between a disjunctive rule head and a choice construct. In particular, we use a generalization of the well-known strategic companies problem [10,40] as an example to show that because the information expressed by a disjunctive rule head $a_1 \mid \cdots \mid a_m$ is incomplete, we cannot use a choice construct $1\{a_1, \cdots, a_m\}u$ to replace the rule head, where the a_i's are ground atoms.

(5) Finally, we show that in the propositional case deciding whether a simple disjunctive program Π has some DI-answer set is NP-complete, and deciding whether a ground literal is true in some (resp. every) DI-answer set of Π is Σ^p_2-complete (resp. Π^p_2-complete). This is in contrast to the GL-semantics, where deciding whether a simple disjunctive program has GL-answer sets is Σ^p_2-complete [18]. For general disjunctive programs, the complexity of the DI-semantics increases by one level in the polynomial hierarchy to Σ^p_2-completeness for DI-answer set existence and to Σ^p_3-completeness and Π^p_3-completeness for brave and cautious reasoning, respectively. This means an increase in expressiveness for declarative problem solving, which even is not achieved by popular ASP constructs such as weak constraints (cf. [11]); an example of such a problem is a further generalization of the strategic companies problem. Furthermore, we discuss the complexity of computing some DI-answer set, which corresponds to computing minimal models of quantified Boolean formulas (QBFs).

Organization. The remainder of this article is organized as follows. In the next section, we introduce a first-order logic language and define general disjunctive programs. In Section 3, we present the new DI-inference answer set semantics for disjunctive programs under a generic base semantics, while in Sections 4 and 5 we consider the classes of simple and general disjunctive programs, respectively, under the GL-answer set semantics and the well-justified answer set semantics [65], respectively. We then compare in Section 6 the DI-semantics with choice constructs, and we study in Section 7 the computational complexity of the DI-semantics. In Section 8 we review related work, and in Section 9 we conclude with a summary and perspectives for future work.

In order not to distract from the flow of reading, proofs of theorems are in the Appendix.

2. Preliminaries

In this section, we first introduce a first-order logic language, and then define general disjunctive programs with first-order formulas.

2.1. A first-order logic language

We define a first-order logic language $\mathcal{L}_\Sigma$ with equality over a *signature* $\Sigma = (\mathcal{P}, \mathcal{F})$, where $\mathcal{P}$ and $\mathcal{F}$ are countable sets of *predicate* and *function* symbols, respectively; $\mathcal{C} \subseteq \mathcal{F}$ denotes the set of 0-ary function symbols, which are called *constants*. *Variables, terms, atoms* and *literals* are defined as usual. We denote variables with strings starting with x, y, z, X, Y or Z.

First-order formulas (briefly formulas) are constructed as usual from atoms using connectives $\neg$, $\wedge$, $\vee$, $\supset$, $\top$, $\bot$, $\exists$ and $\forall$, where $\top$ and $\bot$ are two 0-place logical connectives expressing $true$ and $false$, respectively.[6] Formulas are *closed* if they contain no free variables, i.e., each variable occurrence is in the scope of some quantifier. A *first-order theory* (or theory) is a set of closed formulas. Terms, atoms and formulas are *ground* if they have no variables. By $\mathcal{N}_\Sigma$ we denote the set of all ground terms of Σ, and by $\mathcal{H}_\Sigma$ the set of all ground atoms.

We consider *SNA interpretations*, i.e., interpretations which employ the well-known *standard names assumption* (SNA) [16, 52]. An SNA interpretation (or interpretation for short) I of $\mathcal{L}_\Sigma$ is a subset of $\mathcal{H}_\Sigma$ such that for any ground atom A, I *satisfies* A if $A \in I$, and I satisfies $\neg A$ if $A \notin I$. The notion of *satisfaction/models* of a formula/theory in I is defined as usual. A theory T is *consistent* or *satisfiable* if T has a model. T *entails* a closed formula F, denoted $T \models F$, if all models of T are models of F. Two theories T_1 and T_2 are *logically equivalent*, denoted $T_1 \equiv T_2$, if $T_1 \models T_2$ and $T_2 \models T_1$. Furthermore F is *true* (resp. *false*) in an interpretation I if I satisfies (resp. does not satisfy) F.

For an interpretation I, we let $I^- = \mathcal{H}_\Sigma \setminus I$ and $\neg I^- = \{\neg A \mid A \in I^-\}$.

2.2. General disjunctive programs

We define logic programs by extending the above first-order logic language $\mathcal{L}_\Sigma$ to include the two rule operators $\leftarrow$ and $|$.

Definition 1. A *general disjunctive program* (*disjunctive program or logic program* for short) is a finite set of *rules* of the form

$$H_1 \mid \cdots \mid H_k \leftarrow B \tag{3}$$

where $k > 0$,[7] and B and the H_i's are first-order formulas.

For convenience, for a rule r of form (3) we refer to B and $H_1 \mid \cdots \mid H_k$ as the body and head of r, denoted $body(r)$ and $head(r)$, respectively. We also refer to each H_i as a *head formula*. When $body(r)$ is empty, we drop the if-then rule operator $\leftarrow$. A *constraint* is a rule of the form $\bot \leftarrow B$, and a *fact* is of the form A, where A is a ground atom. Note that a rule $A \leftarrow \neg A$ amounts to a constraint $\bot \leftarrow \neg A$.

Remark 1. Note that we define logic programs using formulas in classical logic as the basic building blocks, as e.g. in many nonmonotonic logics such as Default Logic [59], Autoepistemic Logic [51], etc. Syntactically, general logic programs are first-order logic formulas extended with the two rule operators $\leftarrow$ and $|$. As in [65], default negation will be expressed by giving the connective $\neg$ a special meaning via knowledge minimization on answer sets; specifically, for every answer set I and every ground atom A we assume $\neg A$ to be true in I whenever possible by requiring I to be minimal, i.e., no $J \subset I$ is another answer set (see Definition 6).

As inference rule operators rather than logical connectives, $\leftarrow$ and $|$ do not occur in any formulas in rule bodies and heads.

A general disjunctive program is a *general normal program* (*normal program* for short) if $k = 1$; a *simple disjunctive program* if each H_i is an atom and B is a conjunction of literals, and a *simple normal program* if additionally $k = 1$. A *positive simple normal/disjunctive program is a simple normal/disjunctive program* without negative literals.

The *Herbrand universe* of a logic program Π, denoted $\mathcal{HU}_\Pi$, is the set of all ground terms constructed with constant and function symbols from Π, and the *Herbrand base* of Π, denoted $\mathcal{HB}_\Pi$, is the set of all ground atoms constructed with terms and predicate symbols from Π. Any subset of $\mathcal{HB}_\Pi$ is a *Herbrand interpretation* of Π.

Definition 2. A *propositional program* Π is a logic program which contains no variables, no function symbols except constants, and no equalities.

[6] For convenience, we sometimes write $G \supset H$ and $H \subset G$ exchangeably.

[7] When $k = 0$, the rule can be rewritten as $\bot \leftarrow B$.

2.2.1. Grounding

In a logic program Π, some rules may contain free variables. In ASP, these free variables will be instantiated by constants from a finite set, usually the set $\mathcal{C}_\Pi$ of constants occurring in Π.[8]

A *closed instance* of a rule is the rule with all free variables replaced by constants in $\mathcal{C}_\Pi$. The *grounding* of Π, denoted $ground(\Pi)$, is the set of all closed instances of all rules in Π. Since $\mathcal{C}_\Pi$ is finite, $ground(\Pi)$ is finite.

Note that each rule r of form (3) with the set S of free variables may also be viewed as a *globally* universally quantified rule $\forall S(r)$, 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 $ground(\Pi)$.

To sum up, a logic program Π is viewed as shorthand for $ground(\Pi)$, where each free variable in Π is viewed as shorthand for constants in $\mathcal{C}_\Pi$.

2.2.2. Satisfaction and models

We extend the satisfaction relation of $\mathcal{L}_\Sigma$ to logic programs in the following way. An interpretation I *satisfies* a rule head $H_1 \mid \cdots \mid H_k$ if it satisfies some H_i; 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 Π if I satisfies every rule $r \in ground(\Pi)$. Moreover, a model I of Π is *minimal* if Π has no model J that is a proper subset of I.

2.3. GL-semantics for simple disjunctive/normal programs

Let Π be a simple disjunctive program and I an interpretation. The *GL-reduct* of Π w.r.t. I, written as Π^I, is obtained from $ground(\Pi)$ by (1) removing all rules whose bodies contain some $\neg C_i$ with $C_i \in I$, and (2) removing from the remaining rules all $\neg C_i$. Note that every rule in the GL-reduct Π^I is of the form

$$A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m \tag{4}$$

where all A_i and B_j are ground atoms.

The *GL-semantics* defines I to be an answer set of Π (referred to as *GL-answer set*) if I is a minimal model of Π^I [34]. When Π is a simple normal program, the *GL_{nlp}-semantics* defines I to be an answer set of Π if I is the least model of Π^I [33]. For simple normal programs, the GL-semantics reduces to the GL_{nlp}-semantics.

Remark 2. Observe that if we replace $\mid$ with $\vee$ in rule (4) and let $\Pi^I_\vee$ be Π^I with all occurrences of $\mid$ replaced by $\vee$, then Π^I has the same minimal models as $\Pi^I_\vee$. Therefore, I is an answer set of Π under the GL-semantics if and only if I is a minimal model of Π^I if and only if I is a minimal model of $\Pi^I_\vee$. This means that in the GL-semantics the operator $\mid$ in rule heads amounts in essence to the connective $\vee$.

3. A general answer set semantics for disjunctive programs

In this section we present a new answer set semantics for disjunctive programs. We first introduce a rule head formula selection function *sel* to formalize the operator $\mid$, and with it transform a disjunctive program Π into a normal program Π^I_{sel} w.r.t. an interpretation I. We then define I to be a candidate answer set of Π w.r.t. an answer set semantics $\mathcal{X}$ for normal programs if I is an answer set of Π^I_{sel} under $\mathcal{X}$, and define I to be an answer set of Π if it is a minimal candidate answer set. The new semantics is general in the sense that it is applicable to extend any answer set semantics $\mathcal{X}$ for normal programs to disjunctive programs.

Under the constructive view of the operator $\mid$ as a nondeterministic inference operator, every rule head $\mathcal{H} = H_1 \mid \cdots \mid H_k$ in a disjunctive program can be viewed as a set $\{H_1, \cdots, H_k\}$ of alternatives. As these alternatives may have different variants (i.e., every H_i can be expressed as different yet logically equivalent formulas) and appear in different orders in rule heads, we introduce a notion of variant rule heads as follows.

Definition 3 *(Variant rule heads).* Let $\mathcal{H}_1 = E_1 \mid \cdots \mid E_k$ and $\mathcal{H}_2 = F_1 \mid \cdots \mid F_l$ be two rule heads, where the E_i's and F_j's are closed formulas. $\mathcal{H}_1$ and $\mathcal{H}_2$ are *variant rule heads* if for all i and j $(1 \le i \le k, 1 \le j \le l)$, $\mathcal{H}_1$ has a head formula E_i if and only if $\mathcal{H}_2$ has a head formula F_j with $E_i \equiv F_j$.

Intuitively, variant rule heads represent the same set of alternatives and should be treated the same. For instance, due to $a \wedge (c \vee \neg c) \equiv a$, the following three rule heads are variant rule heads:

$$a \mid b, \qquad b \mid a, \qquad a \wedge (c \vee \neg c) \mid b \mid a.$$

[8] Most of the results in this paper can extend to infinite instantiations as well, and while the extension is routine it is slightly more involving; in practice, the focus is on finitely groundable programs.

It is easy to show that if $\mathcal{H}_1$ and $\mathcal{H}_2$ are variant rule heads and $k > l$, then $\mathcal{H}_1$ must have at least two head formulas that are logically equivalent. Moreover, two rule heads in a simple disjunctive program are variant rule heads if and only if they have the same set of atoms.

We are now ready to present the following principal definition.

Definition 4 *(Head selection functions).* Let Π be a disjunctive program and $\mathcal{I}$ the collection of all interpretations. Let $\mathcal{HD}_\Pi$ be the set of all rule heads in $ground(\Pi)$, and $\mathcal{HF}_\Pi$ the set of all head formulas in $\mathcal{HD}_\Pi$. A *head selection* for Π is a function $sel : \mathcal{HD}_\Pi \times \mathcal{I} \mapsto \mathcal{HF}_\Pi \cup \{\bot\}$ such that for every interpretation $I \in \mathcal{I}$ and every rule $r \in ground(\Pi)$,

$$sel(head(r), I) = \begin{cases} F_i, & \text{if } head(r) \text{ has some head formula } F_i \text{ that is satisfied by } I \\ \bot, & \text{otherwise,} \end{cases}$$

and that for any variant rule heads $\mathcal{H}_1$ and $\mathcal{H}_2$ in $\mathcal{HD}_\Pi$, $sel(\mathcal{H}_1, I) \equiv sel(\mathcal{H}_2, I)$.[9]

A head selection function *sel* formalizes the operator | as a nondeterministic operator; i.e., for any interpretation I, $sel(F_1 \mid \cdots \mid F_k, I)$ returns from a rule head $F_1 \mid \cdots \mid F_k$ one of the alternatives F_i satisfied by I, or it returns $\bot$ if there is no F_i that is satisfied by I. For variant rule heads, it returns logically equivalent alternatives that are satisfied by I.

For example, let $\Pi = \{a \mid b,\ \ b \mid a \wedge (c \vee \neg c),\ \ b \mid c\}$ and $I = \{a, b\}$; then we have the following two head selections for Π on I:

$$(1)\ sel(a \mid b, I) = a, \quad sel(b \mid a \wedge (c \vee \neg c), I) = a \wedge (c \vee \neg c), \quad sel(b \mid c, I) = b;$$
$$(2)\ sel(a \mid b, I) = b, \quad sel(b \mid a \wedge (c \vee \neg c), I) = b, \quad sel(b \mid c, I) = b.$$

By applying a head selection function we can transform a disjunctive program into a normal program as follows.

Definition 5 *(Disjunctive program reducts).* Let Π be a disjunctive program, I an interpretation and *sel* a head selection function. The *reduct* of Π w.r.t. I and *sel* is

$$\Pi^I_{sel} = \{sel(head(r), I) \leftarrow body(r) \mid r \in ground(\Pi) \text{ such that } I \text{ satisfies } body(r)\} \tag{5}$$

A reduct Π^I_{sel} is a normal program; therefore we can apply any existing answer set semantics for normal programs to compute answer sets of Π^I_{sel}. Intuitively I is a candidate answer set of Π if I is an answer set of Π^I_{sel}, and I is an answer set of Π if I is minimal among all candidate answer sets. Formally we have the following definition.

Definition 6 *(General answer set semantics for disjunctive programs).* Let I be a model of a disjunctive program Π, and $\mathcal{X}$ be an answer set semantics for normal programs. Then I is an answer set of Π w.r.t. the base semantics $\mathcal{X}$ if (1) for some head selection function *sel*, I is an answer set of Π^I_{sel} under $\mathcal{X}$, and (2) Π has no model $J \subset I$ satisfying condition (1).

The above answer set semantics is well justified by its two conditions. First, as the rule head operator | is formalized by head selection functions, and that the application of a head selection function amounts to transforming a disjunctive program into a normal program by means of a disjunctive program reduct, condition (1) characterizes the set of candidate answer sets w.r.t. a given interpretation I. Then, according to the rationality principle for knowledge minimization, condition (2) admits as answer sets only those candidate answer sets that are minimal among all of them.

Due to the use of head selection functions, the above semantics interprets the disjunctive rule head operator | differently from the classical connective $\vee$. Let $\mathcal{H}_1 = E_1 \mid \cdots \mid E_k$ and $\mathcal{H}_2 = E_1 \vee \cdots \vee E_k$ be two rule heads and let I be an interpretation that satisfies $\mathcal{H}_2$. Then there may be up to k head selection functions for $\mathcal{H}_1$, each selecting one alternative E_i that is satisfied by I, which leads to at most k disjunctive program reducts; in contrast, there is only one head selection function for $\mathcal{H}_2$, i.e., $sel(\mathcal{H}_2, I) = \mathcal{H}_2$, which leads to only one disjunctive program reduct. Different reducts may lead to different candidate answer sets and thus disjunctive programs with rule heads like $\mathcal{H}_1$ are different from programs with rule heads like $\mathcal{H}_2$.

Moreover, the above semantics does not require that answer sets should be minimal models; it only requires answer sets to be minimal among all candidate answer sets.

In order to stress the intuition that candidate answer sets are nondeterministically determined by means of head selection functions by applying rules $H_1 \mid \cdots \mid H_k \leftarrow Body$ such that when $Body$ is satisfied we infer one alternative H_i from the head, we refer to the above answer set semantics as a *determining inference semantics* for disjunctive programs, abbreviated

[9] Using logical (classical) equivalence $\equiv$ to constrain the selection function is owing to the fact that we consider two-valued interpretations in our development. Other and more restrictive notions of equivalence are conceivable, e.g. if 3-valued interpretations were considered as in partial stable model semantics for defining answer sets [58].

as *DI-semantics*. Answer sets of the DI-semantics are then referred to as *DI-answer sets*, and models satisfying condition (1) of Definition 6 as *candidate DI-answer sets*.

In the following two sections, we illustrate some distinct properties of the DI-semantics by choosing the base answer set semantics $\mathcal{X}$ to be the GL_{nlp}-semantics for simple normal programs [33] and the well-justified answer set semantics for general normal programs [65], respectively.

4. DI-semantics for simple disjunctive programs

In Definition 6, by replacing the base semantics $\mathcal{X}$ with the GL_{nlp}-semantics, we induce a DI-answer set semantics for simple disjunctive programs.

Definition 7 *(DI-semantics for simple disjunctive programs).* Let I be a model of a simple disjunctive program Π. Then I is a *DI-answer set* of Π if (1) for some head selection function *sel*, I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics, and (2) Π has no model $J \subset I$ satisfying condition (1).

Example 4. Consider again the simple disjunctive program Π in Example 1, where we showed that $I = \{a, b\}$ is a desired answer set of Π, but due to the fact that the GL-semantics interprets the operator | in rule heads essentially as the connective $\vee$, it is not a GL-answer set. We illustrate below that I is a DI-answer set.

Consider a head selection function *sel*, where $sel(a \mid b, I) = b$. Applying this selection leads to the following reduct:

$$\Pi^I_{sel}: \quad a \quad (1)$$
$$\qquad\quad b \quad (2)$$

I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics, so condition (1) of Definition 7 is satisfied and thus I is a candidate answer set. As I is a minimal model of Π, condition (2) is also satisfied; hence I is a minimal candidate answer set. Therefore I is a DI-answer set of Π, which agrees with our expectation.

For another example, we consider a generalization of the well-known Strategic Companies problem [10,40], which is popular for ASP benchmark competitions.

Example 5 *(Generalized Strategic Companies problem (GSC)).* Suppose a holding has companies $C = \{c_1, \ldots, c_m\}$, and it produces goods $G = \{g_1, ..., g_n\}$, where each company $c_i \in C$ produces some goods $G_i \subseteq G$. The holding wants to sell some of its companies subject to the following conditions: all products should be still in the portfolio and companies c_i for which a strategy rationale, expressed by justifications $\sigma^i_1, \ldots, \sigma^i_{k_i}$ that are conjunctions $\sigma^i_j = \ell^i_1 \wedge \cdots \wedge \ell^i_{l_i}$ of literals on C, holds true are not sold.

A set of companies $C' \subseteq C$ constitutes a *strategic set* if (1) the companies in C' produce all the goods in G, (2) in addition, only the companies c_i such that some justification σ^i_j holds true relative to C' are in C' as well, and (3) C' is subset-minimal w.r.t. conditions (1) and (2). The classic Strategic Companies problem results if each justification σ^i_j contains only positive literals, intuitively expressing that the companies occurring in it have the power to jointly control c_i. To illustrate the generalized problem, consider companies $C = \{c_1, c_2, c_3\}$ and goods $G = \{g_1, g_2\}$, where g_1 can be produced either by c_1 or c_2, and g_2 produced by c_1 or c_3. Suppose that the strategy conditions are $\sigma^1_1 = c_2 \wedge c_3$, $\sigma^2_1 = c_3$, and $\sigma^3_1 = c_1 \wedge \neg c_2$. We may readily express this as a simple disjunctive program as follows:

$$\Pi: \quad g_1 \quad (1)$$
$$\qquad g_2 \quad (2)$$
$$\qquad c_1 \mid c_2 \leftarrow g_1 \qquad // \; g_1 \text{ is produced either by } c_1 \text{ or } c_2 \quad (3)$$
$$\qquad c_1 \mid c_3 \leftarrow g_2 \qquad // \; g_2 \text{ is produced either by } c_1 \text{ or } c_3 \quad (4)$$
$$\qquad c_1 \leftarrow c_2 \wedge c_3 \qquad // \text{ strategy condition } \sigma^1_1 = c_2 \wedge c_3 \quad (5)$$
$$\qquad c_2 \leftarrow c_3 \qquad // \text{ strategy condition } \sigma^2_1 = c_3 \quad (6)$$
$$\qquad c_3 \leftarrow c_1 \wedge \neg c_2 \qquad // \text{ strategy condition } \sigma^3_1 = c_1 \wedge \neg c_2 \quad (7)$$

We illustrate that Π has a DI-answer set $I = \{g_1, g_2, c_1, c_2\}$, which corresponds to the single strategic set for this scenario. Consider a head selection function *sel*, where $sel(c_1 \mid c_2, I) = c_2$ and $sel(c_1 \mid c_3, I) = c_1$. Applying this selection leads to the following reduct:

$$\Pi^I_{sel}: \quad g_1 \quad (1)$$
$$\qquad\quad g_2 \quad (2)$$
$$\qquad\quad c_2 \leftarrow g_1 \quad (3)$$
$$\qquad\quad c_1 \leftarrow g_2 \quad (4)$$

I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics, so condition (1) of Definition 7 is satisfied, making I a candidate answer set. As I is a minimal model of Π, it is a minimal candidate answer set. Hence I is a DI-answer set of Π, as we expected.

Note that under the GL-semantics, this simple disjunctive program has no answer set.[10]

According to the rationality principle for knowledge minimization it is required that a DI-answer set must be minimal among all candidate DI-answer sets; so a simple disjunctive program Π will never have two DI-answer sets I and J with $J \subset I$. This does not mean that a DI-answer set of Π must be a minimal model of Π, as shown in the following example.

Example 6. Consider the simple disjunctive program Π in Example 3. For $I_1 = \{a, b, c\}$, consider a head selection function sel with $sel(a \mid b, I_1) = a$. Applying this selection yields the following reduct:

$$\Pi^{I_1}_{sel}: \quad a \tag{1}$$
$$b \leftarrow a \tag{2}$$
$$c \leftarrow a \tag{3}$$
$$c \leftarrow \neg c \tag{4}$$

I_1 is an answer set of $\Pi^{I_1}_{sel}$ under the GL_{nlp}-semantics, so condition (1) of Definition 7 is satisfied and thus I_1 is a candidate DI-answer set of Π.

Now consider $I_2 = \{b, c\}$ that is the only model of Π that is smaller than I_1. There is only one selection function sel with $sel(a \mid b, I_2) = b$. By applying this selection we obtain the following reduct:

$$\Pi^{I_2}_{sel}: \quad b \tag{1}$$
$$b \leftarrow a \tag{2}$$
$$c \leftarrow a \tag{3}$$
$$c \leftarrow \neg c \tag{4}$$

I_2 is not an answer set of $\Pi^{I_2}_{sel}$ under the GL_{nlp}-semantics and thus not a candidate DI-answer set of Π.

Therefore, I_1 also satisfies condition (2) of Definition 7; hence it is a DI-answer set of Π. Note that this DI-answer set is not a minimal model of Π.

We see from the above examples that a DI-answer set is not necessarily a GL-answer set. However, for simple normal programs and positive simple disjunctive programs, the DI-semantics agrees with the GL-semantics.

Theorem 1. *Let Π be a simple normal program or a positive simple disjunctive program. Then an interpretation I is a DI-answer set of Π if and only if I is a GL-answer set of Π.*

Example 7 *(Example 5 continued).* In the case where a GSC instance is a classic Strategic Companies instance, i.e., each justification σ^i_j is a conjunction of positive literals, the rules for expressing the strategic sets similar as in Example 5 would be positive. For example, this is the case if the justification σ^3_1 would be $\sigma^3_1 = c_1 \wedge c_2$. Then, by replacing rule (7) in Example 5 with $c_3 \leftarrow c_1 \wedge c_2$ we obtain a positive simple disjunctive program Π' that amounts to the usual (propositional) Strategic Companies encoding. The DI-answer sets of Π' are by Theorem 1 the GL-answer sets of Π', which coincide with the minimal models of Π'. It is easy to check that Π' has the single DI-answer set $I = \{g_1, g_2, c_1\}$, which is the single GL-answer set and the single minimal model of Π'.

It is particularly interesting to observe that the GL-semantics can also be characterized using the disjunctive program reduct Π^I_{sel} of Definition 5 simply by requiring that for *every* (instead of *some*) head selection function sel, I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics. *This reveals the essential difference of the DI-semantics from the GL-semantics.*

Theorem 2. *Let I be a model of a simple disjunctive program Π. Then I is a GL-answer set of Π if and only if for every head selection function sel, I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics.*

Let us illustrate with the following example.

Example 8. In Example 4, for the model $I = \{a, b\}$ of Π if we apply another head selection function sel with $sel(a \mid b, I) = a$, we obtain the following reduct:

[10] Expressing this problem under the GL-semantics using the usual guess and check approach is more involving (see Appendix A).

$$\Pi^{I}_{sel}: \quad a \quad (1)$$
$$a \quad (2)$$

I is not an answer set of Π^{I}_{sel} under the GL$_{nlp}$-semantics, so by Theorem 2 I is not a GL-answer set of Π.

Similarly, in Example 6 for the model $I_1 = \{a, b, c\}$, applying another head selection function sel with $sel(a \mid b, I_1) = b$ would lead to the following reduct:

$$\Pi^{I_1}_{sel}: \quad b \quad (1)$$
$$b \leftarrow a \quad (2)$$
$$c \leftarrow a \quad (3)$$
$$c \leftarrow \neg c \quad (4)$$

I_1 is not an answer set of $\Pi^{I_1}_{sel}$ under the GL$_{nlp}$-semantics, and thus by Theorem 2 is not a GL-answer set of Π.

As shown in [34], every GL-answer set of a simple disjunctive program Π is a minimal model of Π. Then the following corollary is immediate from Theorem 2, showing that every GL-answer set is a DI-answer set.

Corollary 1. *Let Π be a simple disjunctive program. If an interpretation I is a GL-answer set of Π, then I is also a DI-answer set.*

Next we discuss the property of the DI-semantics in the case that new constraints of the form $\bot \leftarrow body(r)$ are added, where $body(r)$ denotes $B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n$.

4.1. Adding constraints

The GL-semantics has the property that adding a constraint $\bot \leftarrow body(r)$ to a simple disjunctive program Π may rule out some answer sets of Π, but would never add new answer sets [45]. Let $\Pi' = \Pi \cup \{\bot \leftarrow body(r)\}$, and let $S = S_1 \cup S_2$ be the set of GL-answer sets of Π, where S_1 (resp. S_2) is the set of GL-answer sets satisfying $body(r)$ (resp. not satisfying $body(r)$). Then, it is trivial to show that S_2 is the set of GL-answer sets of Π'. Hence, all GL-answer sets of Π' are GL-answer sets of Π.

The above property does not hold for the DI-semantics. In addition to S_1 being ruled out and S_2 being kept, adding the constraint $\bot \leftarrow body(r)$ to Π may introduce new DI-answer sets. The reason is simple. Suppose Π has two candidate DI-answer sets I and J with $I \subset J$ and $I \in S_1$. Note that J is not a DI-answer set of Π because it is not a minimal candidate. When I is ruled out because of the addition of the constraint $\bot \leftarrow body(r)$, J may then become minimal among the remaining candidate DI-answer sets and thus be added as a DI-answer set of Π'.

We use the following example to illustrate how answer sets change with the addition of a fact or a constraint.

Example 9. Consider the following three simple disjunctive programs:

$$\Pi_1 = \{a \mid b\},\ \Pi_2 = \{a \mid b,\ a\},\ \Pi_3 = \{a \mid b,\ a,\ b \leftarrow \neg b\}$$

Note that $b \leftarrow \neg b$ represents the same constraint as $\bot \leftarrow \neg b$.

For the GL-semantics, Π_1 has two GL-answer sets: $I = \{a\}$ and $J = \{b\}$. As in the GL-semantics the rule head operator $\mid$ amounts to the logical connective $\vee$, the addition of the fact a makes $a \mid b$ redundant, so Π_2 has a single GL-answer set $I = \{a\}$. This single GL-answer set is ruled out by the addition of the constraint $b \leftarrow \neg b$, hence Π_3 has no GL-answer set.

For the DI-semantics, Π_1 has two candidate DI-answer sets: $I = \{a\}$ and $J = \{b\}$, which are also DI-answer sets. The addition of the fact a updates the two candidate DI-answer sets to $I = \{a\}$ and $J = \{a, b\}$, so Π_2 has a single DI-answer set $I = \{a\}$. Note that J is not a DI-answer set of Π_2 because it is not a minimal candidate. The addition of the constraint $b \leftarrow \neg b$ rules out I, so $J = \{a, b\}$ becomes the only candidate DI-answer set of Π_3 and thus the only DI-answer set of Π_3.

The property of the DI-semantics for constraints is characterized as follows.

Theorem 3. *Let Π and $\Pi' = \Pi \cup \{\bot \leftarrow body(r)\}$ be two simple disjunctive programs, and let I be a DI-answer set of Π' but not a DI-answer set of Π. Then I is a non-minimal candidate DI-answer set of Π.*

As a result, a constraint $\bot \leftarrow body(r)$ acts as a *filter*, whose addition to Π eliminates all of those GL-answer sets, resp. candidate DI-answer sets, of Π that satisfy $body(r)$. The latter would let some non-minimal candidate DI-answer sets of Π become minimal among the remaining candidate DI-answer sets, thus yielding new DI-answer sets for $\Pi' = \Pi \cup \{\bot \leftarrow body(r)\}$. As these new DI-answer sets are non-minimal candidate DI-answer sets of Π, they are not minimal models of Π and therefore not GL-answer sets of Π and Π'.

We observe, however, that adding to Π a negation-free constraint of the form $\bot \leftarrow B_1 \wedge \cdots \wedge B_m$ does not lead to new DI-answer sets. The reason is that if a non-minimal candidate DI-answer set J of Π becomes a DI-answer set of $\Pi' = \Pi \cup \{\bot \leftarrow B_1 \wedge \cdots \wedge B_m\}$, then some DI-answer set $I \subset J$ of Π must have been eliminated due to that I satisfies the body $B_1 \wedge \cdots \wedge B_m$ of the constraint; however, in this case J must also have been eliminated because it also satisfies $B_1 \wedge \cdots \wedge B_m$, which leads to a contradiction.

To conclude this section, we summarize that the DI-semantics is a relaxation of the GL-semantics in the following two aspects: (1) Under the GL-semantics, the rule head operator | amounts to the classical connective $\vee$, while under the DI-semantics, | is viewed as a nondeterministic inference operator that returns from each rule $A_1 \mid \cdots \mid A_k \leftarrow Body$ of Π one A_i of the alternatives once $Body$ is satisfied; such nondeterministic inference allows us to generate different candidate answer sets. (2) Under the GL-semantics, the rationality principle for knowledge minimization requires that every GL-answer set of Π must be a minimal model of Π, while under the DI-semantics, the rationality principle requires that every DI-answer set must be minimal among all candidate DI-answer sets; such a DI-answer set is not necessarily a minimal model of Π. As a result, the DI-semantics admits not only all GL-answer sets, but also those desired answer sets that are missed under the GL-semantics, as illustrated in the examples of this paper.

Finally, we remark that like the GL-semantics, the DI-semantics can be readily adapted to an *extended simple disjunctive program* $\Pi^{\sim}$ with *strong negation literals* of the form $\sim A$, which consists of rules of the form

$$L_1 \mid \cdots \mid L_k \leftarrow L_{k+1} \wedge \cdots \wedge L_m \wedge \neg L_{m+1} \wedge \cdots \wedge \neg L_n \tag{6}$$

where $n \geq m \geq k \geq 0$, and each L_i is either an atom A or a strong negation literal $\sim A$.

Let $Lit = \{L \mid L \text{ is } A \text{ or } \sim A, \text{ where } A \text{ is a ground atom in } \mathcal{H}_\Sigma\}$. A and $\sim A$ are called *complementary literals*. An *extended interpretation* with strong negation, denoted $I^{\sim}$, is a subset of Lit such that for any ground literal $L \in Lit$, $I^{\sim}$ *satisfies* L if $L \in I^{\sim}$, and $I^{\sim}$ satisfies $\neg L$ if $L \notin I$.[11] $I^{\sim}$ is *consistent* if it has no complementary literals. A consistent interpretation $I^{\sim}$ is a model of $\Pi^{\sim}$ if it satisfies all rules of $ground(\Pi^{\sim})$; $I^{\sim}$ is a minimal model if it is subset-minimal among all models of $\Pi^{\sim}$.

Then, as in [34], the GL-semantics for a simple disjunctive program Π can be adapted to the *GL$^{\sim}$-semantics* for an extended simple disjunctive program $\Pi^{\sim}$ by applying extended interpretations $I^{\sim}$ as follows:

GL$^{\sim}$-semantics for extended programs with strong negation. A consistent interpretation $I^{\sim}$ is an answer set of $\Pi^{\sim}$ if it is a minimal model of the GL-reduct $(\Pi^{\sim})^{I^{\sim}}$.

Similarly, we can readily adapt the DI-semantics (Definition 7) to the *DI$^{\sim}$-semantics* by adopting extended interpretations as follows:

DI$^{\sim}$-semantics for extended programs with strong negation. A model $I^{\sim}$ of an extended simple disjunctive program $\Pi^{\sim}$ is an answer set of $\Pi^{\sim}$ if (1) for some head selection function sel, $I^{\sim}$ is an answer set of $(\Pi^{\sim})^{I^{\sim}}_{sel}$ under the GL$^{\sim}$-semantics, and (2) $\Pi^{\sim}$ has no model $J^{\sim} \subset I^{\sim}$ satisfying condition (1).

Due to the fact that the GL$^{\sim}$-semantics requires that an answer set should be a minimal model of rules of the GL-reduct, the two observations on the GL-semantics described in the Introduction apply to the GL$^{\sim}$-semantics as well. Specifically, in the GL$^{\sim}$-semantics the operator | in rule heads amounts to the connective $\vee$ because a consistent interpretation $I^{\sim}$ is an answer set of $\Pi^{\sim}$ if and only if $I^{\sim}$ is a minimal model of the GL-reduct $(\Pi^{\sim})^{I^{\sim}}$ if and only if $I^{\sim}$ is a minimal model of $(\Pi^{\sim})^{I^{\sim}}_{\vee}$, where $(\Pi^{\sim})^{I^{\sim}}_{\vee}$ is $(\Pi^{\sim})^{I^{\sim}}$ with all occurrences of | replaced by $\vee$. Moreover, answer sets of $\Pi^{\sim}$ in the GL$^{\sim}$-semantics must be minimal models of $\Pi^{\sim}$. As a result, like the GL-semantics, the GL$^{\sim}$-semantics may exclude some desired answer sets.

5. DI-semantics for general disjunctive programs

General normal programs consist of rules of the form $H \leftarrow B$, where H and B are first-order formulas. To solve the problem of circular justifications with the answer set semantics for general normal programs such as those defined in [55,69,6,24,27], Shen et al. [65] presented the *well-justified semantics* whose answer sets are well justified by having a level mapping and thus are free of circular justifications. It is analogous to the GL_{nlp}-semantics for simple normal programs whose answer sets are well justified by having a level mapping [25]. Shen et al. [65] further presented an open issue to extend the well-justified semantics to general disjunctive programs. In this section we give a solution to this open task and present a DI-semantics for general disjunctive programs by replacing the base semantics $\mathcal{X}$ in Definition 6 with the well-justified semantics.

[11] The notion of extended interpretations with strong negation makes $\sim$ essentially different from $\neg$. Consider an extended interpretation $I^{\sim} = \emptyset$ and a ground atom A. As $I^{\sim}$ does not contain $\sim A$, it does not satisfy $\sim A$; however, as $I^{\sim}$ does not contain A, it satisfies $\neg A$. This shows that an extended interpretation that satisfies $\neg A$ may not satisfy $\sim A$. However, an extended interpretation that satisfies $\sim A$ must satisfy $\neg A$, unless the interpretation is inconsistent (i.e., it contains complementary literals). In this sense, $\sim A$ is stronger than $\neg A$. Note that $A \vee \neg A$ is a tautology, i.e., every interpretation satisfies $A \vee \neg A$; in contrast, $A \vee \sim A$ is not satisfied by some interpretations like $I^{\sim} = \emptyset$ and thus it is not a tautology.

The well-justified semantics is based on the one-step provability operator $T_\Pi(O, N)$, which is an extension of the van Emden-Kowalski one-step provability operator [71] from positive simple normal to general normal programs.

Definition 8 *([65]).* Let Π be a general normal program, and let O and N be two first-order theories. Define

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

Informally, $T_\Pi(O, N)$ collects all heads of rules in $ground(\Pi)$ whose bodies are entailed by $O \cup N$. When the parameter N is fixed, the entailment relation $\models$ is monotone in O, so $T_\Pi(O, N)$ is monotone w.r.t. O, i.e., for any first-order theories $O_1 \subseteq O_2$, we have $T_\Pi(O_1, N) \subseteq T_\Pi(O_2, N)$. As moreover $T_\Pi(O, N)$ is finitary,[12] it is immediate that the inference sequence $\langle T^i_\Pi(\emptyset, N)\rangle_{i=0}^{\infty}$, where $T^0_\Pi(\emptyset, N) = \emptyset$ and for $i \geq 0$ $T^{i+1}_\Pi(\emptyset, N) = T_\Pi(T^i_\Pi(\emptyset, N), N)$, will converge to a least fixpoint, denoted $lfp(T_\Pi(\emptyset, N))$.

The *well-justified semantics* or *WJ-semantics* for short is defined in terms of the least fixpoint $lfp(T_\Pi(\emptyset, \neg I^-))$ w.r.t. an interpretation I.

Definition 9 *([65]).* Let I be a model of a general normal program Π. Then I is a *WJ-answer set* of Π if $lfp(T_\Pi(\emptyset, \neg I^-)) \cup \neg I^- \models A$ for every $A \in I$.

Then, by replacing the base semantics $\mathcal{X}$ in Definition 6 with the WJ-semantics we induce a DI-answer set semantics for general disjunctive programs.

Definition 10 *(DI-semantics for general disjunctive programs).* Let I be a model of a general disjunctive program Π. Then I is a *DI-answer set* of Π if (1) for some head selection function *sel*, I is a WJ-answer set of Π^I_{sel}, and (2) Π has no model $J \subset I$ satisfying condition (1).

Intuitively, a DI-answer set of a general disjunctive program is a model that is minimal among all of the models that can be nondeterministically (by means of a head selection function) inferred by iteratively applying rules via a bottom up fixpoint sequence.

It was proved in [65] that every WJ-answer set of a general normal program Π is a minimal model of Π. Then the following result is immediate from Definition 10.

Corollary 2. *For a general normal program Π, an interpretation I is a DI-answer set of Π if and only if I is a WJ-answer set of Π.*

As shown in [65], for simple normal programs the WJ-semantics coincides with the GL_{nlp}-semantics. Then, for a simple disjunctive program Π, Π^I_{sel} is a simple normal program and thus I is a WJ-answer set of Π^I_{sel} if and only if I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics. Therefore, the following result is immediate.

Corollary 3. *For a simple disjunctive program Π, an interpretation I is a DI-answer set under Definition 10 if and only if I is a DI-answer set under Definition 7.*

Next we use two examples to illustrate the DI-semantics for general disjunctive programs.

Example 10. Consider again the general disjunctive program Π in Example 2, where $I = \{a, b\}$ is the only model of Π. Consider a head selection function *sel* with $sel(a \supset b \mid b \supset a, I) = a \supset b$. Applying this selection yields the following reduct:

$$\Pi^I_{sel}: \quad a \supset b \tag{1}$$
$$a \tag{2}$$

The least fixpoint $lfp(T_{\Pi^I_{sel}}(\emptyset, \neg I^-)) = \{a \supset b, a\}$ is obtained from the following inference sequence:

$$T^0_{\Pi^I_{sel}}(\emptyset, \neg I^-) = \emptyset,$$
$$T^1_{\Pi^I_{sel}}(\emptyset, \neg I^-) = T_{\Pi^I_{sel}}(T^0_{\Pi^I_{sel}}(\emptyset, \neg I^-), \neg I^-) = \{a \supset b, a\} \text{ (by rules (1) and (2)).}$$

As $lfp(T_{\Pi^I_{sel}}(\emptyset, \neg I^-)) \cup \neg I^-$ entails every atom in I, I is a WJ-answer set of Π^I_{sel}, so condition (1) of Definition 10 is satisfied and thus I is a candidate DI-answer set. As I is a minimal model of Π, condition (2) of Definition 10 is also satisfied, i.e., I is a minimal candidate DI-answer set. Hence I is a DI-answer set of Π, as we expected.

[12] I.e., whenever $\alpha \in T_\Pi(O, N)$, then there is some finite $O' \subseteq O$ such that $\alpha \in T_\Pi(O', N)$.

However, in this example if we replace the operator | with the connective $\vee$ in Π, for $I = \{a, b\}$ we would have a unique reduct as follows, where for every rule r in Π there is only one head selection function sel on I, where $sel(head(r), I) = head(r)$:

$$\Pi^I_{sel}: \quad a \supset b \vee b \supset a \quad (1)$$
$$a \quad (2)$$

The least fixpoint is $lfp(T_{\Pi^I_{sel}}(\emptyset, \neg I^-)) = \{a \supset b \vee b \supset a, a\}$ and it does not entail b in I (note that $a \supset b \vee b \supset a$ is a tautology). Condition (1) of Definition 10 is violated and thus I is not a DI-answer set of Π.

The following interesting example further illustrates that the DI-semantics properly differentiates the two rule operators $\leftarrow$ and | from the two logical connectives $\supset$ and $\vee$.

Example 11. Consider the following simple disjunctive program.

$$\Pi_1: \quad p \mid q \quad (1)$$
$$p \leftarrow q \quad (2)$$
$$q \leftarrow p \quad (3)$$

In answer set construction, rule (1) means to infer p or q; whichever we choose, applying the three rules of Π_1 always infers $I = \{p, q\}$. Note that I is a minimal model of Π_1. We next show that I is a DI-answer set of Π_1. Consider a head selection function with $sel(p \mid q, I) = p$. We have the following reduct:

$$\Pi^I_{1_{sel}}: \quad p \quad (1)$$
$$p \leftarrow q \quad (2)$$
$$q \leftarrow p \quad (3)$$

$I = \{p, q\}$ is an answer set of $\Pi^I_{1_{sel}}$ under the WJ-semantics, where the least fixpoint is $lfp(T_{\Pi^I_{1_{sel}}}(\emptyset, \neg I^-)) = \{p, q\}$, so it is a candidate DI-answer set. As I is a minimal model of Π_1, it is a DI-answer set of Π_1.

Next, let's consider the following variant of Π_1, where the rule head operator | is replaced by the logical connective $\vee$.

$$\Pi_2: \quad p \vee q \quad (1)$$
$$p \leftarrow q \quad (2)$$
$$q \leftarrow p \quad (3)$$

Rule (1) says that the logical formula $p \vee q$ is true; as neither of the bodies of rules (2) and (3) is entailed by $p \vee q$, applying the three rules only infers $p \vee q$. As neither p nor q is entailed by $p \vee q$, $I = \{p, q\}$ should not be an answer set of Π_2.

We next show that $I = \{p, q\}$ is not a DI- answer set of Π_2. For every rule r in Π_2 there is only one head selection function sel on I, where $sel(head(r), I) = head(r)$. We have the following reduct:

$$\Pi^I_{2_{sel}}: \quad p \vee q \quad (1)$$
$$p \leftarrow q \quad (2)$$
$$q \leftarrow p \quad (3)$$

$I = \{p, q\}$ is not an answer set of $\Pi^I_{2_{sel}}$ under the WJ-semantics, where the least fixpoint is $lfp(T_{\Pi^I_{2_{sel}}}(\emptyset, \neg I^-)) = \{p \vee q\}$. Thus I is not a DI-answer set of Π_2.

Finally, consider the following variant of Π_2, where the if-then rule operator $\leftarrow$ is replaced by the logical implication connective $\supset$:

$$\Pi_3: \quad p \vee q \quad (1)$$
$$q \supset p \quad (2)$$
$$p \supset q \quad (3)$$

As Π_3 is a first-order theory, applying its rules infers Π_3 itself. As both p and q are entailed by Π_3, $I = \{p, q\}$ is expected to be an answer set of Π_3.

We next show that $I = \{p, q\}$ is a DI- answer set of Π_3. There is only one head selection function sel on I, thus leading to the following reduct:

$$\Pi^I_{3_{sel}}: \quad p \vee q \tag{1}$$
$$q \supset p \tag{2}$$
$$p \supset q \tag{3}$$

$I = \{p, q\}$ is an answer set of $\Pi^I_{3_{sel}}$ under the WJ-semantics, where the least fixpoint is $lfp(T_{\Pi^I_{3_{sel}}}(\emptyset, \neg I^-)) = \Pi^I_{3_{sel}}$, so it is a candidate DI-answer set. As I is a minimal model of Π_3, it is a DI-answer set of Π_3.

6. Difference between disjunctive rule heads and choice constructs

Like disjunctive rule heads, *choice constructs* [67,28,11] are also used to express a set of alternatives. However, they are essentially different. In this section, we clarify the difference between a disjunctive rule head $a_1 \mid \cdots \mid a_m$ and a choice construct of the form $u_1\{a_1, \cdots, a_m\}u_2$, where $m > 0$, $0 \leq u_1 \leq u_2 \leq m$, and the a_i's are ground atoms.

Let $\alpha = \{a_1, \cdots, a_m\}$ and let $\beta = \{J_1, \cdots, J_n\}$ be the collection of all subsets of α consisting of at least u_1 and no more than u_2 atoms. The choice construct $u_1\{a_1, \cdots, a_m\}u_2$ says that every subset $J_i \in \beta$ of α can be chosen as an answer.

For a logic program Π, let $AS(\Pi)$ denote the set of answer sets of Π. Let Π' be Π extended with a choice construct $u_1\{a_1, \cdots, a_m\}u_2$. Then the set of answer sets of Π' is

$$AS(\Pi') = \bigcup_{J_i \in \beta} AS(\Pi \cup \{a \mid a \in J_i\} \cup \{\neg b \mid b \in (\alpha \setminus J_i)\})$$

As an example, let $\Pi = \{b\}$ and $\Pi' = \Pi \cup \{1\{a, b\}2\}$. Then

$$AS(\Pi') = AS(\Pi \cup \{a, \neg b\}) \cup AS(\Pi \cup \{\neg a, b\}) \cup AS(\Pi \cup \{a, b\})$$

$\Pi \cup \{a, \neg b\}$ has no model and thus no answer set, $\Pi \cup \{\neg a, b\}$ has a single answer set $\{b\}$, and $\Pi \cup \{a, b\}$ has a single answer set $\{a, b\}$. Therefore, Π' has in total two answer sets, $\{b\}$ and $\{a, b\}$.

A disjunctive rule head $a_1 \mid \cdots \mid a_m$ infers one atom a_i from α and differs essentially from a choice construct $1\{a_1, \cdots, a_m\}u$.

When $u = 1$, the choice construct $1\{a_1, \cdots, a_m\}1$ in a logic program Π enforces every answer set of Π to contain exactly one a_i from α. In contrast, though $a_1 \mid \cdots \mid a_m$ infers only one a_i from α, it allows a DI-answer set to contain other atoms $a_j \in \alpha$, which are inferred by other rules in a disjunctive program.

When $u > 1$, the choice construct $1\{a_1, \cdots, a_m\}u$ allows for answer sets I and J with $I \subset J$. This will not happen with $a_1 \mid \cdots \mid a_m$ for DI-answer sets.

Next we use an example to show that because the information expressed by a disjunctive rule head $a_1 \mid \cdots \mid a_m$ is incomplete, we cannot guess a choice construct $1\{a_1, \cdots, a_m\}u$ and use it to replace the rule head.

Example 12. In Example 5, we have three companies $C = \{c_1, c_2, c_3\}$ and two goods $G = \{g_1, g_2\}$, where g_1 can be produced either by c_1 or c_2, and g_2 produced by c_1 or c_3. These conditions were modeled in the disjunctive program Π in Example 5 by the following four rules:

$$g_1 \tag{1}$$
$$g_2 \tag{2}$$
$$c_1 \mid c_2 \leftarrow g_1 \tag{3}$$
$$c_1 \mid c_3 \leftarrow g_2 \tag{4}$$

Because the information available is insufficient to tell whether the two goods g_1 and g_2 are produced by c_1 or c_2 or c_3, we can only choose among these alternatives c_i's nondeterministically; thus we cannot guess in advance two choice constructs and use them to replace the heads of rules (3) and (4). For example, if we used choice constructs $1\{c_1, c_2\}1$ and $1\{c_1, c_3\}1$ to replace the heads of rules (3) and (4) of Π in Example 5 respectively, Π would have no answer set/strategic set; if we used $1\{c_1, c_2\}2$ and $1\{c_1, c_3\}2$ to replace the two rule heads, Π would have two answer sets $I = \{g_1, g_2, c_1, c_2\}$ and $J = \{g_1, g_2, c_1, c_2, c_3\}$, which however violates the minimality condition of a strategic set.

7. Computational complexity

When all rules of a general normal program have an empty body, this program amounts to a first-order theory. As it is undecidable to determine whether an arbitrary first-order theory is satisfiable, it is undecidable to determine whether a general disjunctive program has a DI-answer set w.r.t. any base answer set semantics for general normal programs. Therefore, we address in this paper the computational complexity of propositional logic programs under Herbrand interpretations, and we focus on the DI-semantics with the well-justified semantics [65] as the base semantics and refer to it as *DI-WJ answer set semantics*; we briefly comment on other base semantics in Section 7.1.

Table 1
Complexity of the DI-WJ answer set semantics for propositional logic programs (entries denote completeness).

Program\Problem	Answer set existence	Cautious reasoning	Brave reasoning
General disjunctive	Σ_2^p	Π_3^p	Σ_3^p
General normal	Σ_2^p*	Π_2^p*	Σ_2^p*
Simple disjunctive	NP	Π_2^p	Σ_2^p
Simple normal	NP*	co-NP*	NP*

* Results that are immediate from Theorem 2 and the complexity results in [65].

As usual, we consider three canonical decision problems:

(1) *answer set existence*, i.e., the problem of deciding whether a given logic program Π has some DI-answer set;
(2) *cautious reasoning*, i.e., the problem of deciding whether a ground literal L is true in all DI-answer sets of Π; and
(3) *brave reasoning*, i.e., the problem of deciding whether a ground literal L is true in some DI-answer set of Π.

The complexity results are compactly summarized in Table 1. Besides the general case, also some of the syntactic fragments that we have discussed are considered. As can be seen from the table, the occurrence of the rule head operator | increases the complexity of the reasoning problems, except for answer set existence, by one level in the polynomial hierarchy. If we restrict the literal L to a ground atom, then the complexity of cautious reasoning for general disjunctive programs drops to Π_2^p, but stays the same in all other cases.

The results are derived as follows. By Corollary 2, the well-justified semantics from [65] coincides for general normal programs with the DI-WJ answer set semantics; the entries marked with an asterisk $*$ in Table 1 are thus justified by the complexity results in that paper.

Next, the results for simple disjunctive programs are easily obtained from the fact that DI-WJ answer sets and GL-answer sets coincide for simple normal programs and simple positive programs by Theorem 1, and by the well-known complexity results for the decision problems that we consider shown in [18] for these program classes.

Theorem 4. *Given a propositional simple disjunctive program Π and a ground literal L, deciding (i) whether Π has some DI-WJ answer set is* NP*-complete, (ii) whether L is true in every DI-WJ answer set of Π is Π_2^p-complete, and (iii) whether L is true in some DI-WJ answer set of Π is Σ_2^p-complete.*

It remains to justify the entries for general disjunctive programs. To this end, we first note the following lemma.

Lemma 1. *Given a propositional general disjunctive program Π and an interpretation I of Π, deciding whether condition (1) of Definition 10 holds is in Σ_2^p.*

Intuitively, to verify that condition (1) of Definition 10 holds we need to guess a selection *sel* witnessing that I is a well-justified answer set of Π_{sel}^I; that *sel* respects variant rule heads for I can be checked with an NP oracle in polynomial time, and checking whether I is indeed a WJ-answer set of Π_{sel}^I is in co-NP. Overall, this yields Σ_2^p membership.

For DI-WJ answer set existence, condition (2) in Definition 10 is irrelevant. We thus establish the following result.

Theorem 5. *Given a propositional general disjunctive program Π, deciding whether Π has some DI-WJ answer set is Σ_2^p-complete.*

Lemma 1 allows us furthermore to derive the following bound on DI-WJ answer set checking, i.e., deciding whether a given model of a general disjunctive program is a well-justified answer set.

Proposition 1. *Given a propositional general disjunctive program Π and an interpretation I of Π, deciding whether I is a DI-WJ answer set of Π is in D_2^p (thus in $\mathrm{P}^{\Sigma_2^p}$).*

We recall here that $\mathrm{D}_2^p = \{A \cap B \mid A \in \Sigma_2^p,\ B \in \Pi_2^p\}$ is the class of decision problems that informally are the "conjunction" of a problem in Σ_2^p and in Π_2^p, respectively; the prototypical D_2^p-complete problem is deciding whether given two QBFs $\Phi_i = \exists X_i \forall Y_i\, E_i(X_i, Y_i))$, $i = 1, 2$, it holds that Φ_1 and Φ_2 evaluate to true and false, respectively. Intuitively, the Σ_2^p-component is here the same as in Lemma 1 and the Π_2^p component is to check that no smaller $J \subset I$ satisfies condition (1), which is in co-$\Sigma_2^p = \Pi_2^p$. (In fact, DI-WJ answer set checking can also be shown to be D_2^p-hard, by a construction derived from the proof of Theorem 6 below, and is thus D_2^p-complete.)

As a consequence, brave reasoning has a Σ_3^p upper bound, and dually cautious reasoning a Π_3^p upper bound. The matching lower bound can be shown by a polynomial reduction from the following problem MINQASAT: given a QBF $\forall Y.E(X, Y)$,

where X and Y are lists (sets) of Boolean variables, and an atom A in X, is there some minimal variable assignment σ to X (viewed as model $\{X_i \in X \mid \sigma(X_i) = 1\}$), such that $\forall Y.E(\sigma(X), Y)$ evaluates to true and A is true in σ?

Lemma 2. *Deciding MINQASAT is Σ_3^p-complete.*

Theorem 6. *Given a propositional general disjunctive program Π and a ground literal L, deciding whether L is true in some (resp. every) DI-WJ answer set of Π is Σ_3^p-complete (resp. Π_3^p-complete).*

We note that by restricting the literal L in Theorem 6 to a ground atom, cautious reasoning is in Π_2^p; the reason is that an interpretation I of the program Π that fulfills condition (1) of Definition 10 but does not satisfy L suffices to refute the query; Lemma 1 implies that deciding the existence of such a model I is in Σ_2^p.

The higher computational complexity of brave and cautious reasoning in the general disjunctive case, compared to GL-answer sets or the well-known FLP-answer sets defined by Faber et al. [24] (where the operator | is viewed as the logical connective $\vee$) for which the problems are Σ_2^p-complete and Π_2^p-complete, respectively, offers higher problem solving capacity. However, even if some complex formulas are used in a general disjunctive program, the reasoning complexity may stay within Σ_2^p resp. Π_2^p. For example, if in the encoding of strategic sets in Example 5 the justifications σ_j^i were monotone DNFs $\sigma_j^i = D_1 \vee \cdots \vee D_k$, the problem of deciding relevance of a given company c_i for strategic sets, i.e., whether c_i occurs in some strategic set remains in Σ_2^p; this is because the rule $c_i \leftarrow \sigma_j^i$ can be replaced with rules $c_i \leftarrow D_1, \ldots, c_i \leftarrow D_k$, such that we obtain a positive simple disjunctive program. Note that the strategic sets which are given by the DI-answer sets (equivalently, the GL-answer sets resp. minimal models) of this program, are free from self-support and circular justification. If the justifications σ_j^i would be arbitrary propositional formulas and we are interested in strategic sets with that property, we can consider the DI-WJ answer sets of the respective general disjunctive program, which encode such strategic sets. As can be seen, deciding relevance of a company c_i for strategic sets is Σ_3^p-complete in this setting; this could not be expressed with a disjunctive program under the GL-answer set or the FLP-answer set semantics.

Computing some DI-WJ answer set

Furthermore, we note that computing some DI-WJ answer set of a simple normal respectively general normal program Π can be seen to be complete for the (multi-valued) functional analogs of NP and Σ_2^p, respectively.[13]

Notably, computing the DI-WJ answer sets of a general disjunctive program Π can be reduced to computing the minimal models of a QBF of the form $\exists Z \forall Y.E(X, Y, Z)$ that is constructible in polynomial time. To this end, one can express condition (1) in Definition 10 as a QBF of this form in polynomial time, where the free variables X describe I. The converse direction is also possible; the QBF reduction in the proof of Theorem 6 can be extended for the "$\exists Z$" part such that the minimal models of Φ correspond one-to-one to the DI-WJ answer sets of the constructed program Π_Φ. Thus, computing the DI-WJ answer sets of a general disjunctive program Π and the minimal models of a QBF $\exists Z \forall Y.E(X, Y, Z)$ are polynomial-time equivalent problems; this analogously holds for simple disjunctive programs and QBFs $\exists Z.E(X, Z)$.

Computing some minimal model of a QBF $\exists Z.E(X, Z)$ (resp., $\exists Z \forall Y.E(X, Y, Z)$) is possible using an oracle for NP (resp. Σ_2^p) in polynomial time. In fact, a bounded number of calls to a witness oracle for NP (resp. Σ_2^p) is sufficient, i.e., an oracle that not only answers "yes" or "no" to a query, but provides in the "yes" case also a polynomial-size *witness* to the query; e.g. in case of a SAT oracle, this is some satisfying assignment, and in case of an $\exists X \forall Y.E(X, Y)$ oracle some assignment σ to X such that $E(\sigma(X), Y)$ is a tautology.

In fact, computing some minimal model of a QBF $\exists Z.E(X, Z)$ is complete for the class $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$, which contains the class of multi-valued functions f for which some possible function value $y \in f(x)$ is computable in polynomial time with logarithmically many calls to a witness oracle in NP [9]. This result, which in essence was shown by Chen and Toda [13] – who considered a class $\mathrm{FNP}//\mathrm{OptP}[\log n]$ that coincides with $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$ on total multi-valued functions –, carries by the polynomial-time equivalence over to computing some DI-WJ answer set of a simple disjunctive program Π.

Based on techniques in [13], one can show that computing some minimal model of a QBF $\exists X \forall Y.E(X, Y)$ is complete for the class $\mathrm{FP}^{\Sigma_2^p}[\log, \mathrm{wit}]$, which is analogous to $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$ but allows for a Σ_2^p witness oracle. Consequently, computing some DI-WJ answer set of a general disjunctive program Π is $\mathrm{FP}^{\Sigma_2^p}[\log, \mathrm{wit}]$-complete.

For more details on witness oracles, we refer to [38,22] and references therein (see also Appendix B).

7.1. DI-answer sets for other base semantics

If in Definition 6 we take the GL-semantics as the base semantics, then we obtain for simple normal and simple disjunctive programs the same complexity results as for DI-WJ answer sets in Table 1; this is an immediate consequence of the fact that for simple normal programs, the GL_{nlp}-answer sets, the GL-answer sets and the well-justified answer sets coincide as

[13] The completeness is in the sense of strong computability, i.e. that all (and only) the solutions to the input instance are computed by a nondeterministic transducer without resp. with an NP oracle access.

shown by Shen et al. [65]; hence the DI-answer sets coincide under the three base semantics. The same complexity results hold if we take the FLP-answer set semantics as the base semantics, as the FLP-answer sets coincide with the GL-answer sets on simple normal programs. Note that deciding whether a propositional simple disjunctive program Π has DI-answer sets is NP-complete, which is lower than deciding whether Π has GL-answer sets; the latter is Σ_2^p-complete [18].

For the class of general normal and general disjunctive programs (which was not considered by Gelfond and Lifschitz [34]), the results in Table 1 also hold if we take the FLP-semantics as the base semantics. In fact, the results for general normal programs have been established by Shen et al. [65]; the membership parts for general disjunctive programs can be shown by simple guess-and-check algorithms, and the matching hardness results carry over from simple disjunctive programs (for answer set existence) and the proof of Theorem 6, as the reduction from QBFs given there works for FLP-semantics as the base semantics as well.

8. Related work

The seminal work of ASP is by Gelfond and Lifschitz [33,34], who considered simple normal and simple disjunctive programs, respectively and defined the seminal answer set semantics (i.e., the GL-semantics) for them. In the late 1990s, answer set programming was identified as a new declarative problem solving paradigm by Lifschitz [42], Marek and Truszczynski [49], and Niemela [53]. Since then, various extensions have been presented to expand the expressive power and applicability of ASP, including logic programs with aggregates [56,66,68,24,3], with external sources such as description logic programs [20], with propositional or first-order formulas [55,57,69,6,27,65], and with epistemic negations [31,70,39,64]. All of these extensions agree with the GL-semantics for simple disjunctive programs.

In this section we describe two representative related answer set semantics for disjunctive programs: the FLP-semantics by Faber et al. [24] and the Equilibrium logic-based semantics in [26,55,27]. We also describe split programs and give a solution to the open problem presented by Hitzler and Seda [36] about characterizing split normal derivatives of a simple disjunctive program Π. For illustration of their differences, we present one more example below.

Example 13. Consider the following disjunctive programs.

$$\Pi_1: \quad p \mid \neg p \tag{1}$$
$$p \leftarrow \neg p \tag{2}$$

$$\Pi_2: \quad p \vee \neg p \tag{1'}$$
$$p \leftarrow \neg p \tag{2'}$$

Π_1 differs from Π_2 in that by applying rule (1) we nondeterministically infer p or $\neg p$, while by applying rule (1′) we infer $p \vee \neg p$.

For Π_1, rule (1) generates two potential answer sets $I_1 = \{p\}$ and $I_2 = \emptyset$, and together with the constraint rule (2) (i.e., there is no answer set that does not contain p) yields a unique candidate answer set I_1. As I_1 is a minimal model of Π_1, it is a desired answer set of Π_1.

In contrast, as $p \vee \neg p$ is a tautology, Π_2 amounts to the program $\{\top,\ p \leftarrow \neg p\}$ and thus should have no answer set.

It is easy to check that $I_1 = \{p\}$ is a DI-answer set of Π_1 and Π_2 has no DI-answer set.

8.1. FLP-semantics

The FLP-semantics [23,24] was originally oriented towards giving an answer set semantics to simple disjunctive programs extended with aggregates; it was later deployed to other extensions of logic programs, including description logic programs [21,20,47], modular logic programs [14], and logic programs with first-order formulas [6]. It is straightforward to lift the FLP-semantics from simple disjunctive programs to general disjunctive programs as follows. Let Π be a general disjunctive program and I an interpretation; then I is an *FLP-answer set* of Π if I is a minimal model of the *FLP-reduct*

$$f\Pi^I = \{r \in ground(\Pi) \mid I \text{ satisfies } body(r)\}$$

of Π w.r.t. I.

The FLP-semantics differs from the DI-semantics mainly in the following three aspects: First, it does not distinguish between the rule head operator | and the logical connective $\vee$; specifically it interprets | essentially as $\vee$. As a result, the two disjunctive programs in Example 13 are semantically the same under the FLP-semantics and have no FLP-answer set. Second, for simple disjunctive programs, it coincides with the GL-semantics. Finally, as shown in [65], FLP-answer sets may have circular justifications that are caused by self-supporting loops.

8.2. Equilibrium logic-based semantics

Another extensively studied answer set semantics is by Ferraris [26], which defines answer sets for logic programs with propositional formulas and aggregates based on a new definition of equilibrium logic [54]. This semantics was further

extended to first-order formulas in terms of a modified circumscription [27]. Following a similar way, Pearce [55] proposed to identify answer sets with equilibrium models in equilibrium logic. It turns out that the answer set semantics of [55] coincides with that of [26] in the propositional case and with that of [27] in the first-order case. Therefore, we refer to them as *Equilibrium logic-based semantics*.

For any ground formula F, the *Ferraris-reduct* of F w.r.t. an interpretation I, denoted $F^{\underline{I}}$, is defined recursively as follows [26]:

- $\bot^{\underline{I}} = \bot$
- $A^{\underline{I}} = \begin{cases} A & \text{if } A \text{ is a ground atom and } I \text{ satisfies } A \\ \bot & \text{otherwise} \end{cases}$
- $(\neg F)^{\underline{I}} = \begin{cases} \bot & \text{if } I \text{ satisfies } F \\ \top & \text{otherwise} \end{cases}$
- $(G \odot H)^{\underline{I}} = \begin{cases} G^{\underline{I}} \odot H^{\underline{I}} & \text{if } I \text{ satisfies } G \odot H \ (\odot \in \{\wedge, \vee, \supset\}) \\ \bot & \text{otherwise} \end{cases}$

Let Π be a propositional program and Π' be Π with $|$ and $\leftarrow$ replaced by $\vee$ and $\subset$, respectively. Then I is an answer set of Π under the Equilibrium logic-based semantics if I is a minimal model of the Ferraris-reduct $\Pi'^{\underline{I}}$ of Π'.

For simple disjunctive programs, the Equilibrium logic-based semantics coincides with the GL-semantics. Moreover, it does not distinguish between the rule head operator $|$ and the logical connective $\vee$; specifically it does not treat $p \vee \neg p$ as a tautology. As a result, the two disjunctive programs in Example 13 have the same answer set $I = \{p\}$ under the Equilibrium logic-based semantics.

Truszczynski [69] also presented a reduct that slightly differs from the Ferraris-reduct in the way to handle $(G \supset H)^{\underline{I}}$, i.e.,

- $(G \supset H)^{\underline{I}} = \begin{cases} G \supset H^{\underline{I}} & \text{if } I \text{ satisfies both } G \text{ and } H \\ \top & \text{if } I \text{ does not satisfy } G \\ \bot & \text{otherwise} \end{cases}$

8.3. Split programs

For a simple disjunctive program Π, Hitzler and Seda [36] proposed to split Π into a collection of simple normal programs, called *normal derivatives*. Informally, a normal derivative $P(\Pi)$ of Π is obtained from $ground(\Pi)$ by replacing every disjunctive rule $A_1|\cdots|A_k \leftarrow body(r)$ (where $k \geq 2$) arbitrarily with one or more rules of the form $A_i \leftarrow body(r)$, $1 \leq i \leq k$. For example, take $\Pi = \{p \mid q \leftarrow \neg s\}$; then Π has three normal derivatives: $P_1(\Pi) = \{p \leftarrow \neg s\}$, $P_2(\Pi) = \{q \leftarrow \neg s\}$, and $P_3(\Pi) = \{p \leftarrow \neg s, q \leftarrow \neg s\}$.

Hitzler and Seda [36] attempted to use the normal derivatives to characterize the GL-semantics of a simple disjunctive program Π and showed that if an interpretation I is an answer set of Π under the GL-semantics, then I is an answer set of some normal derivative of Π under the GL_{nlp}-semantics. Take the above logic program Π; $I = \{p\}$ is an answer set of Π under the GL-semantics, which is also an answer set of the normal derivative $P_1(\Pi)$ under the GL_{nlp}-semantics.

However, Hitzler and Seda [36] raised an open question of characterizing normal derivatives whose answer sets are answer sets of Π. Specifically, the problem states that for any interpretation I, determine some normal derivatives such that I is an answer set of Π under the GL-semantics if and only if I is an answer set of these normal derivatives under the GL_{nlp}-semantics.

It is particularly interesting to note that our characterization of the GL-semantics using the disjunctive program reduct (see Theorem 2) leads to a satisfactory solution for this open problem. For a logic program Π, an interpretation I and a head selection *sel* on I, we denote

$$P_{sel}(\Pi, I) = \{sel(head(r), I) \leftarrow body(r) \mid r \in ground(\Pi)\}$$

and

$$ND(\Pi, I) = \{P_{sel}(\Pi, I) \mid sel \text{ is a head selection on } I\}.$$

Note that $ND(\Pi, I)$ is the collection of normal derivatives obtained by applying every head selection on I. Thus for any head selection *sel* on a model I, we have

$$\begin{aligned} P_{sel}(\Pi, I) &= \{sel(head(r), I) \leftarrow body(r) \mid r \in ground(\Pi)\} \\ &= \Pi^I_{sel} \cup \{sel(head(r), I) \leftarrow body(r) \mid r \in ground(\Pi) \text{ and} \\ &\qquad\qquad body(r) \text{ is not satisfied by } I\} \end{aligned}$$

Then the following theorem gives a solution to the above open problem.

Theorem 7. *Let Π be a simple disjunctive program and I an interpretation. Then I is an answer set of Π under the GL-semantics if and only if I is an answer set of every normal derivative in $ND(\Pi, I)$ under the GL_{nlp}-semantics.*

The proof of this theorem requires the following lemma.

Lemma 3. *Let $(\Pi^I_{sel})^I$ and $(P_{sel}(\Pi, I))^I$ be the GL-reduct of Π^I_{sel} and $P_{sel}(\Pi, I)$, respectively. Then an interpretation I is a minimal model of $(\Pi^I_{sel})^I$ if and only if I is a minimal model of $(P_{sel}(\Pi, I))^I$.*

For example, the above logic program $\Pi = \{p \mid q \leftarrow \neg s\}$ has three models: $I_1 = \{p\}$, $I_2 = \{q\}$ and $I_3 = \{p, q\}$. For I_1, $ND(\Pi, I_1)$ has only one normal derivative $P_{sel}(\Pi, I_1) = \{p \leftarrow \neg s\}$, where *sel* selects p from the rule head $p \mid q$. I_1 is an answer set of $P_{sel}(\Pi, I_1)$ under the GL_{nlp}-semantics, and by Theorem 7 is an answer set of Π under the GL-semantics. Similarly, I_2 is an answer set of Π. For I_3, $ND(\Pi, I_3)$ consists of two normal derivatives, $P_{sel_1}(\Pi, I_3) = \{p \leftarrow \neg s\}$ and $P_{sel_2}(\Pi, I_3) = \{q \leftarrow \neg s\}$, where sel_1 and sel_2 select p and q from the rule head $p \mid q$, respectively. I_3 is not an answer set of $P_{sel_1}(\Pi, I_3)$, and by Theorem 7 is not an answer set of Π.

Finally, we note that the normal derivatives of a simple disjunctive program Π coincide with the split programs of [61], whose answer sets constitute the so called "possible worlds" of Π. In the preliminary work, Sakama [60] had defined possible worlds of a positive simple disjunctive program Π as the minimal models of all its split programs resp. derivatives $P(\Pi)$, which coincide with the "possible models" in the semantics of [12] and the sustained models considered by Decker and Casamayor [17], who used a level mapping on the atoms to foster derivability from facts similar as Fages [25]. Apparently, the possible worlds semantics is more liberal than both the GL-semantics and the DI-semantics; regarding answer set computation, it is for simple disjunctive programs "cheaper," as computing some possible world is feasible in nondeterministic polynomial time, while computing some DI- resp. GL-answer set is $\Delta^p_2[\log n]$-hard resp. Π^p_2-hard.

8.4. Approximation of GL-answer sets

As every GL-answer set of a simple disjunctive program is a DI-answer set (using GL_{nlp} as the base semantics), we might view DI-answer sets also as approximations of GL-answer sets whose approximation degree may be controlled by a parameter measuring the discrepancy to being a GL-answer set. To this end, for instance the well-known notion of unfounded sets can be used, in combination with the characterization of GL-answer sets by Theorem 2.

We recall that a set U of ground atoms is an unfounded set of a (ground) simple normal program Π w.r.t. an interpretation I if for every atom $A \in I$ and every rule $r \in \Pi$ with head A, either (i) for some positive body literal B_i we have $B_i \notin I$ or $B_i \in U$, or (ii) for some negative body literal $\neg C_j$, we have $C_j \in I$ [72]. It is well-known that always a greatest unfounded set exists which we denote by $U_\Pi(I)$, and that a model I of Π is a GL-answer set iff $I \cap U_\Pi(I) = \emptyset$ (implicit in [41]).

By Theorem 2, a GL-answer set I is characterized by the property that for every head selection function *sel*, I is a GL-answer set of the program Π^I_{sel}; that is, $U_{\Pi^I_{sel}}(I) \cap I = \emptyset$ holds for every *sel*. In order to assess the approximation degree of a DI-answer set I to a GL-answer set, we may take the value

$$\max_{sel \text{ w.r.t. } I} |U_{\Pi^I_{sel}}(I) \cap I|;$$

intuitively, this is the maximum discrepancy to the condition that a GL-answer set would satisfy over all head selections.[14] Other forms of aggregations (sum, average) and individual discrepancy valuations are of course conceivable.

By imposing a limit on this value, the degree of the approximation can be controlled. This lends itself for handling inconsistency in ASP programs under GL-semantics; as such, it would enrich the repertoire of respective methods that include among others paracoherent semantics [62,2] and CR-Prolog [4], but with a different underpinning:

- In paracoherent semantics, we add missing (non-provable) assumptions, while in DI-semantics, we are allowed to make additional inferences. In Example 1, paracoherent semantics would yield an answer set in which a is true and b is possibly true (by assumption).
- In CR-Prolog, each ordinary program is equipped with further, user-defined rules of which a part can be added to the program in order to enable answer sets. This approach is at another (the meta) level, while using DI-answer sets is at no extra effort and does not require further specifications by the user.

9. Conclusion

We have presented a new answer set semantics, namely the DI-semantics for disjunctive programs, which avoids interpreting of the disjunctive operator | in the way similar to classical connective $\vee$ in general and does not enforce that

[14] Notably, $\max_{U \in UFS(\Pi, I)} |U \cap I|$, where $UFS(\Pi, I)$ is the set of all (disjunctive) unfounded sets of Π w.r.t. I according to [41], would yield the same value, due to a correspondence between the disjunctive unfounded sets of Π w.r.t. I and the unfounded sets of all Π^I_{sel} w.r.t. I.

answer sets should be minimal models. Briefly, we introduced a head selection function to formalize the operator | as a nondeterministic inference operator, and with it transformed a disjunctive program into a normal program called a disjunctive program reduct. We then defined candidate answer sets from the reduct w.r.t. a base answer set semantics $\mathcal{X}$ for normal programs. Finally, we defined DI-answer sets to be minimal candidate answer sets. The new semantics is generic and applicable to extend any answer set semantics $\mathcal{X}$ for normal programs to disjunctive programs.

By replacing the base semantics $\mathcal{X}$ with the GL_{nlp}-semantics, we induced a DI-semantics for simple disjunctive programs. We also presented a novel characterization of the GL-semantics in terms of a disjunctive program reduct. This characterization reveals the essential difference of the DI-semantics from the GL-semantics; it also leads us to giving a satisfactory solution to the open problem presented by Hitzler and Seda [36] about characterizing split normal derivatives of a simple disjunctive program Π such that answer sets of the normal derivatives are answer sets of Π under the GL-semantics.

By replacing $\mathcal{X}$ with the well-justified semantics, we induced a DI-semantics for general disjunctive programs. This closed an open task presented in [65] about extending the well-justified semantics from general normal programs to general disjunctive programs.

We have characterized the complexity of the new semantics for propositional programs. We showed that deciding DI-answer set existence for a simple disjunctive program Π is NP-complete, and whether a ground literal is true in some (resp. every) DI-answer set of Π is Σ_2^p-complete (resp. Π_2^p-complete). For general disjunctive programs, the complexity of the DI-semantics increases by one level in the polynomial hierarchy.

Perspectives for future work. An important aspect for the DI-answer set semantics is integration into the existing ASP landscape, and in particular in solvers in connection with the GL-semantics. For simple disjunctive programs, as the DI-semantics is a relaxation of the GL-semantics, one may focus in practice on those DI-answer sets that are also GL-answer sets; if no GL-answer set exists, one then resorts to more permissive DI-answer sets. In order to discriminate among different DI-answer sets, one may establish a preference on them for singling out those which are closer to the definition of a GL-answer set than others, as we briefly discussed in Section 8.4. It remains as an interesting issue to study and develop such preference relations.

Finally, an efficient implementation of the DI-semantics remains to be developed. By exploiting the existing ASP reasoner DLVHEX,[15] which allows one to access via its plugin architecture external computations and in particular to evaluate answer set programs, building a simple prototype implementation can be easily done. However, alternative, more efficient implementations, e.g. by means of efficient ASP solvers such as clasp, clingo and wasp, or by genuine algorithms will have to be considered.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgement

We thank the Associate Editor and anonymous reviewers for their valuable comments on a draft of this article, which helped to improve its quality a lot. This work is supported in part by NSFC grants 61976205 and 61379043, and the Austrian Science Fund (FWF) via the project P27730.

Appendix A. Generalized Strategic Companies

Example 14 *(Generalized Strategic Companies problem continued).* In order to express the GSC problem in Example 5 under the GL-semantics, following the customary guess-and-check (i.e., generate and test) approach one needs (1) to generate a candidate solution and (2) to check that the candidate is indeed a solution to the problem. Part (1) may consist in generating sets $C' \subseteq C$ of companies that satisfy all rules of the program Π (i.e., the conditions (1) and (2) of a strategic set). To this end, we may add for the concrete instance guessing clauses

$$c_i \mid \overline{c}_i \qquad i = 1, 2, 3$$

in Π, where the $\overline{c}_i$'s are fresh atoms; the answer sets of the resulting program Π_1 include each strategic set for the problem instance. In Part (2), we must eliminate those candidates C' which violate condition (3) of a strategic set, i.e., there exists some smaller candidate $C'' \subset C'$. This is more difficult to achieve; we essentially have to resort to the saturation technique, by which we can generate and test smaller candidates C'' (cf. [19] for an extensive discussion of the subject). To do so, we can create a copy of Π_1, in which each atom c_i and $\overline{c}_i$ are replaced by c'_i and $\overline{c'}_i$, respectively, the negative literal $\neg c'_2$ in the so obtained rule $c'_3 \leftarrow c'_1 \wedge \neg c'_2$ is replaced by $\overline{c'}_2$ (as the latter emulates $\neg c'_2$), and the following *saturation rules* are added, where *sat* and the eq_i are fresh atoms, and $i = 1, 2, 3$:

[15] http://www.kr.tuwien.ac.at/research/systems/dlvhex.

$$\bot \leftarrow \neg sat$$
$$c'_i \leftarrow sat$$
$$\overline{c'}_i \leftarrow sat$$
$$eq_i \leftarrow sat$$
$$sat \leftarrow \overline{c}_i \wedge c'_i$$
$$sat \leftarrow c'_i \wedge \overline{c'}_i$$
$$sat \leftarrow eq_1 \wedge eq_2 \wedge eq_3$$
$$eq_i \leftarrow c_i \wedge c'_i$$
$$eq_i \leftarrow \overline{c}_i \wedge \overline{c'}_i$$

The resulting program Π_2 has, equipped with an answer set I of Π_1 as further facts, a single answer set I'_{sat} iff I represents a strategic set C' for the considered GSC instance. In I'_{sat}, the "saturation atom" sat is necessarily contained by the constraint at the top, as well as all other atoms c'_i, $\overline{c'}_i$, and eq_i by the next three rules. Intuitively, the presence of sat shows that a guess for a smaller answer set $J \subset I$, which as J' in the copy would represent a smaller set $C'' \subset C'$, is eliminated: the first rule deriving sat catches that $J \not\subseteq I$ holds, and the second that the guess is inconsistent; the rules involving eq_i serve to catch the case where $C'' = C'$. We remark that catching the case $C'' = C'$ is more cumbersome if we had a uniform encoding in which goods, companies and control are represented in terms of relations; we refer to [19] for more details. For this, (recursive) aggregates may alternatively be used in saturation, e.g.

$$sat \leftarrow \#count\{X : c(X)\} = N,\ \#count\{X : c'(X)\} = M,\ M = N$$

where $c/1$ (resp., $c'/1$) would contain the companies that are selected for a candidate solution (resp., a smaller candidate solution); however the particular semantics of aggregates used has to be considered with care (cf. [1] for discussion).

Appendix B. Proofs

Proof of Theorem 1. For any GL-answer set I of Π, I is both a minimal model of Π and a minimal model of the GL-reduct Π^I. For any head selection sel, let $(\Pi^I_{sel})^I$ be the GL-reduct of Π^I_{sel}.

We first show the case that Π is a simple normal program. Then for any interpretation there is a unique selection function sel. For a model I we also have $(\Pi^I_{sel})^I = \Pi^I$.

($\Longrightarrow$) Let I be a DI-answer set of Π. Then I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics, i.e., I is a minimal model of $(\Pi^I_{sel})^I$ and thus is a minimal model of Π^I. Hence I is an answer set of Π under the GL-semantics.

($\Longleftarrow$) Let I be an answer set of Π under the GL-semantics. Then I is a minimal model of Π^I and thus is a minimal model of $(\Pi^I_{sel})^I$. This means that I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics. As I is a minimal model of Π, I is a DI-answer set of Π.

Next we show the case that Π is a positive simple disjunctive program. Note that for any I and sel, $(\Pi^I_{sel})^I = \Pi^I_{sel}$. For a model I, $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m$ is a rule in Π^I whose body is satisfied by I iff for $sel(A_1 \mid \cdots \mid A_k, I) = A_i$, $A_i \leftarrow B_1 \wedge \cdots \wedge B_m$ is a rule in Π^I_{sel} and thus in $(\Pi^I_{sel})^I$.

($\Longrightarrow$) Let I be a DI-answer set of Π. Then, (1) for some head selection function sel, I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics, i.e., I is a minimal model of Π^I_{sel}, and (2) Π has no model $J \subset I$ satisfying condition (1). The condition (2) means that for any model $J \subset I$ of Π and any head selection function sel, J is not a minimal model of Π^J_{sel}.

Assume towards the contradiction that I is not a minimal model of Π. Let $J \subset I$ be a minimal model of Π. Then (1) J is a minimal model of Π^J, and (2) for any head selection function sel, J is not a minimal model of Π^J_{sel}. Let sel be an arbitrary head selection function and $L \subset J$ be a minimal model of Π^J_{sel}. For any rule r in Π^J of the form $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m$ such that L satisfies $body(r)$ (and thus J satisfies $body(r)$), there must be a rule r' in Π^J_{sel} of the form $A_i \leftarrow B_1 \wedge \cdots \wedge B_m$, where $sel(A_1 \mid \cdots \mid A_k, J) = A_i$, such that L satisfies A_i and thus satisfies $head(r)$. This means that L is a model of Π^J, contradicting the assumption that J is a minimal model of Π^J. Hence I is a minimal model of Π and thus an answer set under the GL-semantics.

($\Longleftarrow$) Let I be an answer set of Π under the GL-semantics. Then I is both a minimal model of Π and Π^I. Assume towards a contradiction that for any head selection sel, I is not an answer set of Π^I_{sel} under the GL_{nlp}-semantics, i.e., I is not a minimal model of Π^I_{sel}. Let $J \subset I$ be a minimal model of Π^I_{sel}. For any rule r in Π^I of the form $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m$ such that J satisfies $body(r)$ (and thus I satisfies $body(r)$), there must be a rule in Π^I_{sel} of the form $A_i \leftarrow B_1 \wedge \cdots \wedge B_m$, where $sel(A_1 \mid \cdots \mid A_k, I) = A_i$, such that J satisfies A_i and thus satisfies $head(r)$. This means that J is a model of Π^I, contradicting the assumption that I is a minimal model of Π^I. Hence for some head selection sel, I is an answer set of Π^I_{sel} under the GL_{nlp}-semantics. As I is a minimal model of Π, I is a DI-answer set of Π. □

Proof of Theorem 2. ($\Longrightarrow$) For any GL-answer set I of Π, I is both a minimal model of Π and a minimal model of the GL-reduct Π^I. For any head selection sel, let $(\Pi^I_{sel})^I$ be the GL-reduct of Π^I_{sel}. Then, $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n$ is a rule in $ground(\Pi)$ whose rule body is satisfied by I iff $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m$ is a rule in Π^I iff for $sel(A_1 \mid \cdots \mid A_k, I) = A_i$, $A_i \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n$ is a rule in Π^I_{sel} iff $A_i \leftarrow B_1 \wedge \cdots \wedge B_m$ is a rule in $(\Pi^I_{sel})^I$.

Let I be a GL-answer set of Π. Then I is both a minimal model of Π and Π^I. Assume towards a contradiction that for some head selection sel, I is not an answer set of Π^I_{sel} under the GL$_{nlp}$-semantics. Then I is not a minimal model of $(\Pi^I_{sel})^I$. Let $J \subset I$ be a minimal model of $(\Pi^I_{sel})^I$. For any rule r in Π^I of the form $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m$ such that J satisfies $body(r)$ (and thus I satisfies $body(r)$), there must be a rule in $(\Pi^I_{sel})^I$ of the form $A_i \leftarrow B_1 \wedge \cdots \wedge B_m$, where $sel(A_1 \mid \cdots \mid A_k, I) = A_i$, such that J satisfies A_i and thus satisfies $head(r)$. This means that J is a model of Π^I, contradicting the assumption that I is a minimal model of Π^I. Hence for every head selection function sel, I is an answer set of Π^I_{sel} under the GL$_{nlp}$-semantics.

($\Longleftarrow$) Assume that for every head selection sel, I is an answer set of Π^I_{sel} under the GL$_{nlp}$-semantics, i.e., I is a minimal model of $(\Pi^I_{sel})^I$. Assume towards a contradiction that I is not a GL-answer set of Π, i.e., I is not a minimal model of Π^I. Let $J \subset I$ be a minimal model of Π^I. Consider a head selection sel with the property that for every rule r in Π^I of the form $A_1 \mid \cdots \mid A_k \leftarrow B_1 \wedge \cdots \wedge B_m$ such that J satisfies $body(r)$ (and thus I satisfies $body(r)$), $sel(A_1 \mid \cdots \mid A_k, I) = A_i$ with A_i being satisfied by J. Such A_i must exist because J is a model of Π^I. Then J is a model of $(\Pi^I_{sel})^I$. This means that I is not a minimal model of $(\Pi^I_{sel})^I$, contradicting our assumption. Hence, I is a GL-answer set of Π. □

Proof of Theorem 3. Let Π and $\Pi' = \Pi \cup \{\bot \leftarrow body(r)\}$ be two simple disjunctive programs, and let I be a DI-answer set of Π' but not of Π. Then for some selection function sel, I is an answer set of Π'^I_{sel} under the GL$_{nlp}$-semantics. As I does not satisfy $body(r)$, I is also an answer set of Π^I_{sel} under the GL$_{nlp}$-semantics, which means that I is a candidate DI-answer set of Π. As I is not a DI-answer set of Π, by Definition 7 it must be a non-minimal candidate DI-answer set of Π. □

Proof of Theorem 4. The membership parts can be shown with simple guess and check algorithms: a head selection sel and an interpretation I for condition (1) of a DI-WJ answer set can be guessed and checked, by verifying that sel respects variant rule heads for I (this is easily done in polynomial time) and that I is a GL-answer set of Π^I_{sel}; the latter is also feasible in polynomial time (as the program Π^I_{sel} is normal). If this succeeds (which is in NP), then some DI-WJ answer set exists, which proves (i). For (ii) and (iii), we require that I does not satisfy L resp. satisfies L and check that no $J \subset I$ exists that satisfies condition (1) for some sel'; the latter check is in co-NP. In conclusion, we obtain membership (i) in NP, (ii) in Π^p_2, and (iii) in Σ^p_2, respectively.

The matching hardness results are immediate by Theorem 1 and the facts that GL-answer set existence for a simple normal program is NP-complete, cf. [34], and that cautious and brave inference of a literal from the GL-answer sets of a simple positive disjunctive program is Π^p_2- resp. Σ^p_2-hard, as shown in [18]. □

Proof of Lemma 1. Indeed, if condition (1) holds, then there exists some head selection function sel on I such that for every atom $A \in I$ it holds that $lfp(T_\Pi(\emptyset, I, sel)) \cup \neg I^- \models A$. To witness this, we can guess sel, check that sel respects variant rule heads for I, compute $lfp(T_\Pi(\emptyset, I, sel))$ and then do all the entailment checks in polynomial time using an NP oracle. As for respecting variant rule heads, we can compute the equivalence relation $F_i \equiv F_j$ for all formulas $F_1, \ldots, F_m$ that occur in rule heads of Π naively with $m(m-1)/2$ calls to an NP oracle; once available, checking the condition for is easy (in fact, computing $\equiv$ first to constrain the guess for sel may be more efficient). This shows membership in Σ^p_2. Note that if Π is a simple disjunctive program, then checking that sel respects variant rule heads for I is easily done in polynomial time; furthermore, computing $lfp(T_\Pi(\emptyset, I, sel))$ and doing all checks $lfp(T_\Pi(\emptyset, I, sel)) \cup \neg I^- \models A$ is feasible in polynomial time, which implies membership in NP for this class of programs. □

Proof of Theorem 5. Membership in Σ^p_2 holds since we can guess a DI-WJ answer set I of Π and check it by Lemma 1 using a Σ^p_2 oracle call. The guess and oracle check can be combined into a single (polynomial size) guess followed by a call to a co-NP oracle; this shows membership in Σ^p_2. The Σ^p_2-hardness is inherited from the Σ^p_2-completeness of deciding answer set existence under the well-justified answer set semantics of general normal programs [65], which by Corollary 2 coincides with the well-justified answer set semantics. □

Proof of Proposition 1. By Lemma 1, we can test using a Σ^p_2 oracle that condition (1) of Definition 10 holds for I. Furthermore, we can test using a Π^p_2 oracle that no smaller $J \subset I$ exists that satisfies condition (1); that is, that for every $J \subset I$ and selection function sel',[16] it holds that (*) some atom $A \in J$ exists such that $lfp(T_\Pi(\emptyset, J, sel')) \cup \neg J^- \not\models A$. To witness (*) for sel' and J, we can guess (a) for each $i = 1, \ldots, m$, where $m = |\Pi|$, a set $N_i \subseteq \Pi$ of rules r with associated interpretations J^r_i such that

[16] As discussed in Lemma 1, equivalence $F_j \equiv F_j$ of formulas F_i, F_j in rule heads may be computed ahead with an NP oracle so that the guess for sel' can be constrained to respect variant rule heads for I.

$$J_i^r \text{ satisfies the formulas } S_{i-1} \cup \neg J^- \cup \{\neg body(r)\} \tag{B.1}$$

where $S_0 = \emptyset$ and $S_i = \{sel'(head(r), J) \mid r \in \Pi \setminus N_i\}$, and (b) an interpretation J' such that

$$J' \text{ satisfies } S_m \cup \neg J^- \cup \{\neg A\}. \tag{B.2}$$

Informally, N_i is a subset of those rules r in Π whose body is not entailed by $T_\Pi^{i-1}(\emptyset, J, sel')) \cup \neg J^-$, witnessed by the interpretation J_i^r; consequently, S_i is a superset of $T_\Pi^i(\emptyset, J, sel')$. Hence if (B.1) holds for all $j = 1, \ldots, m$ then S_m is a superset of $lfp(T_\Pi(\emptyset, J, sel'))$, and if in addition (B.2) holds, then we have $lfp(T_\Pi(\emptyset, J, sel)) \cup \neg J^- \not\models A$. Conversely, if $lfp(T_\Pi(\emptyset, J, sel')) \cup \neg J^- \not\models A$ then by guessing all N_i, all J_i^r and J' appropriately (B.1) and (B.2) will hold. As the guess has polynomial size, it follows that deciding (*) is in NP; consequently, to decide that for all $J \subset I$ and sel' condition (*) holds is in Π_2^p, i.e., deciding whether condition (1) does not hold for some $J \subset I$ is in Π_2^p. Thus as the problem amounts to the conjunction of a Σ_2^p and a Π_2^p problem, it is in D_2^p. □

Proof of Lemma 2. Membership in Σ_3^p is immediate by a guess and check argument, using a Π_2^p oracle.

As for the hardness part, let $\Phi = \exists X^1 \forall X^2 \exists X^3 F(X^1, X^2, X^3)$ be a QBF with variables $X^i = \{X_1^i, \ldots, X_{n_i}^i\}$, $i = 1, 2, 3$. Let $\bar{X}^1 = \{\bar{X}_1^1, \ldots, \bar{X}_{n_1}^1\}$ be a copy of X^1, and let X_0^2 be a fresh variable. Define

$$\Psi(X^1 \bar{X}^1 X^2 X_0^2) = \forall X^3 (\textstyle\bigwedge_{i=1}^{n_1} (X_i^1 \equiv \neg \bar{X}_i^1) \wedge (\bigwedge_{j=0}^{n_2} X_j^2 \vee \neg F(X^1, X^2, X^3))).$$

Informally, every assignment σ to $X = X^1 \bar{X}^1 X^2 X_0^2$ in which every X_j^i and $\bar{X}_j^i$ have opposite values and all variables in $X^2 X_0^2$ are true will be a model of $\Psi(X)$, and is a candidate for a minimal model. In fact, σ is not minimal if and only if there exists some assignment σ' that (a) coincides with σ on $X^1 \bar{X}^1$, (b) assigns false to some variable in X^2, and (c) is such that $\forall X^3 \neg F(\sigma(X), X^3)$ evaluates to true. That is, σ is minimal if and only if the QBF $\forall X^2 \exists X^3 F(\sigma(X^1), X^2, X^3)$ evaluates to true. Note that in every such σ, the variable X_0^2 is true, and that in every minimal model σ' smaller than σ we must have that X_0^2 is false. Hence, $A = X_0^2$ is true in some minimal model of $\Psi(X)$ if and only if the QBF Φ is true. This proves the Σ_3^p-hardness. □

Proof of Theorem 6. As for the membership parts, we can guess a DI-WJ answer set I of Π such that I satisfies L (resp. does not satisfy L) and verify I with the help of a Σ_2^p oracle in polynomial time as follows from Proposition 1.

The Σ_3^p hardness of brave reasoning is shown by the following reduction from MINQASAT. Let $\Phi = \forall Y.E(X, Y)$ be a QBF as in Lemma 2, where $X = \{X_1, \ldots, X_n\}$ and $Y = \{Y_1, \ldots, Y_m\}$. We construct the following logic program Π:

$$\begin{aligned} X_i \mid \neg X_i \leftarrow & \qquad \text{for all } i = 1, \ldots, n \\ Y \wedge p \leftarrow & \; E(X, Y) \\ p \leftarrow & \; \neg p \end{aligned}$$

where p is a fresh atom and Y stands for $Y_1 \wedge \cdots \wedge Y_m$. Intuitively, the first rules guess a truth assignment σ to the variables in X, and the second rule checks whether under this assignment $E(X, Y)$ is a tautology, i.e., $\forall Y.E(\sigma(X), Y)$ evaluates to true; in this case, the atom p is derived, which must be true in every DI-WJ answer set of Π. Likewise all Y_i must be true, and hence no $\neg Y_j$ occurs in $\neg I^-$. In fact, the truth assignments σ to X such that $\forall Y.(\sigma(X), Y)$ evaluates to true correspond one-to one to the models $I = I_\sigma$ of Π that fulfill condition (1) of Definition 9, where $I_\sigma = \{X_i \mid \sigma(X_i) = 1\} \cup Y \cup \{p\}$. Consequently, by condition (2) of this definition, the DI-WJ answer sets of Π correspond to the minimal assignments σ to X such that $\forall Y.E(\sigma(X), Y)$ evaluates to true. Hence, Φ with $A \in X$ is a yes-instance of MINQASAT if and only if A is true in some DI-WJ answer set of Π. This proves Σ_3^p-hardness of brave reasoning. The Π_3^p-hardness of cautious reasoning results by asking whether $\neg A$ is true in all DI-WJ answer sets of Π.

Remarks. (1) We can extend the construction to show that the minimal models of a QBF $\exists Z \forall Y.E(X, Y, Z)$ correspond to the DI-WJ answer sets of a program Π constructed in polynomial time. To this aim, we add rules

$$Z_i \mid \hat{Z}_i \leftarrow \qquad \text{for all } Z_i \in Z$$

and change the rule $Y \wedge p \leftarrow E(X, Y)$ to

$$Z \wedge \hat{Z} \wedge Y \wedge p \leftarrow \hat{E}(X, Y, Z) \vee \bigvee_{i=1}^{k} (Z_i \wedge \hat{Z}_i)$$

where Z (resp. $\hat{Z}$) stands for $\bigwedge_{Z_i \in Z} Z_i$, (resp. $\bigwedge_{Z_i \in Z} \hat{Z}_i$), and $\hat{E}(X, Y, Z)$ stands for $E(X, Y, Z)$ put in negation normal form, where each occurrence of $\neg Z_i$ is replaced by $\hat{Z}_i$. Intuitively, any DI-WJ answer set must contain each Z_i and the fresh $\hat{Z}_i$, which simulates the complement of Z_i. When it comes to applying the rule with $\hat{E}(X, Y, Z)$ in the body in the fixpoint construction, one of Z_i or $\hat{Z}_i$ is selected and informally set to true, in addition to the selected literals X_i resp. $\neg X_i$. We can

then derive p iff $E(X, Y, Z)$ is under the assignment σ of X and Z a tautology. Thus, the DI-WJ answer sets of Π are given by extending I_σ from above with Z and $\hat{Z}$, i.e., by $\{X_i \mid \sigma(X_i) = 1\} \cup Y \cup Z \cup \hat{Z} \cup \{p\}$ where $\exists Z \forall Y . E(\sigma(X), Y, Z)$ evaluates to true.

(2) From an extended construction, we can conclude D_2^p-hardness of DI-WJ answer set checking. First assume that $E(X, Y)$ is satisfied if all variables in X are set to 1, i.e., $\forall Y . E(X = 1, Y)$ evaluates to true. Then $I_1 = X \cup Y \cup \{p\}$ satisfies condition (1) and is a WI-DJ answer set of the program, denoted Π_1, iff for every assignment σ that does not set all X_i to 1, we have that $\neg E(\sigma(X), Y)$ is satisfiable, i.e., $\Phi_1 = \exists X \neq 1 \forall Y E(X, Y)$ evaluates to false; deciding this is Π_2^p-hard. Second, if we use the extended construction for QBFs $\exists Z \forall Y . E(X, Y, Z)$ from Remark (1) and set X void, then $I_2 = Y \cup Z \cup \hat{Z} \cup \{p\}$ is a WI-DJ answer set of the program, denoted Π_2, iff $\Phi_2 = \exists Z \forall Y . E(Y, Z)$ evaluates to true. It follows that $I = I_1 \cup I_2$ is a DI-WJ answer set of the program $\Pi = \Pi_1 \cup \Pi_2$ iff the QBFs Φ_1 and Φ_2 evaluate to false and true, respectively; hence, DI-WJ answer set checking D_2^p-hard.

(3) The above reduction works also for FLP answer sets as the base semantics: the formula $E(X, Y)$ can not be derived from the literals X_i resp. $\neg X_i$ chosen by *sel* for an assignment σ in the least fixpoint construction iff I_σ is not a minimal model of the reduct $\Pi_{sel}^{I_\sigma}$. □

Proof of Lemma 3. As

$$P_{sel}(\Pi, I) = \Pi_{sel}^I \cup \{sel(head(r), I) \leftarrow body(r) \mid r \in ground(\Pi) \text{ and } body(r) \text{ is not satisfied by } I\}$$

$(P_{sel}(\Pi, I))^I$ consists of $(\Pi_{sel}^I)^I$ plus some rules of the form $sel(head(r), I) \leftarrow B_1 \wedge \cdots \wedge B_m$, where every B_i is a ground atom and some B_i is not in I (thus the rule body is not satisfied by I). Therefore, for any $J \subset I$, for every rule r in $(P_{sel}(\Pi, I))^I \setminus (\Pi_{sel}^I)^I$, J does not satisfy $body(r)$.

($\Longrightarrow$) Let I be a minimal model of $(\Pi_{sel}^I)^I$ and assume towards a contradiction that I is not a minimal model of $(P_{sel}(\Pi, I))^I$. Let $J \subset I$ be a minimal model of $(P_{sel}(\Pi, I))^I$. Then J is also a model of $(\Pi_{sel}^I)^I$ since $(\Pi_{sel}^I)^I \subseteq (P_{sel}(\Pi, I))^I$, contradicting that I is a minimal model of $(\Pi_{sel}^I)^I$.

($\Longleftarrow$) Let I be a minimal model of $(P_{sel}(\Pi, I))^I$. Then I is a model of $(\Pi_{sel}^I)^I$. Assume towards a contradiction that I is not a minimal model of $(\Pi_{sel}^I)^I$. Let $J \subset I$ be a minimal model of $(\Pi_{sel}^I)^I$. As mentioned above, for every rule r in $(P_{sel}(\Pi, I))^I \setminus (\Pi_{sel}^I)^I$, J does not satisfy $body(r)$. This means that J is also a model of $(P_{sel}(\Pi, I))^I$, contradicting that I is a minimal model of $(P_{sel}(\Pi, I))^I$. □

Proof of Theorem 7. I is a GL-answer set of Π iff by Theorem 2, for every head selection function *sel*, I is an answer set of Π_{sel}^I under the GL_{nlp}-semantics iff for every head selection function *sel*, I is a minimal model of $(\Pi_{sel}^I)^I$ iff for every head selection function *sel*, by Lemma 3 I is a minimal model of $(P_{sel}(\Pi, I))^I$ iff I is an answer set of every normal derivative in $ND(\Pi, I)$ under the GL_{nlp}-semantics. □

More information on function complexity

The class $\mathrm{FP}^{\Sigma_i^p}[\log, \mathrm{wit}]$ consists of all multi-valued total functions f such that for every input x some value $y \in f(x)$ is computable by a deterministic Turing machine in polynomial time with at most $O(\log |x|)$ many queries to a witness oracle $\mathcal{O}$ in Σ_i^p [9]. The witness oracle informally decides a problem in Σ_i^p, but provides for an oracle query $x_\mathcal{O}$ in case the answer is "yes" in addition a string $y_\mathcal{O}$ of size polynomial in $|x_\mathcal{O}|$(a "witness" for the answer) that can be checked with a single call to a Π_{i-1}^p oracle; e.g., for a SAT oracle ($i = 1$), a natural witness would be a satisfying assignment to the input formula. Notably, different witnesses may lead to different output of the machine, which is given by the content of a designated output tape at the end of each run.[17]

Chen and Toda [13] introduced the class FNP//OptP[$\log n$], which consists of the partial multi-valued functions that can be solved by a non-deterministic Turing machine in polynomial time with logsize oracle advice $a(x)$ from an NP-optimization problem. More precisely, for an input x and $a(x)$, some value $y \in f(x)$ is nondeterministically computed by the machine, where $a(x)$ is the binary presentation of the maximum output of a metric Turing machine, i.e., a nondeterministic Turing machine that outputs an integer in each branch, on input x, where each integer is polynomial in the size of x (thus $|a(x)| = O(\log |x|)$ holds); for example, $a(x)$ may be the maximum size of a clique in a graph, or the maximum size of a model for a SAT instance. It can be shown that FNP//OptP[$\log n$] and $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$ coincide on total multi-valued functions.[18]

Chen and Toda [13] proved that computing some maximal model of a QBF $\exists Z . E(X, Z)$ (= X-MAX-MODEL) is complete for FNP//OptP[$\log n$], under the following polynomial-time reduction for (partial) multi-valued functions: f_1 reduces to f_2, if

[17] Importantly, the machine always accepts and produces some output; otherwise Σ_2^p-hard problems, e.g. finding models of QBF $\forall Y . E(X, Y)$, could be solved in $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$. Note also that verifying whether $y \in f(x)$ holds is in general Σ_2^p-complete for functions f in $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$ but "only" $\mathrm{P}^{\mathrm{NP}}[\log n]$-complete for functions f in FNP//OptP[$\log n$]; this is due to the possibly exponentially many witnesses for a query answer.

[18] Technically, $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$ requires that each input has some output. However, each partial function f in FNP//OptP[$\log n$] can be turned into a total function f_∇ such that $f_\nabla(x) = f(x)$ if $f(x)$ is defined and $f_\nabla(x) = \nabla$ otherwise, where ∇ is a fresh symbol; this function f_∇ is computable in $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$.

there are functions g_1, g_2 computable in polynomial time such that for every x, (a) $g_1(x)$ is an instance of f_2 and has some output, i.e., $f_2(g_1(x)) \neq \emptyset$, iff $f_1(x)$ has some output, and (b) every $y \in f_2(g_1(x))$ gives rise to some output $g_2(x, y) \in f_1(x)$. Note that without loss of generality, one can assume that $\exists Z.E(X, Z)$ has some model. In this setting, the problem is then also complete for $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$.

To provide more detail, membership of computing some maximal model of a QBF $\exists Z.E(X, Z)$ is in $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$ is established as follows: we do a binary search on $k \geq 0$, whether $\exists Z.E(X, Z)$ has some model σ of size at least k; to this end, $\exists Z.E(X, Z)$ and k are transformed into a respective SAT instance F_k that we ship to a SAT oracle. When we know the maximum size k^*, we can again ask the oracle for a witness of F_{k^*}, which is then the result. The membership of computing some maximal model of a QBF $\exists Z \forall Y.E(X, Y, Z)$ in $\mathrm{FP}^{\Sigma_2^p}[\log, \mathrm{wit}]$ can be established analogously, using a Σ_2^p oracle.

The $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$-hardness of computing some maximal model of a QBF $\exists Z.E(X, Z)$ can be established, using Chen and Toda's key ideas by a generic Turing machine encoding as follows. A possible run of the machine M with witness oracle $\mathcal{O}$ in NP on input x, *relative to hypothetical results* $a = a_1 a_2 \cdots a_{\log |x|}$ *of the answer* $a_i \in$ {*"yes"*, *"no"*} *to the i-th query* x_i, can be encoded to a SAT instance $E(M, x, a)$, where for $a_i =$ "yes" a proper witness y_i for x_i is characterized as a SAT instance E_{x_i} (this is possible by Cook's Theorem). If we view a as a number in binary, where "yes" $= 1$ and "no" $= 0$, then any model of $E(M, x, a)$ where a is maximum reflects a proper run of M on x, as no $a_i =$ "no" can be flipped to $a_i =$ "yes" to obtain a larger value.[19]

We couch this now into computing some maximal model of a QBF $\exists Y.E(X, Y)$ by introducing variables $X = \{X_1, \ldots, X_n\}$ where $n = 2^{\log |x|}$ and expressing in E that for some i, $X_1, \ldots, X_i$ must be true and $X_{i+1}, \ldots, X_n$ must be false; intuitively, i expresses a number $a^{(i)} = a_1^{(i)} \cdots a_{\log |x|}^{(i)}$ in tally notation. In E we express that if $X_i \wedge \neg X_{i+1}$ is true, then $a = a^{(i)}$ holds; the rest of E contains $E(M, x, a)$, and we let Y be all variables not in X. The maximal models of $\exists Y.E(X, Y)$ then correspond to proper runs of M on x (but not every proper run might be covered). However, at this point we miss the output y computed by M on x. In order to bring it in, we can duplicate each X_j to a block $X_{j,1}, \ldots, X_{j,m}$ of variables, where m is large enough, and request that they are all true if $j < i$, all false if $j > i$ and for $j = i$ encode y (in binary). Each maximal model of the adapted QBF $\exists Y.E(X, Y)$ encodes then a proper run of M on x with some output y. In order to get all outputs for the answers $a(x)$, we may add the set complement $\overline{y}$ to X; this will ensure that different outputs y will be incomparable under model maximization.

Notably, the restriction of X-MAX-MODEL to *cardinality*-maximal (i.e., largest size) models of a QBF $\exists Z.E(X, Z)$ is also complete for $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$ and $\mathrm{FNP}//\mathrm{OptP}[\log n]$; the hardness part follows by the complement addition $\overline{y}$, as then cardinality- and subset-maximal models coincide. In fact, the restriction is hard even in absence of $\exists Z$, i.e., for SAT instances $E(X)$; this can be shown by eliminating $\exists Z$ from $\exists Z.E(X, Z)$ with further variable duplication, such that the block for each variable in X is larger than Z.[20]

The $\mathrm{FP}^{\Sigma_2^p}[\log, \mathrm{wit}]$-hardness of computing some maximal model of a QBF $\exists Z \forall Y.E(X, Y, Z)$ can be shown similarly, where the witness oracle calls are modeled as QBFs $\exists X_i \forall Y_i.E(X_i, Y_i)$ and the resulting formula is transformed into the desired form.

Computing minimal models of a QBF $\exists Z.E(X, Z)$ (resp., $\exists Z \forall Y.E(X, Y, Z)$) is trivially polynomial-time equivalent to computing maximal models of such a QBF.

It thus follows from Remark (1) of proof of Theorem 6 that computing some DI-WJ answer set of a general disjunctive program Π is $\mathrm{FP}^{\Sigma_2^p}[\log, \mathrm{wit}]$-hard. The construction there can be adapted for QBFs $\exists Z.E(X, Z)$, where $E(X, Z)$ is in CNF, to simple disjunctive programs, which establishes hardness for $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$.

References

[1] M. Alviano, W. Faber, Aggregates in answer set programming, KI 32 (2–3) (2018) 119–124, https://doi.org/10.1007/s13218-018-0545-9.
[2] G. Amendola, T. Eiter, M. Fink, N. Leone, J. Moura, Semi-equilibrium models for paracoherent answer set programs, Artif. Intell. 234 (2016) 219–271.
[3] V. Asuncion, Y. Chen, Y. Zhang, Y. Zhou, Ordered completion for logic programs with aggregates, Artif. Intell. 224 (2015) 72–102.
[4] M. Balduccini, M. Gelfond, Logic programs with consistency-restoring rules, in: J. McCarthy, M.-A. Williams (Eds.), International Symposium on Logical Formalization of Commonsense Reasoning, in: AAAI 2003 Spring Symposium Series, 2003, pp. 9–18.
[5] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving with Answer Sets, Cambridge University Press, 2003.
[6] M. Bartholomew, J. Lee, Y. Meng, First-order extension of the FLP stable model semantics via modified circumscription, in: Proc. 22nd Int. Joint Conference on Artificial Intelligence (IJCAI-11), 2011, pp. 724–730.
[7] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (12) (2011) 92–103.
[8] G. Brewka, T. Eiter, M. Truszczyński (Eds.), Special Issue on Answer Set Programming, AI Mag. 37 (3) (2016).
[9] S. Buss, J. Krajíček, G. Takeuti, On provably total functions in bounded arithmetic theories, in: P. Clote, J. Krajíček (Eds.), Arithmetic, Proof Theory and Computational Complexity, Oxford University Press, 1993, pp. 116–161.
[10] M. Cadoli, T. Eiter, G. Gottlob, Default logic as a query language, IEEE Trans. Knowl. Data Eng. 9 (3) (1997) 448–463.
[11] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, T. Schaub, ASP-Core-2: input language format, https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.01c.pdf, 2012.
[12] E. Chan, A possible worlds semantics for disjunctive databases, IEEE Trans. Knowl. Data Eng. 5 (2) (1993) 282–292.
[13] Z.-Z. Chen, S. Toda, The complexity of selecting maximal solutions, Inf. Comput. 119 (1995) 231–239.

[19] Note that here it is important that each run of the machine produces some output.
[20] Whether subset-minimal X-MAX-MODEL for $E(X)$ is $\mathrm{FP}^{\mathrm{NP}}[\log, \mathrm{wit}]$-hard is unknown.

[14] M. Dao-Tran, T. Eiter, M. Fink, T. Krennwallner, Modular nonmonotonic logic programming revisited, in: P.M. Hill, D.S. Warren (Eds.), Logic Programming, 25th International Conference, ICLP 2009, Proceedings, Pasadena, CA, USA, July 14–17, 2009, in: Lecture Notes in Computer Science, vol. 5649, Springer, 2009, pp. 145–159.
[15] J. de Bruijn, T. Eiter, A. Polleres, H. Tompits, Embedding non-ground logic programs into autoepistemic logic for knowledge base combination, ACM Trans. Comput. Log. 12 (3) (2011), Article 20.
[16] J. de Bruijn, T. Eiter, H. Tompits, Embedding approaches to combining rules and ontologies into autoepistemic logic, in: Proc. 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), AAAI Press, 2008, pp. 485–495.
[17] H. Decker, J.C. Casamayor, Sustained models and sustained answers in first-order databases, in: M. Alpuente, et al. (Eds.), Proceedings of the 1994 Joint Conference on Declarative Programming (GULP-PRODE'94), vol. 2, Valencia, Spain, Servicio de Publicaciones, Univ. Politéc. Valencia, 1994, pp. 32–46. Preliminary version (invited paper) in Proc. DAISD 1993, pp. 267–286.
[18] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: propositional case, Ann. Math. Artif. Intell. 15 (3–4) (1995) 289–323.
[19] T. Eiter, G. Ianni, T. Krennwallner, Answer set programming: a primer, in: S. Tessaris, E. Franconi, T. Eiter, C. Gutiérrez, S. Handschuh, M. Rousset, R.A. Schmidt (Eds.), Reasoning Web. Semantic Technologies for Information Systems, Tutorial Lectures, 5th International Summer School 2009, Brixen–Bressanone, Italy, August 30–September 4, 2009, in: Lecture Notes in Computer Science, vol. 5689, Springer, 2009, pp. 40–110.
[20] 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.
[21] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, A uniform integration of higher-order reasoning and external evaluations in answer-set programming, in: L.P. Kaelbling, A. Saffiotti (Eds.), Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI-05, Edinburgh, Scotland, UK, July 30–August 5, 2005, Professional Book Center, 2005, pp. 90–96.
[22] U. Endriss, R. de Haan, Complexity of the winner determination problem in judgment aggregation: Kemeny, Slater, Tideman, Young, in: G. Weiss, P. Yolum, R.H. Bordini, E. Elkind (Eds.), Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4–8, 2015, ACM, 2015, pp. 117–125.
[23] W. Faber, N. Leone, G. Pfeifer, Recursive aggregates in disjunctive logic programs: semantics and complexity, in: Logics in Artificial Intelligence: European Workshop, 2004, pp. 200–212.
[24] W. Faber, G. Pfeifer, N. Leone, Semantics and complexity of recursive aggregates in answer set programming, Artif. Intell. 175 (1) (2011) 278–298.
[25] F. Fages, Consistency of Clark's completion and existence of stable models, Methods Logic Comput. Sci. 1 (1994) 51–60.
[26] P. Ferraris, Answer sets for propositional theories, in: C. Baral, G. Greco, N. Leone, G. Terracina (Eds.), Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LPNMR 2005, Proceedings, Diamante, Italy, September 5–8, 2005, in: Lecture Notes in Computer Science, vol. 3662, Springer, 2005, pp. 119–131.
[27] P. Ferraris, J. Lee, V. Lifschitz, Stable models and circumscription, Artif. Intell. 175 (1) (2011) 236–263.
[28] P. Ferraris, V. Lifschitz, Mathematical foundations of answer set programming, in: S.N. Artëmov, H. Barringer, A.S. d'Avila Garcez, L.C. Lamb, J. Woods (Eds.), We Will Show Them! Essays in Honour of Dov Gabbay, vol. 1, College Publications, 2005, pp. 615–664.
[29] M. Gelfond, On stratified autoepistemic theories, in: K.D. Forbus, H.E. Shrobe (Eds.), Proceedings of the 6th National Conference on Artificial Intelligence, Seattle, WA, USA, July 1987, Morgan Kaufmann, 1987, pp. 207–211, http://www.aaai.org/Library/AAAI/1987/aaai87-037.php.
[30] M. Gelfond, Answer sets, in: F. van Harmelen, V. Lifschitz, B.W. Porter (Eds.), Handbook of Knowledge Representation, in: Foundations of Artificial Intelligence, vol. 3, Elsevier, 2008, pp. 285–316.
[31] M. Gelfond, New semantics for epistemic specifications, in: Logic Programming and Nonmonotonic Reasoning – 11th International Conference LPNMR, 2011, pp. 260–265.
[32] M. Gelfond, Y. Kahl, Knowledge Representation, Reasoning, and the Design of Intelligent Agents, Cambridge University Press, 2014.
[33] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: Logic Programming, Proceedings of the Fifth International Conference and Symposium, 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] M. Gelfond, Y. Zhang, Vicious circle principle and formation of sets in ASP based languages, in: 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-2017), 2017, pp. 146–159.
[36] P. Hitzler, A.K. Seda, Multivalued mappings, fixed-point theorems and disjunctive databases, in: Proceedings of the 3rd Irish Conference on Formal Methods, Galway, Eire, British Computer Society, 1999, pp. 113–131.
[37] T. Janhunen, I. Niemelä, The answer set programming paradigm, AI Mag. 37 (3) (2016) 13–24.
[38] M. Janota, J. Marques-Silva, On the query complexity of selecting minimal sets for monotone predicates, Artif. Intell. 233 (2016) 73–83.
[39] P.T. Kahl, Refining the Semantics for Epistemic Logic Programs, Ph.D. thesis. Texas Tech University, USA, 2014.
[40] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, F. Scarcello, The DLV system for knowledge representation and reasoning, ACM Trans. Comput. Log. 7 (3) (Jul. 2006) 499–562, https://doi.org/10.1145/1149114.1149117.
[41] N. Leone, P. Rullo, F. Scarcello, Disjunctive stable models: unfounded sets, fixpoint semantics and computation, Inf. Comput. 135 (2) (June 1997) 69–112.
[42] V. Lifschitz, Action languages, answer sets and planning, in: The Logic Programming Paradigm: A 25-Year Perspective, Springer, 1999, pp. 357–373.
[43] V. Lifschitz, Answer set programming and plan generation, Artif. Intell. 138 (1–2) (2002) 39–54.
[44] V. Lifschitz, Twelve definitions of a stable model, in: M.G. de la Banda, E. Pontelli (Eds.), Logic Programming, 24th International Conference, ICLP 2008, Proceedings, Udine, Italy, December 9–13, 2008, in: Lecture Notes in Computer Science, vol. 5366, Springer, 2008, pp. 37–51.
[45] V. Lifschitz, L.R. Tang, H. Turner, Nested expressions in logic programs, Ann. Math. Artif. Intell. 25 (1–2) (1999) 369–389.
[46] J. Lobo, J. Minker, A. Rajasekar, Foundations of Disjunctive Logic Programming, MIT Press, Cambridge, MA, 1992.
[47] 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.
[48] V.W. Marek, M. Truszczynski, Reflective autoepistemic logic and logic programming, in: L.M. Pereira, A. Nerode (Eds.), Logic Programming and Nonmonotonic Reasoning, Proceedings of the Second International Workshop, Lisbon, Portugal, June 1993, MIT Press, 1993, pp. 115–131.
[49] V.W. Marek, M. Truszczynski, Stable models and an alternative logic programming paradigm, in: The Logic Programming Paradigm: A 25-Year Perspective, Springer, 1999, pp. 375–398.
[50] J. Minker, On indefinite databases and the closed world assumption, in: D.W. Loveland (Ed.), 6th Conference on Automated Deduction, Proceedings, New York, USA, June 7–9, 1982, in: Lecture Notes in Computer Science, vol. 138, Springer, 1982, pp. 292–308.
[51] R.C. Moore, Semantical considerations on nonmonotonic logic, Artif. Intell. 25 (1) (1985) 75–94.
[52] B. Motik, R. Rosati, Reconciling description logics and rules, J. ACM 57 (5) (2010).
[53] I. Niemela, Logic programs with stable model semantics as a constraint programming paradigm, Ann. Math. Artif. Intell. 25 (1999) 241–273.
[54] D. Pearce, A new logical characterisation of stable models and answer sets, in: J. Dix, L.M. Pereira, T.C. Przymusinski (Eds.), Non-Monotonic Extensions of Logic Programming, NMELP '96, Selected Papers, Bad Honnef, Germany, September 5–6, 1996, in: Lecture Notes in Computer Science, vol. 1216, Springer, 1996, pp. 57–70.
[55] D. Pearce, Equilibrium logic, Ann. Math. Artif. Intell. 47 (1–2) (2006) 3–41.
[56] N. Pelov, Semantics of Logic Programs with Aggregates, Ph.D. thesis. Katholieke Universiteit Leuven, 2004, http://www.cs.kuleuven.ac.be/publicaties/doctoraten/cw/cw2004_02.abs.html.

[57] 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.
[58] T.C. Przymusinski, Stable semantics for disjunctive programs, New Gener. Comput. 9 (3–4) (1991) 401–424.
[59] R. Reiter, A logic for default reasoning, Artif. Intell. 13 (1980) 81–132.
[60] C. Sakama, Possible model semantics for disjunctive databases, in: Proceedings First Int. Conf. on Deductive and Object-Oriented Databases (DOOD-89), North-Holland, Kyoto, Japan, 1989, pp. 369–383.
[61] C. Sakama, K. Inoue, Negation in disjunctive logic programs, in: D.S. Warren (Ed.), Logic Programming, Proceedings of the Tenth International Conference on Logic Programming, Budapest, Hungary, June 21–25, 1993, MIT Press, 1993, pp. 703–719.
[62] C. Sakama, K. Inoue, Paraconsistent stable semantics for extended disjunctive programs, J. Log. Comput. 5 (3) (1995) 265–285.
[63] T. Schaub, S. Woltran, Special issue on answer set programming, KI 32 (2–3) (2018) 101–103, https://doi.org/10.1007/s13218-018-0554-8.
[64] Y.D. Shen, T. Eiter, Evaluating epistemic negation in answer set programming, Artif. Intell. 237 (2016) 115–135.
[65] Y.D. Shen, K. Wang, T. Eiter, M. Fink, C. Redl, T. Krennwallner, J. Deng, FLP answer set semantics without circular justifications for general logic programs, Artif. Intell. 213 (2014) 1–41.
[66] Y.D. Shen, J.H. You, A generalized Gelfond-Lifschitz transformation for logic programs with abstract constraints, in: AAAI, 2007, pp. 483–488.
[67] P. Simons, I. Niemela, T. Soininen, Extending and implementing the stable model semantics, Artif. Intell. 138 (1–2) (2002) 181–234.
[68] 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.
[69] M. Truszczynski, Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs, Artif. Intell. 174 (16–17) (2010) 1285–1306.
[70] M. Truszczynski, Revisiting epistemic specifications, in: Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning – Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday, 2011, pp. 315–333.
[71] M.H. van Emden, R.A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (4) (1976) 733–742.
[72] A. van Gelder, K. Ross, J. Schlipf, The well-founded semantics for general logic programs, J. ACM 38 (3) (1991) 620–650.