# Well-Supported Semantics for Logic Programs with Generalized Rules

Jia-Huai You[1], Yi-Dong Shen[2], Kewen Wang[3]

[1] Department of Computing Science, University of Alberta, Canada
[2] State Key Laboratory of Computer Science, Chinese Academy of Sciences, China
[3] School of Computing and Information Technology, Griffith University, Australia

**Abstract.** Logic programming under the stable model semantics has been extended to arbitrary formulas. A question of interest is how to characterize the property of well-supportedness, in the sense of Fages, which has been considered a cornerstone in answer set programming. In this paper, we address this issue by considering *general logic programs*, which consist of disjunctive rules with arbitrary propositional formulas in rule bodies. We define the justified stable semantics for these programs, propose a general notion of well-supportedness, and show the relationships between the two. We address the issue of computational complexity for various classes of general programs. Finally, we show that previously proposed well-supported semantics for aggregate programs and description logic programs are rooted in the justified stable semantics of general programs.

## 1 Introduction

Logic programs under the stable model semantics has been extended to general forms of formulas, including nested expressions [12, 17], general stable models [13], and the FLP-semantics for various kinds of logic programs [2, 8, 10, 26]. These extensions are in part motivated by the need to provide a semantic account for logic programs with aggregates, and by supporting external atoms in combination of answer set programming (ASP) with other reasoning formalisms.

For normal programs, the notion of well-supportedness is well understood [11], which can be informally described as non-circular justification of atoms in answer sets. Intuitively, this means that no atoms in an answer set may solely depend on the truth of themselves in derivations by rules. However, when more general forms of formulas are introduced, question arises as what could be a suitable notion of well-supportedness. This has contributed to the question what exactly a phrase like *self-supporting loop* means. Recall that latter was introduced in [7] to describe seemingly a counterintuitive behavior of weak answer sets for description logic programs (i.e., dl-programs).

The problem is non-trivial when disjunction is involved. One difficulty is, when disjunction appears in the body of a rule, it is not clear how to capture the notion of dependency. Another difficulty is due to the uniform treatment of logic implication in arbitrary formulas: when a formula contains several implications, we may have to view the formula as one with nested rules in it. An interesting alternative is recently proposed [2], where rules are of the form $H \leftarrow F$, with $H$ and $F$ being arbitrary (first-order) formulas. For the semantics, circumscription is applied in such a way that only the implication between $H$ and $F$ is strengthened to behave like a rule.

In this paper, we consider the class of propositional logic programs, called *general programs*, in which a rule has a disjunction of atoms in the head and an arbitrary formula in the body. We define a semantics for these programs by formulating the notion of *justified stable models*, propose a notion of well-supportedness, and address the issue of computational complexity for various classes of general programs. We show that the justified stable semantics for general programs provides a uniform representation framework for the family of well-supported semantics currently known in the literature. It is capable of representing the well-supported semantics for aggregate programs, known as the *ultimate stable model semantics* [19], or alternatively as the answer set semantics based on *conditional satisfaction* [24]. This was shown in [23]. In this paper, we further show that the justified stable semantics for general programs is capable of representing the well-supported semantics of dl-programs, recently proposed in [22].

The paper is organized as follows. Following the preliminaries, we define the justified stable semantics for general programs and propose a general notion of well-supportedness. Then we present the complexity results, and discuss related work. This is followed by a sketch of how well-supported semantics for dl-programs may be captured by justified stable models of general programs. The paper is concluded by final remarks and questions for further investigation.

## 2 Preliminary

We restrict attention to a propositional language, as we consider only Herbrand interpretations so that the first-order case reduces to a propositional one via grounding. We assume a propositional language, $\mathcal{L}_\Sigma$, determined by a fixed countable set $\Sigma$ of propositional *atoms*.

*Formulas* are built from atoms using standard (classical) connectives $\neg$, $\wedge$, $\vee$, $\supset$, and $\equiv$. A *literal* is an atom or a negated atom. A *theory* is a set of formulas.

A *rule* $r$ is of the form: $\alpha_1; \ldots; \alpha_k \leftarrow F$, where $k \geq 1$ and each $\alpha_i$ is an atom and $F$ a formula. We use $head(r)$ and $body(r)$ to refer to the head set $\{\alpha_1, \ldots, \alpha_k\}$ and the body formula $F$ of $r$, respectively.

A *general program* (or *program*) $\Pi$ is a set of rules. $\Pi$ is called *non-disjunctive* if the head of every rule in $\Pi$ consists of a single atom.

A *disjunctive (logic) program* is a program in which each rule body is a conjunction of literals. A *normal program* is a disjunctive program whose rule heads consist of a single atom. Traditionally in normal and disjunctive programs we use default negation $not$, but in general programs we replace it with $\neg$.

An *interpretation* is a subset of $\Sigma$. We say that an interpretation $I$ *satisfies* an atom $\alpha$ if $\alpha \in I$; $\neg\alpha$ if $\alpha \notin I$. The satisfaction of a formula $F$ by an interpretation $I$ is defined as in propositional logic. $I$ satisfies a rule $r$ if it satisfies $head(r)$ or it does not satisfy $body(r)$. $I$ is a *model* of a program $\Pi$ if it satisfies all rules in $\Pi$. A *minimal model* is a model none of whose proper subsets is also a model. For any expression $E$, we say $E$ is *true* (resp. *false*) in $I$ if and only if $I$ satisfies (resp. does not satisfy) $E$.

For an interpretation $I$, let $I^- = \{\neg a \mid a \in \Sigma \setminus I\}$.

For any two theories $F$ and $G$, $F$ *entails* $G$, denoted $F \models G$, if $G$ is true in all models of $F$. Note that an interpretation $I$ satisfying $F$ is different from $I$ entailing $F$

(i.e., $I \models F$ where $I$ is treated as the conjunction of atoms in $I$). The former amounts to the entailment $I \cup I^- \models F$.

Given a disjunctive program $\Pi$ and an interpretation $I$, the standard *Gelfond-Lifschitz transformation* of $\Pi$ w.r.t. $I$, written as $\Pi^I$, is obtained from $\Pi$ by performing two operations: (1) remove from $\Pi$ all rules whose bodies contain a negative literal $\neg a$ with $a \in I$, and (2) remove from the remaining rules all negative literals. The *standard ASP semantics* defines $I$ to be a stable model (also called an answer set) of $\Pi$ if it is a minimal model of $\Pi^I$ [14].

## 3 Logic Programs with Generalized Rules

We define a notion of *deductive closure*.

**Definition 1.** *Let $\Pi$ be a program and $I$ an interpretation. A* deductive closure *of $\Pi$ and $I$, denoted as $\Omega(\Pi, I)$, is a minimal set of atoms satisfying the condition: for any rule $\alpha_1; \ldots; \alpha_k \leftarrow F \in \Pi$, if $\Omega(\Pi, I) \cup I^- \models F$, then for some $\alpha_i$ $(1 \leq i \leq k)$, $\alpha_i \in \Omega(\Pi, I)$.*

Note that $I^-$ above is fixed in determining if $\Omega(\Pi, I)$ is minimal. That is, let $S = \Omega(\Pi, I)$. That $S$ is minimal satisfying the stated condition means there exists no $S' \subset S$ such that for any $r \in \Pi$, if $S' \cup I^- \models body(r)$ then $\exists \alpha \in head(r)$ such that $\alpha \in S'$.

**Definition 2.** *Let $\Pi$ be a program and $I$ an interpretation. $I$ is a* justified stable model *of $\Pi$ iff for some deductive closure $\Omega(\Pi, I)$, $I = \Omega(\Pi, I)$.*

Intuitively, a justified stable model is a minimal set of atoms satisfying the closure property where the entailed body of a rule derives at least one head atom.

*Example 1.* Consider the following disjunctive program, $\Pi_1$:

$$a; b \leftarrow \qquad a \leftarrow b \qquad d \leftarrow \neg b$$

Under the standard ASP semantics, the only stable model of $\Pi_1$ is $I_1 = \{a, d\}$, which is also the unique justified stable model of $\Pi_1$. Note that $I_2 = \{a, b\}$ is not a justified stable model of $\Pi_1$, since it is not a minimal set satisfying the stated condition in Definition 1, as the condition is satisfied by a proper subset, $\{a\}$.

*Example 2.* As an example beyond the class of disjunctive programs, consider $\Pi_2$:

$$a \leftarrow b \qquad b \leftarrow \neg b \vee a$$

The only model of $\Pi_2$ is $I = \{a, b\}$, which is a minimal model of $\Pi_2$, but not a justified stable model of $\Pi_2$. In terms of Definition 1, $I$ is not a minimal set satisfying the stated condition, as the empty set satisfies the condition. This shows that not all minimal models are justified in the sense of Definition 2.

With a clear understanding that $I^-$ is fixed in the minimization of a set, we can combine Definitions 1 and 2 to arrive at an even simpler yet equivalent definition.

**Definition 3.** *Given a program $\Pi$ and an interpretation $I$, $I$ is a justified stable model of $\Pi$ iff $I$ is a minimal set satisfying the condition, for any $r \in \Pi$, if $I \cup I^- \models body(r)$ then $\exists \alpha \in head(r)$, $\alpha \in I$.*

The justified stable semantics extends the stable model semantics for disjunctive logic programs, as it is easy to check that in this case the definition coincides with the standard one.

**Theorem 1.** *For a disjunctive logic program $\Pi$, an interpretation $I$ is a justified stable model of $\Pi$ iff $I$ is a stable model of $\Pi$ under the standard ASP semantics.*

It is also easy to show that justified stable models are minimal models.

**Proposition 1.** *A justified stable model of a program $\Pi$ is a minimal model of $\Pi$.*

The justified stable semantics also extends the *progression semantics*, from disjunctive logic programs [29] to general programs, for the propositional case. That is, there is an iterative definition of justified stable model, which extends the concept of *progression*, based on expansions by *hitting sets*.

**Definition 4.** *Let $S$ be a set of atoms and $\xi = \{S_1, ..., S_i, ...\}$ a collection of sets of atoms such that $S_i \subseteq S$, for all $i$. A subset $H \subseteq S$ is said to be a* hitting set *of $\xi$ if for all $i$, $H \cap S_i \neq \emptyset$. Furthermore, $H$ is said to be a* minimal hitting set *of $\xi$ if $H$ is a hitting set of $\xi$ and there is no $H' \subset H$ such that $H'$ is also a hitting set.*

**Definition 5.** *Let $\Pi$ be a program and $I$ a model of $\Pi$. An* evolution sequence of $\Pi$ based on $I$, *is a sequence of sets of atoms $\sigma_I^0(\Pi), \sigma_I^1(\Pi), ..., \sigma_I^k(\Pi), ...$, denoted as $\sigma_I(\Pi)$ (or simply as $\sigma$ when the context is clear), such that*

1. *$\sigma_I^0(\Pi) = \emptyset$, and*
2. *For $i \geq 0$, $\sigma_I^{i+1}(\Pi) = \sigma_I^i(\Pi) \cup H_i$, if there exists $H_i \subseteq I$ such that $H_i$ is a minimal hitting set of the sets,*

$$head(r),$$

*where $r \in \Pi$, $head(r) \cap \sigma_I^i(\Pi) = \emptyset$, and $\sigma_I^i(\Pi) \cup I^- \models body(r)$; otherwise $\sigma_I^{i+1}(\Pi) = \sigma_I^i(\Pi)$.*

As commented in [29], the basic idea in an evolution sequence is that we start with the fixed $I^-$ and progress by nondeterministically selecting a minimal hitting set at each step, from heads of the rules whose bodies are logically entailed by $I^-$ and the set of atoms already derived. The condition $H_i \subseteq I$ ensures that the construction is guarded, and the condition $head(r) \cap \sigma_I^i(\Pi) = \emptyset$ only allows the rule heads that are not already satisfied to be considered.

**Lemma 1.** *Let $\Pi$ be a program and $I$ a model of $\Pi$.*

*(i) For any evolution sequence $\rho$, $\rho_I^\infty(\Pi) \subseteq I$.*
*(ii) Any set of atoms in any evolution sequence of $\Pi$ based on $I$ is consistent.*

The interest in the following theorem is two-fold. On the one hand, it shows a constructive definition of justified stable models, and on the other hand, along with Theorem 1, it shows that for propositional disjunctive programs, the progression semantics [29] has a much simpler definition (i.e., Definition 2), without the need of resorting to the concept of evolution sequences. In addition, as we will see shortly, an evolution sequence serves a witness for justified stable models to be well-supported, in a native manner.

**Theorem 2.** *Let $\Pi$ be a program and $I$ a model of $\Pi$. $I$ is a justified stable model of $\Pi$ iff there exists at least one evolution sequence of $\Pi$ based on $I$, and for all evolution sequences $\sigma_I(\Pi)$, $\sigma_I^\infty(\Pi) = I$.*

*Proof.* (Sketch) ($\Rightarrow$) Let $I$ be a justified stable model of $\Pi$. Then, there is a deductive closure $\Omega(\Pi, I)$ such that $I = \Omega(\Pi, I)$. It can be shown that $\Omega(\Pi, I)$ induces an evolution sequence $\sigma_I(\Pi)$ such that $\Omega(\Pi, I) = \sigma_I^\infty(\Pi)$. That is, there exists an evolution sequence $\sigma$ such that $I = \sigma_I^\infty(\Pi)$. We show that this is the case for all evolution sequences of $\Pi$ based on $I$. Now suppose there is some evolution sequence $\delta$ such that $\delta_I^\infty(\Pi) \neq I$. It follows from Lemma 1 that $\delta_I^\infty(\Pi) \subset I$. By construction, $\delta_I^\infty(\Pi)$ satisfies the condition in Definition 1, i.e., for any rule $r \in \Pi$ such that $\delta_I^\infty(\Pi) \cup I^- \models body(r)$, for some $\alpha \in head(r)$, $\alpha \in \delta_I^\infty(\Pi)$. As $\delta_I^\infty(\Pi) \subset I$ and $I = \Omega(\Pi, I)$, we have $\delta_I^\infty(\Pi) \subset \Omega(\Pi, I)$. Thus, $\Omega(\Pi, I)$ is not a minimal set satisfying the stated condition, as $\delta_I^\infty(\Pi)$ also satisfies the condition. Contradiction. Hence the evolution sequence $\delta$ does not exist. Thus, for all evolution sequences $\sigma$ of $\Pi$ based on $I$, we have $\sigma_I^\infty(\Pi) = I$.

($\Leftarrow$) Assume there is some evolution sequence $\sigma$ such that $\sigma_I^\infty(\Pi) = I$, and this is the case for all evolution sequences of $\Pi$ based on $I$. Since $\sigma$ is consistent, all rules in $\Pi$ are satisfied by $I$, hence $I$ is a model of $\Pi$, Towards a contradiction, assume $I$ is not a justified stable model of $\Pi$. Then, for all deductive closure $\Omega(\Pi, I)$, $\Omega(\Pi, I) \neq I$. If for every deductive closure $\Omega(\Pi, I)$ there exists an atom $a \notin I$ and $a \in \Omega(\Pi, I)$, then some rule in $\Pi$ is not satisfied by $I$. This is not possible since $I$ is a model of $\Pi$. It follows that for some deductive closure $\Omega(\Pi, I)$, we have $\Omega(\Pi, I) \subset I$, for which there is a corresponding evolution sequence $\rho_I(\Pi)$ such that $\rho_I^\infty(\Pi) = \Omega(\Pi, I)$. Since $\Omega(\Pi, I) \neq I$, it follows $\rho_I^\infty(\Pi) \neq I$. Contradiction.

### 3.1 A generalized notion of well-supportedness

The notion of well-supportedness is introduced in [11]: For a normal program $\Pi$, an interpretation $I$ is *well-supported* if there exists a strict well-founded partial order $\prec$ on $I$ such that for any $a \in I$, there is a rule $a \leftarrow body(r)$ in $\Pi$ such that $I$ satisfies $body(r)$ and for every positive literal $b$ in $body(r)$, $b \prec a$. A binary relation $\prec$ is *well-founded* if there is no infinite decreasing chain $a_0 \succ a_1 \succ \cdots$. Intuitively, a well-supported interpretation $I$ guarantees that every positive conclusion in $I$ has a non-circular justification, i.e., $I$ is free of self-supporting loops.

There are two problems when attempting to apply this definition to general programs. Firstly, the definition does not handle disjunctive rules. Secondly, when the body of a rule is an arbitrary formula, the condition "$I$ satisfies $body(r)$ and for every positive literal $b$ in $body(r)$, $b \prec a$" is no longer applicable. For example, for the program

$\{a \leftarrow a \vee b,\ b \leftarrow\}$, the interpretation $\{a, b\}$ is a justified stable model, but a blind application would require $a \prec a$ to be in a partial order, which cannot be well-founded.

We propose a generalized notion of well-supportedness.

**Definition 6.** *Let $\Pi$ be a program and $I$ an interpretation. $I$ is* well-supported *iff there exists a strict well-founded partial order $\prec$ on $I$ such that for any atom $a \in I$, there exist a rule $a_1; \ldots; a; \cdots; a_k \leftarrow body(r) \in \Pi$ and a subset $S \subset I$ such that $S \cup I^- \models body(r)$ and for every $b \in S$, $b \prec a$.*

Following the literature, we call such a well-founded partial order a *level mapping*. As for any justified stable model there is at least one corresponding evolution sequence, which serves as a witness of well-supportedness, we have

**Proposition 2.** *Any justified stable model of a general program is well-supported.*

Recall that according to [11], the stable models of a normal program $P$ are precisely well-supported models of $P$. This can easily be extended to non-disjunctive general programs under the generalized notion of well-supportedness, by a routine proof.

**Proposition 3.** *Let $\Pi$ be a non-disjunctive program and $I$ an interpretation. $I$ is a justified stable model of $\Pi$ iff $I$ is a model of $\Pi$ and is well-supported.*

The iff statement in this proposition is important. It answers the question, given a non-disjunctive program, what if a model (which could be a stable model defined in a different way) is not a justified stable model - then we know that it does not possess a level mapping as defined in Definition 6, and therefore necessarily incurs the phenomenon that some atoms in it depend on the truth of themselves.

From now on, we can say that a model $M$ of a non-disjunctive program is *self-supporting*, or $M$ has *self-supporting loops*, if and only if $M$ is not well-supported. Due to Proposition 3, this is a formal statement.

## 4 Complexity

In this section, we address the computational complexity of the justified stable semantics for general programs. The following three problems will be considered:

- *Consistency*: The problem of deciding whether a justified stable model exists.
- *Credulous reasoning*: The problem of deciding whether an atom is in a justified stable model.
- *Skeptical reasoning*: The problem of deciding whether an atom is in all justified stable models.

Recall that $\Delta_2^p = P^{NP}$ is the class of problems that are *deterministically* decidable in polynomial time with the help of an NP oracle; $\Sigma_2^p = NP^{NP}$ is the class of problems that are *nondeterministically* decidable in polynomial time with the help of an NP oracle; $\Pi_2^p =$ co-$NP^{NP}$ is the class of problems such that the complementary problem is in $\Sigma_2^p$.

Our main complexity results are summarized in Table 1.

We need to define the class of basic programs, which are motivated from representing aggregate programs by non-disjunctive general programs [23]. In this paper, for notational convenience, we assume that an aggregate is a constraint whose semantics can be defined by an *abstract constraint* (or a *c-atom*) of the form $(D, S)$, where $D$ is a set of atoms serving as the *domain* and $S$ is a set of subsets of $D$ representing *admissible solutions* of the aggregate.

**Definition 7.** *Let $A = (D, S)$ be a c-atom, where $S = \{\phi_1, ..., \phi_m\}$, $\phi_i = \{a_1, ..., a_k\}$, and $D \setminus \phi_i = \{b_1, ..., b_l\}$. Define $\tau(A)$ to be the formula: $\tau(A) = false$ when $m = 0$, otherwise $\tau(A) = C_1 \vee ... \vee C_m$, where each $C_i$ is the conjunction $a_1 \wedge ... \wedge a_k \wedge \neg b_1 \wedge ... \wedge \neg b_l$.*

That is, $\tau(A)$ is a special type of DNF. Let us call this type of DNFs *complete DNFs*.

**Definition 8.** *A* basic program *is a non-disjunctive general program consisting of rules whose bodies are a conjunction of complete DNFs and their negations.*

From Table 1, it is interesting to observe that the complexity for general programs is on the same level of the polynomial hierarchy as the complexity for non-disjunctive programs which is also on the same level as the complexity for disjunctive logic programs. However, for the class of basic programs, which is a subclass of non-disjunctive general programs, the complexity drops to the first level.

| program $\Pi$ | basic | disjunctive | non-disjunctive | general |
|---|---|---|---|---|
| Consistency | NP | $\Sigma_2^p$ | $\Sigma_2^p$ | $\Sigma_2^p$ |
| Credulous | NP | $\Sigma_2^p$ | $\Sigma_2^p$ | $\Sigma_2^p$ |
| Skeptical | co-NP | $\Pi_2^p$ | $\Pi_2^p$ | $\Pi_2^p$ |

**Table 1.** Complexity of justified stable semantics (entries are completeness results)

For basic programs, by [23] it is easy to show that the justified stable semantics for general programs agrees with the conditional-satisfaction based semantics for aggregate programs, under the translation $\tau$. It is known that the complexity for these programs (where admissible solutions are explicitly listed) under the conditional-satisfaction based semantics is NP-complete, for consistency [28]. Thus, the complexity for basic programs under the justified stable semantics is NP-complete for both consistency and credulous reasoning. By a similar proof, it is easy to show that for skeptical reasoning the complexity is co-NP-complete.

For disjunctive programs, the justified stable semantics agrees with the standard ASP semantics (Theorem 1). So the complexity for the justified stable semantics is the same as that for the standard ASP semantics, which is $\Sigma_2^p$-complete for consistency and credulous reasoning, and $\Pi_2^p$-complete for skeptical reasoning [5, 6].

For non-disjunctive and general programs, the complexity for the justified stable semantics is formally stated as follows.

**Theorem 3.** *(1) The problem* Consistency *is $\Sigma_2^p$-complete for general programs. (2) The problem* Credulous reasoning *is $\Sigma_2^p$-complete for general programs. (3) The problem* Skeptical reasoning *is $\Pi_2^p$-complete for general programs. (4) These problems for non-disjunctive programs are still complete on the second level of the polynomial hierarchy.*

The most interesting part of this theorem is the non-disjunctive case. Recall that a basic program is a non-disjunctive general program where the body of a rule is a conjunction of complete DNFs and their negations, where each complete DNF explicitly encodes the admissible solutions of an aggregate atom. While the complexity for basic programs is on the first level of the polynomial hierarchy, if we generalize this to an arbitrary formula in the body of a rule, the complexity rises to the second level.

The main idea in the proof of this theorem is the following. We can show that general programs can be mapped to default theories by a polynomial translation. We recall that the $\Sigma_2^p$-membership of Reiter's default logic for Consistency is shown by guessing a set of generating defaults for a given default theory first and then testing if this set determines an extension of the default theory [15]. However, this algorithm cannot be directly lifted to the justified stable semantics for general programs, because we need also to nondeterministically determine a subset of the head atoms for each rule in the general program. However, by the results in [21, 25], the complexity for the nonmonotonic modal logic S4F is on the second level and each general program can be equivalently transformed into a theory in the nonmonotonic modal logic S4F in polynomial time. Thus, the justified stable semantics for general programs is on the second level.

We now give the details.

**Proof of Theorem 3:** We first prove the statements (1)-(3) and then the statement (4).

*Hardness for general programs*: By Theorem 1, the justified stable semantics coincides with standard stable model semantics for disjunctive programs. It is shown in [6] that the complexity for the latter is hard on the second level (i.e. *Consistency* and *Credulous reasoning* are $\Sigma_2^p$-hard while *Skeptical reasoning* is $\Pi_2^p$-hard). Therefore, the justified stable semantics for general programs is at least as hard as problems on the second level of the polynomial hierarchy.

*Membership for general programs*: It can be obtained by combining results in [21, 25]. The basic idea is to embed general programs into nonmonotonic logic S4F.

Let $\mathcal{L}$ be the classical propositional language. The language of autoepistemic logic $\mathcal{L}_{ae}$ is the propositional language obtained from $\mathcal{L}$ by augmenting a unary modal operator $L$. For a formula $\phi$ in $\mathcal{L}_{ae}$, $L\phi$ means that "$\phi$ is believed". A theory $T$ is a finite subset of $\mathcal{L}_{ae}$. Let $\models_{S4F}$ be the consequence relation in modal logic S4F and $cons_{S4F}(T)$ denotes the set of consequences of $T$ in S4F.

**Lemma 2.** *The problem of deciding $T \models_{S4F} \phi$ for a theory $T$ and a formula $\phi$ in S4F is in $\Delta_2^p$.*

A deterministic algorithm can be designed similar to the algorithm in [15] (page 12). The algorithm is polynomial with a polynomial number of queries to NP-oracle.

Let $\Delta$ be a theory in S4F. A theory $E$ is an *S4F expansion* of $\Delta$ iff $E = cons_{S4F}(\Delta \cup \{\neg L\phi \mid \phi \notin E\})$. Based on the above result, the following lemma can be proven [21].

**Lemma 3.** *(1) The problem of deciding whether a theory $T$ has an S4F expansion is in $\Sigma_2^p$.*
*(2) The problem of deciding whether a formula is in an S4F expansion is in $\Sigma_2^p$.*
*(3) The problem of deciding whether a formula is in all S4F expansions is in $\Pi_2^p$.*

Given a general $P$, each rule $r$ of the form $a_1|\cdots|a_k \leftarrow body(r)$ can be transformed into a formula $tr(r)$ in S4F:

$$L\, body(r) \rightarrow La_1 \vee \cdots \vee La_k.$$

Denote $T_P = \{tr(r) \mid r \in P\} \cup \{L\neg La \rightarrow L\,\neg a \mid a \in \Sigma\}$. Then $T_P$ is a theory in $\mathcal{L}_{ae}$.

The following lemma is a special case of the result in [25].

**Lemma 4.** *Let $P$ be a general program and $I$ be an interpretation. $I$ is a default answer set of $P$ iff there exists an S4F expansion $E$ of $tr(P)$ such that $I$ coincides with the set of atoms in $E$.*

By Lemmas 3 and 4, it is easy to see that *Consistency* and *Credulous reasoning* for general programs under the justified stable semantics are in $\Sigma_2^p$ while *Skeptical reasoning* is in $\Pi_2^p$.

We then show that the justified stable semantics for non-disjunctive programs is still complete on the second level.

*Membership for non-disjunctive programs*: While the membership can be seen from the above argument for general programs, here we present a direct proof by providing a $\Sigma_2^p$-algorithm for deciding whether a program has a justified stable set: Let $P$ be a non-disjunctive program. Guess an interpretation $I$, the literal closure $\Sigma(P, I)$ can be computed in polynomial time using the NP-oracle of verifying the validity of propositional consequences. Then check if $I = \Sigma(P, I)$. If yes, $I$ is a justified stable model of $P$. Since only a linear number of queries to NP-oracle are used, the problem is in $\Sigma_2^p$.

*Hardness for non-disjunctive programs*: The hardness is not straightforward. We prove it by a polynomial transformation from $QBF_{2,\exists}$. That is, we construct a polynomial time transformation mapping each QBF $Q = \exists p_1 \cdots \exists p_n \forall q_1 \cdots \forall q_m E$ to a program $P_Q$ such that $Q$ is valid for a truth assignment iff $P_Q$ has an answer set. Here $E$ is a propositional formula made of the atoms $p_1, \ldots, p_n, q_1, \ldots, q_m$. We introduce $n + 3$ new atoms: $p'_1, \ldots, p'_n, f, f'$ and $g$. Denote $P_i = \{p'_i \leftarrow \neg p_i.\ p_i \leftarrow \neg p'_i.\ f \leftarrow p \wedge p' \wedge \neg f\}$ for $i = 1, \ldots, n$ and $P' = \{f' \leftarrow \neg q_j \wedge \neg f' \mid 1 \leq j \leq m\}$.

Let $P_Q = \{g \leftarrow \neg F \wedge \neg g\} \cup P' \cup \bigcup_{1 \leq i \leq n} P_i$.

The sub-program $P_i$ is to guess a truth value for each $p_i$. By the definition of the justified stable semantics, the CWA automatically applies to every negative literals. However, we do not intend to apply the CWA on the set $\{\neg q_1, \ldots, \neg q_m\}$ and thus $P'$ is used here.

The program $P_Q$ can be constructed from $Q$ in polynomial time. Also, each truth assignment to $p_1, \ldots, p_n$, under which $Q$ is true, corresponds to a justified stable model of $P$. That is, $Q$ is valid iff $P_Q$ has justified stable model. Therefore, *Consistency* for program is $\Sigma_2^p$-hard.

In a similar way, it can be shown that *Credulous reasoning* is $\Sigma_2^p$-hard. As a result, *Skeptical reasoning* is $\Pi_2^p$-hard. □

## 5 Related Work

### 5.1 Well-supportedness

The idea of *unfounded sets* for normal programs [27] has been applied to logic programs with monotone and antimonotone aggregates [3]. But this approach only provides a necessary condition when extended to arbitrary aggregates [18], i.e., an answer set is unfounded-free, but a model being unfounded-free is not guaranteed to be an answer set. In [24], the existence of a level mapping w.r.t. a model is shown to be a necessary condition for the model to be an answer set. The notion of well-supportedness formulated in this paper is a necessary condition in general, and a necessary and sufficient condition for non-disjunctive programs.

### 5.2 Relation with the FLP-semantics

The FLP-semantics [9, 10] is a simple and elegant semantics for aggregate programs, which has been extended to languages that combine ASP with external atoms [8], to arbitrary propositional formulas [26], and to first-order general programs [2].

The key idea in the original formulation of FLP-semantics is a notion of *reduct* that keeps the rules whose bodies are satisfied by an interpretation $I$. Then, $I$ is an FLP-stable model if $I$ is a minimal model of the reduct. For aggregate programs, the difference between the FLP-semantics and the conditional satisfaction-based semantics has been studied in [23]. For programs with arbitrary formulas, let us consider propositional general programs defined in this paper, for which the FLP-semantics is also defined. In the sequel, FLP-semantics refers to the one defined in [2].

Let $\Pi$ be a general program, where a rule is denoted by $H \leftarrow B$. Tailored to the propositional case, let $\mathbf{p}$ be the list of distinct predicate constants $(p_1, \ldots, p_n)$ appearing in $\Pi$ (i.e., they are 0-ary predicates corresponding to atoms). By $\mathbf{FLP}[\Pi, \mathbf{p}]$, we denote the seconde-order sentence

$$\Pi \wedge \neg\exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge \Pi^{\star}(\mathbf{u})) \tag{1}$$

where $\mathbf{u} = (u_1, \ldots, u_n)$ is a list of $n$ distinct predicate variables, and $\Pi^{\star}(\mathbf{u})$ is the conjunction of

$$H(\mathbf{u}) \leftarrow B \wedge B(\mathbf{u}) \tag{2}$$

for all rules in $\Pi$, where $G(\mathbf{u})$ denotes the formula obtained from formula $G$ by replacing all occurrences of the atoms from $\mathbf{p}$ with the corresponding atom variables from $\mathbf{u}$. For propositional logic, $\mathbf{u} < \mathbf{p}$ is a shorthand for $(u_1, \ldots, u_n) < (p_1, \ldots, p_n)$, which means $u_i \leq p_i$ for all $i$ and there exists some $k$ ($1 \leq k \leq n$) such that $u_k < p_k$, where the truth value $true$ is 1 and $false$ is 0.

An *FLP-stable model* of $\Pi$ is defined as a model of $\mathbf{FLP}[\Pi, \mathbf{p}]$.

We can show the following proposition.

**Proposition 4.** *Let $\Pi$ be a general program and $I$ a model of $\Pi$. If $I$ is a justified stable model of $\Pi$ then it is an FLP-stable model of $\Pi$, but not vice versa.*

*Example 3.* Consider again $\Pi_2 = \{a \leftarrow b,\ b \leftarrow \neg b \vee a\}$. It can be verified that the interpretation $I = \{a, b\}$ is an FLP-stable model of $\Pi_2$.

$$\mathbf{FLP}[\Pi_2, a, b] = \\ \Pi_2 \wedge \neg\exists(a^*, b^*)((a^*, b^*) < (a, b)) \wedge \Pi_2^\star(a^*, b^*)$$

where $\Pi_2^\star(a^*, b^*)$ is

$$(a^* \leftarrow b \wedge b^*) \wedge (b^* \leftarrow (\neg b \vee a) \wedge (\neg b^* \vee a^*))$$

Since $I$ is a model of $\mathbf{FLP}[\Pi_2, a, b]$, it is an FLP-stable model of $\Pi_2$. But $I$ is not a justified stable model of $\Pi_2$. By Proposition 3, we know that, for any non-disjunctive program, any model that is not justified does not possess a level mapping. We see that this is the case for $I$, which incurs a self-supporting loop.

### 5.3 Relation with Aggregate Programs

The class of programs studied in [23] correspond to non-disjunctive programs of this paper. The definition of answer set there is tied to the notion of default extensions, but the semantics coincides with the justified stable semantics of this paper, for non-disjunctive programs. It is shown in [23] that the conditional satisfaction-based semantics for aggregate programs [24] can be captured by non-disjunctive general programs. However, the complexity issue is completely absent, and the notion of well-supportedness is only discussed informally. In this paper, not only this is defined formally for the class of all general programs, but is also shown to be a sufficient and necessary condition for a model to be a justified stable model, for non-disjunctive programs. In doing so, we are able to answer the question, formally, what if a model of an aggregate program is not a justified stable model - then we know that it is necessarily the case that the model does not possess a level mapping, and as a result, some atoms in it depend on the truth of themselves.

To see further that the ultimate stable semantics [19], or alternatively the conditional satisfaction-based semantics [24], are rooted on the justified stable semantics, we can define a semantics for *general programs with aggregates*, in which the body of a rule is built from atoms and aggregate atoms using standard (classical) connectives. We extend the mapping $\tau$ in Definition 7 to arbitrary formulas as follows: Given a general program with aggregates $\Pi$, $\tau(\Pi)$ is the general program obtained from $\Pi$ where each aggregate $A$ in $\Pi$ is replaced by $\tau(A)$. Then, we define the justified stable semantics of $\Pi$ via the justified stable semantics of $\tau(\Pi)$. Immediately, we have

**Proposition 5.** *Let $\Pi$ be a general program with aggregates.*

- *Justified stable models of $\Pi$ are well-supported.*
- *If $\Pi$ is an aggregate program, justified stable models of $\Pi$ are precisely the answer sets of $\Pi$ as defined in [24].*

Note that, aggregates in a general program with aggregates can be defined in the form of [24]. We use the form of abstract atoms only for semantic purposes. As such, the complexity results presented in this paper for general programs do not automatically apply to general programs with aggregates.

## 6 Well-Supported Answer Sets for DL-Programs

We assume that the reader has some familiarity with description logics (DLs) [1], which are decidable fragments of first order logic.

A *dl-program* is a pair $\mathcal{K} = (L, R)$, where $L$ is a DL knowledge base and $R$ is a rule base. In [7] *weak* and *strong* answer sets are defined, and it is commented that the reason to introduce the latter is because there may exist self-supporting loops in weak answer sets. Later, it is also found that even strong answer sets may incur self-supporting loops. To circumvent this problem, the well-supported semantics for dl-programs is proposed [22]. We can show that the well-supported semantics for dl-programs is rooted in the justified stable semantics of general programs. This is sketched below.

A rule in a dl-program $\mathcal{K} = (L, R)$ is of the form

$$H \leftarrow A_1 \wedge \cdots \wedge A_m \wedge not\ B_1 \wedge \cdots \wedge not\ B_n \tag{3}$$

where $H$ is an atom, and $A_i$ and $B_i$ are atoms or *dl-atoms*,[1] which are of the form

$$DL[S_1 op_1 p_1, \cdots, S_m op_m p_m; Q](\mathbf{t}) \tag{4}$$

in which $S_i$ is a concept or role from the vocabulary of $L$, $op_i \in \{\uplus, \cup, \cap\}$, $p_i$ is a predicate symbol only appearing in $R$ whose arity matches that of $S_i$, and $Q(\mathbf{t})$ is a dl-query.[2]

In this paper, we assume that rules in $R$ are ground.

The *Herbrand base* of $R$, denoted $HB_R$, is the set of all ground atoms $p(t_1, ..., t_m)$, where $p$ appears in $R$ and $t_i$ is a constant. Any subset of $HB_R$ is an *interpretation* of $R$.

**Definition 9.** Let $\mathcal{K} = (L, R)$ be a dl-program and $I$ an interpretation (of $R$). The *satisfaction relation under* $L$, denoted $\models_L$, is defined as:

1. $I \models_L \top$ and $I \not\models_L \bot$.[3]
2. For any atom $a \in HB_R$, $I \models_L a$ if $a \in I$.
3. For any dl-atom $A{=}DL[S_1 op_1 p_1, \cdots, S_m op_m p_m; Q](\mathbf{c})$ occurring in $ground(R)$, $I \models_L A$ if $L \cup \bigcup_{i=1}^{m} A_i \models Q(\mathbf{c})$, where

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

4. For any ground atom or dl-atom $A$, $I \models_L not\ A$ if $I \not\models_L A$.

Given a dl-program $\mathcal{K} = (L, R)$, we map a dl-atom to a c-atom, which can then be translated to a propositional formula. The resulting program is a general program.[4]

Let $A$ be a dl-atom in $R$. We represent $A$ by a c-atom $(D, C)$, produced by a mapping $\beta$, i.e., $\beta(A) = (D, C)$, where $D$ is a set of atoms appearing in $R$, and $C = \{I \subseteq HB_R \mid I \models_L A\}$.

[1] For simplicity, we assume that equality doesn't appear in rules.

[2] A *dl-query* is of the form $Q(\mathbf{t})$, where $\mathbf{t}$ is a list of terms, and $Q$ is a concept, a role, or a concept inclusion axiom, built from the vocabulary of $L$.

[3] In DL, $\top$ is the most general concept and $\bot$ is the empty concept.

[4] The complexity of this translation depends on the underlying DL.

**Definition 10.** *Let $\mathcal{K} = (L, R)$ be a dl-program and denote by $\xi_L(R)$ the general program obtained from $R$ by (1) replacing any dl-atom $A$ in $R$ by $\tau \circ \beta(A)$, (2) replacing the default negation symbol $not$ by $\neg$, (3) replacing $\bot$ by $false$ and $\top$ by $true$.*

We refer the reader to [22] for the definition of well-supported answer set, which, as stated in the following theorem, is rooted on the justified stable semantics.

**Theorem 4.** *Let $\mathcal{K} = (L, R)$ be a dl-program and $I \subseteq HB_R$ an interpretation. $I$ is a (strongly) well-supported answer set of $\mathcal{K}$ iff $I$ is a justified stable model of the general program $\xi_L(R)$.*

## 7 Final Remarks

For normal and disjunctive programs, there exist a number of different ways to define the same semantics (cf. [16]). However, beyond normal and disjunctive logic programming, this agreement begins to depart in two different directions in the literature. One is based on minimal models in terms of a reduct or modified circumscription. This approach is mathematically elegant, and general enough to deal with arbitrary formulas, including first-order formulas. The other is the family of the semantics defined by fixpoints of a monotonic operator. Such a definition typically carries a form of justified reasoning, natively.

Minimization in the forms mentioned above and justified reasoning are two related but separate issues. The former does not guarantee the latter. Unfortunately, this issue has not been paid sufficient attention in the recent literature.

Under this circumstance, in this paper we define the justified stable semantics for general programs, formulate a notion of well-supportedness for these programs, and present the complexity results for the semantics. Furthermore, perhaps most interestingly, the work is not only about a specific semantics for a specific class of programs. The framework is unifying in that the semantics for various kinds of logic programs, known to be well-supported in the literature, are all rooted in the justified stable semantics of general programs.

The two approaches mentioned above are closely related, in the sense that the well-supported stable models are always FLP-stable models but not vice versa - when they differ, we now have a precise understanding of what exactly the difference is: for an aggregate program/dl-program/general program with aggregates, any FLP-stable model that is not a justified stable model is not well-supported, i.e., it is necessarily the case that some atoms in it depend on the truth of themselves.

There are several interesting questions that require further investigation. In this paper we only showed that a justified stable model is well-supported. A question is how to characterize justified stable models in terms of well-supportedness, for general programs where rules may have a disjunctive head. That is, we would be interested in a sufficient and necessary condition between justified stable models and well-supportedness, for the class of all general programs. Another question is what would be the counterpart of justified stable models for first-order general programs. Finally, although we do not expect a huge impact on implementation techniques, since the justified stable semantics is closely related to default logic [20], it is interesting to observe that recent effort in implementing the latter is by adopting ASP techniques [4].

## References


1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. Michael Bartholomew, Joohyung Lee, and Yunsong Meng. First-order extension of the flp stable model semantics via modified circumscription. In *Proc. IJCAI-11*, pages 724–730, 2011.
3. Francesco Calimeri, Wolfgang Faber, Nicola Leone, and Simona Perri. Declarative and computational properties of logic programs with aggregates. In *Proc. IJCAI-05*, pages 406–411, 2005.
4. Yin Chen, Hai Wan, Yan Zhang, and Yi Zhou. dl2asp: Implementing default logic via answer set programming. In *Proceedings 12th European Conference on Logics in Artificial Intelligence*, pages 104–116, 2010.
5. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Survey*, 33(3):374–425, 2001.
6. T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, 1995.
7. Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artifical Intelligence*, 172(12-13):1495–1539, 2008.
8. Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proc. IJCAI-05*, pages 90–96, 2005.
9. Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proc. JELIA'04*, 2004.
10. Wolfgang Faber, Gerald Pfeifer, and Nicola Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298, 2011.
11. François Fages. Consistency of clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.
12. Paolo Ferraris. Answer sets for propositional theories. In *Proc. LPNMR-05*, pages 119–131, 2005.
13. Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription. *Artificial Intelligence*, 175(1):236–263, 2011.
14. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
15. G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2(3):397–425, 1992.
16. Vladimir Lifschitz. Twelve definitions of a stable model. In *Proc. ICLP-08*, pages 37–51, 2008.
17. Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):369–389, 1999.
18. Guohua Liu and Jia-Huai You. Lparse programs revisited: Semantics and representation of aggregates. In *Proc. ICLP-08*, pages 347–361, 2008.
19. N. Pelov, M. Denecker, and M. Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming*, 7:301–353, 2007.
20. Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.
21. G. Schwarz and M. Truszczynski. Nonmonotonic reasoning is sometimes simpler! *Journal of Logic and Computation*, 6(2):295–308, 1996.

22. Yi-Dong Shen. Well-supported semantics for description logic programs. In *Proc. IJCAI-11*, pages 1081–1086, 2011.
23. Yi-Dong Shen and Jia-Huai You. A default approach to semantics of logic programs with constraint atoms. In *Proc. LPNMR-09*, pages 277–289, 2009.
24. T. C. Son and E. Pontelli. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming*, 7:355–375, 2007.
25. M. Truszczynski. Modal interpretations of default logic. In *Proc. IJCAI-91*, pages 393–398, 1991.
26. Miroslaw Truszczynski. Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. *Artificial Intelligence*, 174(16-17):1285–1306, 2010.
27. Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.
28. Jia-Huai You, Li Yan Yuan, Guohua Liu, and Yid-Dong Shen. Logic programs with abstract constraints: representation, disjunction, and complexities. In *Proc. LPNMR-07*, pages 228–240, 2007.
29. Yi Zhou and Yan Zhang. Progression semantics for disjunctive logic programs. In *Proc. AAAI-11*, pages 286–291, 2011.