Complexity of fuzzy answer set programming under Łukasiewicz semantics

Marjon Blondeel[a,1], Steven Schockaert[b], Dirk Vermeir[a], Martine De Cock[c]

[a] *Vrije Universiteit Brussel, Department of Computer Science, Pleinlaan 2, 1050 Brussel, Belgium*

[b] *Cardiff University, School of Computer Science and Informatics, 5 The Parade, Cardiff, CF24 3AA, UK*

[c] *Ghent University, Department of Applied Mathematics, Computer Science and Statistics, Krijgslaan 281 (S9), 9000 Gent, Belgium*

Abstract

Fuzzy answer set programming (FASP) is a generalization of answer set programming (ASP) in which propositions are allowed to be graded. Little is known about the computational complexity of FASP and almost no techniques are available to compute the answer sets of a FASP program. In this paper, we analyze the computational complexity of FASP under Łukasiewicz semantics. In particular we show that the complexity of the main reasoning tasks is located at the first level of the polynomial hierarchy, even for disjunctive FASP programs for which reasoning is classically located at the second level. Moreover, we show a reduction from reasoning with such FASP programs to bilevel linear programming, thus opening the door to practical applications. For definite FASP programs we can show P-membership. Surprisingly, when allowing disjunctions to occur in the body of rules – a syntactic generalization which does not affect the expressivity of ASP in the classical case – the picture changes drastically. In particular, reasoning tasks are then located at the second level of the polynomial hierarchy, while for simple FASP programs, we can only show that the unique answer set can be found in pseudo-polynomial time. Moreover, the connection to an existing open problem about integer equations suggests that the problem of fully

Email addresses: `Marjon.Blondeel@vub.ac.be` (Marjon Blondeel), `S.Schockaert@cs.cardiff.ac.uk` (Steven Schockaert), `Dirk.Vermeir@vub.ac.be` (Dirk Vermeir), `Martine.DeCock@UGent.be` (Martine De Cock)

[1]Funded by a joint Research Foundation-Flanders (FWO) project

characterizing the complexity of FASP in this more general setting is not likely to have an easy solution.

Keywords: Answer set programming, Łukasiewicz logic, Computational complexity

1. Introduction

Answer set programming (ASP) [1] is a form of declarative programming that can be used to model combinatorial search problems. Specifically, a search problem is translated into a disjunctive ASP program, i.e. a set of rules of the form

$$r : a_1 \vee \ldots \vee a_n \leftarrow b_1 \wedge \ldots \wedge b_m \wedge \text{not}\, c_1 \wedge \ldots \wedge \text{not}\, c_k,$$

with a_i, b_j, c_l literals (atoms or negated atoms) or constants ("true" or "false") and "not" the negation-as-failure operator. In ASP there are two types of negation: strong negation "$\neg$" and negation-as-failure "not". The intuitive difference is that a literal $\neg a$ is true when $\neg a$ can be derived, whereas $\text{not}\, a$ is true if a cannot be derived. Rule r indicates that whenever the body $b_1 \wedge \ldots \wedge b_m \wedge \text{not}\, c_1 \wedge \ldots \wedge \text{not}\, c_k$ holds, the head $a_1 \vee \ldots \vee a_n$ should hold as well. For example, consider the following ASP program P.

$$\begin{array}{lll} r_1 : & \text{beach} & \leftarrow & \text{sunny} \wedge \text{not raining} \\ r_2 : & \text{sunny} & \leftarrow & \overline{1} \end{array}$$

Rule r_1 informally means that we will go to the beach if there is no reason to think that it is raining and if we are sure that it is sunny. A rule such as r_2 is called a fact; the head is unconditionally true, it is sunny. Given such a program, the idea is to find a minimal consistent set of literals that can be derived from the program, where consistent means that an atom a and its negation $\neg a$ cannot be both elements of this set. These "answer sets" then correspond to the solutions of the original search problem. By rule r_2 it follows that the literal "sunny" must be an element of each answer set. Explicitely, the only answer set of P is $\{\text{sunny}, \text{beach}\}$.

If the head of each rule in an ASP program consists of at most one literal, the program is called *normal*. If, in addition, a normal program does not contain "not", it is called *definite*. Given a disjunctive ASP program P and a literal l, we are interested in the following three decision problems.

1. **Existence**: Does P have an answer set?
2. **Set-membership**: Does there exist an answer set I of P such that $l \in I$?
3. **Set-entailment**: Does $l \in I$ hold for each answer set I of P?

A summary of the complexity for these decision problems in classical ASP is given in Table 1. Recall that $\Pi_2^P = \text{co}\Sigma_2^P$, where Σ_2^P is the class of problems that can be solved in polynomial time on a non-deterministic machine using an NP oracle.

Table 1: Complexity of inference in ASP [1, 2]

	existence	set-membership	set-entailment
disjunctive	Σ_2^P-complete	Σ_2^P-complete	Π_2^P-complete
normal	NP-complete	NP-complete	coNP-complete
definite	in P	in P	in P

Although ASP has been successfully applied to model combinatorial problems in a concise and declarative manner, it is not directly suitable for expressing problems with continuous domains. Fuzzy answer set programming (FASP) (e.g. [3, 4]) is a generalization of ASP based on fuzzy logics [5] that is capable of modeling continuous systems by using an infinite number of truth values corresponding to intensities of properties. Łukasiewicz logic, a particular type of fuzzy logic, is often used in applications because it preserves many desirable properties from classical logic. It is closely related to mixed integer programming, as was first shown by McNaughton [6] in a non-constructive way. Later, Hähnle [7] gave a concrete, semantics-preserving, translation from a set of formulas in Łukasiewicz logic into a mixed integer program. Checking the satisfiability of a Łukasiewicz logic formula thus essentially corresponds to checking the feasibility of a mixed integer program. Under Łukasiewicz semantics, a *strict disjunctive FASP program* is a set of rules of the form

$$r : a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m \otimes \text{not}\, c_1 \otimes \ldots \otimes \text{not}\, c_k$$

with a_i, b_j, c_l literals (atoms or negated atoms) or constants $\overline{c}$ (with $c \in [0,1] \cap \mathbb{Q}$), "not" the negation-as-failure operator, $\oplus$ and $\otimes$ resp. the Łukasiewicz disjunction and conjunction and $\leftarrow$ the Łukasiewicz implicator (see Section 2.2). Special kinds of strict disjunctive FASP programs we are interested in are *strict normal FASP programs*, i.e. strict disjunctive FASP programs in which each rule has at most one literal in the head, and *strict definite FASP programs*, i.e. strict normal FASP programs that do not contain "not". The label "strict" refers to the fact that in terms of syntax they adhere strictly to the standard syntax of classical ASP, i.e. the body of a rule consists of a conjunction of (negation-as-failure) literals or constants, while the head is a disjunction. In this paper, we will also consider FASP programs with more syntactical freedom, in particular sets of rules with other connectives from Łukasiewicz logic in the body and the head. We will refer to this more general class of FASP programs as *regular FASP programs.* The class of strict FASP programs is a subclass of the class of regular FASP programs.

As will become clear when defining the semantics, in a FASP program the truth value of the head of each rule must be greater or equal to the truth value of the body of this rule. In line with the idea of ASP – which attempts to make as few literals true as possible to satisfy the rules of a program – we are interested in finding the lowest truth degrees that we can assign to each of the literals such that the rules are still satisfied. In this way FASP can model search problems in continuous domains entirely similar as ASP does for search problems with discrete domains. For example, consider the following program P.

$$\begin{array}{rlcl} r_1 : & \text{open} & \leftarrow & \text{not closed} \\ r_2 : & \text{closed} & \leftarrow & \text{not open} \end{array}$$

The properties "open" and "closed" can be given a value in $[0, 1]$ depending on the extent, e.g. the angle, to which a door is opened resp. closed. Rule r_1 intuitively means that the door is open to at least a degree greater or equal to the extent to which the door is not closed and vice versa for rule r_2. As will become clear in Section 2.3, all answer sets of this program are of the form $I(\text{open}) = x$ and $I(\text{closed}) = 1 - x$ for some $x \in \mathbb{Q}$ and there are no other answer sets. Note that answer sets are now fuzzy sets on the set of literals, i.e. mappings I from the set of literals into $[0, 1]$.

In recent years, a variety of approaches to FASP have been proposed (e.g. [8, 9, 10, 11, 12, 13, 14]). The main differences are the type of connectives that are allowed, the truth lattices that are used, the definition of a model of a program and the way that partial satisfaction of rules is handled. Note that the idea behind FASP is not to deal with uncertainty, but with partial truth. See [15] for a discussion on the difference between these two concepts. Although there exist fuzzy frameworks in which uncertainty can be modelled, e.g. by using intervals of truth values [14], it is more naturally to extend ASP with possibility theory (e.g. [16]) or with probability theory (e.g. [17]) to deal with uncertainty. Still, FASP is sometimes useful as a vehicle to simulate probabilistic or possibilistic extensions of ASP, as its ability to model continuity can be used to manipulate certainty degrees [16, 18].

In this paper, we will investigate the complexity of important decision problems for FASP. Given a regular FASP program P, a literal l and a value $\lambda_l \in [0, 1] \cap \mathbb{Q}$, we are interested in the following decision problems which are generalizations of the ones for ASP.

1. **Existence**: Does there exist an answer set I of P?
2. **Set-membership**: Does there exist an answer set I of P such that $I(l) \geq \lambda_l$?
3. **Set-entailment**: Is $I(l) \geq \lambda_l$ for each answer set I of P?

By our particular choice of semantics, FASP relates to Łukasiewicz logic as ASP does to classical logic. For Łukasiewicz logic, satisfiability is an NP-complete problem [19]. Since this problem has the same complexity for classical logic, one would expect ASP and FASP to have the same complexity as well. This expectation is reinforced by the fact that in the case of probabilistic ASP, the complexity of the existence problem has been shown to be Σ_2^P-complete [20], i.e. the same as the complexity of the existence problem in classical ASP. On the other hand, it does not necessarily need to hold that the computational complexity remains the same. There are for instance fuzzy description logics that, unlike the classical case, do not have the finite model property under Łukasiewicz logic or under product logic [21] and there are description logics that are undecidable under Łukasiewicz logic [22].

The main contributions of this paper are the following:

- Although existence and set-membership are Σ_2^P-complete for disjunctive ASP, for strict disjunctive and strict normal FASP we will show NP-completeness. Moreover, we will show that not allowing constraints, i.e. rules in which the head contains exactly one constant, and strong negation does not affect the complexity for set-membership.

- We will show that the existence of an answer set for a strict normal FASP program without constraints and without strong negation is always guaranteed and hence that the complexity of the existence problem for this class of FASP programs is "constant". However, for strict disjunctive FASP without constraints and without strong negation we are only able to show membership in NP for the existence problem.

- If more syntactic freedom is allowed, i.e. for regular FASP programs, then we can show Σ_2^P-completeness for set-membership and existence and Π_2^P-completeness for set-entailment by using known complexity results about fuzzy equilibrium logic [23]. However, if we restrict ourselves to programs with at most one literal in the head of each rule, then we can only show Σ_2^P-membership and NP-hardness for set-membership and existence and Π_2^P-membership and coNP-hardness for set-entailment. If in addition, we do not allow "not" in the rules we can only find a pseudo-polynomial time algorithm to compute answer sets based on computing least fixpoints.

- Although in general we can only show membership in NP $\cap$ coNP, for several subclasses of the class of regular definite FASP programs we can show P-membership. In particular, for regular definite FASP programs with only conjunction and maximum or only disjunction in the body of rules we can provide a polynomial time algorithm to compute answer sets. This is also the case for regular definite FASP programs with a cycle free dependency graph or with only polynomially bounded constants.

An overview of the complexity results that we can establish is provided in Tables 2 and 3.

Table 2: Complexity of inference in strict FASP

strict FASP		existence	set-membership	set-entailment
no restrictions	disjunctive	NP-complete	NP-complete	coNP-complete
	normal	NP-complete	NP-complete	coNP-complete
	definite	in P	in P	in P
no constraints, no strong negation	disjunctive	in NP	NP-complete	in coNP
	normal	in P	NP-complete	in coNP
	definite	in P	in P	in P

Finally, we will provide an implementation into bilevel linear programming for strict disjunctive FASP which opens the door to practical applications. Intuitively, in a bilevel linear programming problem there are two agents: the leader and the follower. The leader goes first and attempts to optimize his/her objective function. The follower observes this and subsequently makes his/her decision. Since it caught the attention in the 1970s,

Table 3: Complexity of inference in regular FASP

regular FASP		existence	set-membership	set-entailment
no restrictions	disjunctive	Σ_2^P-complete	Σ_2^P-complete	Π_2^P-complete
	normal	NP-hard, in Σ_2^P	NP-hard, in Σ_2^P	coNP-hard, in Π_2^P
	definite	in NP $\cap$ coNP	in NP $\cap$ coNP	in NP $\cap$ coNP
only $\otimes$ and max in body	definite	in P	in P	in P
only $\oplus$ in body	definite	in P	in P	in P
cycle free	definite	in P	in P	in P
polynomially bounded constants	definite	in P	in P	in P

there have been many algorithms proposed for solving bilevel linear programming problems (e.g. [24, 25, 26]). A popular way to solve such a problem, e.g. in [24], is to translate the bilevel linear programming problem into a nonlinear programming problem using Kuhn-Tucker constraints. This new program is a standard mathematical program and relatively easy to solve because all but one constraint is linear. In a later study [27], an implicit approach to satisfying the nonlinear complementary constraint was proposed, which proved to be more efficient than the known strategies. By showing a reduction of strict disjunctive FASP into bilevel programming we thus provide a basis to build solvers for FASP.

The paper is structured as follows. In the next section, we provide the necessary background on ASP, Łukasiewicz logic and FASP followed by some motivating examples for FASP in Section 3. In Section 5, resp. Section 6, we will present complexity results for strict FASP, resp. regular FASP using propositions and lemmas provided in Section 4. Finally, in Section 7, we will show a reduction to bilevel linear programming for disjunctive FASP and in Section 8 we present some concluding remarks. Furthermore, the proofs for all results have been added in an appendix.

2. Preliminaries

2.1. Answer set programming (ASP)

A *disjunctive* ASP program is a finite set of rules of the form

$$r : a_1 \vee \ldots \vee a_n \leftarrow b_1 \wedge \ldots \wedge b_m \wedge \text{not}\, c_1 \wedge \ldots \wedge \text{not}\, c_k,$$

with a_i, b_j, c_l literals (atoms a or negated atoms $\neg a$) and/or the constants $\overline{1}$ (true) or $\overline{0}$ (false) with $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$ and $l \in \{1, \ldots, k\}$. The operator "not" is the *negation-as-failure operator* and "$\neg$" is called *strong* *negation*. Intuitively, the expression $\text{not}\, a$ is true if there is no proof that supports a. On the other hand, $\neg a$ is essentially seen as a new literal, which has no connection to a, except for the fact that answer sets containing both a and $\neg a$ will be designated as inconsistent. (Strongly) negated literals are defined as follows: $\neg l := \neg a$ if $l = a$ and $\neg l := a$ if $l = \neg a$ (with a an atom). An expression of the form $\text{not}\, l$ (with l a literal) will be called a *negation-as-failure literal*.

We refer to the rule by its label r. The expression $a_1 \vee \ldots \vee a_n$ is called the *head* r_h of r and $b_1 \wedge \ldots \wedge b_m \wedge \text{not}\, c_1 \wedge \ldots \wedge \text{not}\, c_k$ is the *body* r_b of r. In ASP, a rule of the form "$\overline{0} \leftarrow \alpha$", i.e. a *constraint*, is usually written as "$\leftarrow \alpha$" and a rule of the form "$\alpha \leftarrow \overline{1}$", i.e. a *fact*, as "$\alpha \leftarrow$". In a constraint the body is unconditionally false and in a fact the head is unconditionally true.

Different classes of ASP programs are often considered, depending on the type of rules they contain. If each rule in P has at most one literal in the head, it is called a *normal* ASP program. If P is a normal ASP program not containing negation-as-failure, it is called a *definite* ASP program. A definite ASP program not containing strong negation with exactly one atom in the head of each rule is called a *simple* ASP program.

An *interpretation* I of P is any consistent set of literals $I \subseteq \mathcal{L}_P$ with

$$\mathcal{L}_P = \{l \mid l \text{ literal in } P\} \cup \{\neg l \mid l \text{ literal in } P\}$$

and where we say that I is consistent if for no literal l in $\mathcal{L}_P$ we have that $l \in I$ and $\neg l \in I$. The set of interpretations $I \subseteq \mathcal{L}_P$ will be denoted by $\mathcal{P}(\mathcal{L}_P)$. A literal l is *true in* I, written as $I \models l$, iff $l \in I$. An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ can be extended to rules as follows:

- $I \models \overline{1}$, $I \not\models \overline{0}$,
- $I \models \text{not}\, l$ iff $I \not\models l$,
- $I \models (\alpha \wedge \beta)$ iff $I \models \alpha$ and $I \models \beta$,
- $I \models (\alpha \vee \beta)$ iff $I \models \alpha$ or $I \models \beta$,
- $I \models (\alpha \leftarrow \beta)$ iff $I \models \alpha$ or $I \not\models \beta$.

with l a literal and α and β propositional formulas in negation-normal form.

An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ is called a *model* of a disjunctive ASP program P if $I \models r$ for each rule $r \in P$. A model I of P is *minimal* if there exists no model J of P such that $J \subset I$, i.e. $J \subseteq I$ and $J \neq I$. An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ is called an *answer set* of a disjunctive ASP program P without negation-as-failure if it is a minimal model of P. It can be shown that a simple ASP program has exactly one answer set. To define the semantics for disjunctive ASP programs P that contain negation-as-failure, one starts from a candidate answer set $I \in \mathcal{P}(\mathcal{L}_P)$ and computes the Gelfond-Lifschitz reduct P^I [28] by removing all rules in P that contain expressions of the form $\text{not}\, l$ with $l \in I$ and removing all expressions of the form $\text{not}\, l$ in the remaining rules. An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ is called an answer set of P if it is an answer set of the negation-as-failure free program P^I.

Example 1. *Consider the normal ASP program P*

$$\begin{aligned} b &\leftarrow \text{not}\, a \\ a &\leftarrow \text{not}\, b \end{aligned}$$

with a and b atoms. For the interpretation $I_1 = \{a\}$, we have that P^{I_1} is equal to

$$a \leftarrow$$

Since I_1 is a minimal model of P^{I_1}, we conclude that I_1 is an answer set of P. Similarly, $I_2 = \{b\}$ is also an answer set of P. One can easily check that I_1 and I_2 are the only answer sets.

Remark 1. *A disjunctive ASP program P with strong negation can be translated to a disjunctive ASP program P' without strong negation, by replacing each literal of the form $\neg a$ with a new atom a' and adding the constraint $\leftarrow a \wedge a'$. An interpretation $I \in \mathcal{P}(\mathcal{L}_P)$ is an answer set of P iff there exists an answer set $I' \in \mathcal{P}(\mathcal{L}_{P'})$ of P' such that $a \in I$ iff $a \in I'$ and $\neg a \in I$ iff $a' \in I'$ for each atom $a \in \mathcal{L}_P$.*

2.2. Łukasiewicz logic

Fuzzy logics [5] are a class of logics whose semantics are based on truth degrees taken from the unit interval $[0, 1]$. Łukasiewicz logic is a particular type of fuzzy logic that is often used in applications since it preserves many properties from classical logic and all the main reasoning tasks can be reduced to mixed integer programming.

In this paper, we will consider formulas built from a set of atoms A, constants $\overline{c}$ for each element $c \in [0, 1] \cap \mathbb{Q}$ and the connectives conjunction

$\otimes$, disjunction $\oplus$, max, min, negation $\sim$ and implication $\rightarrow$. The semantics of this logic are defined as follows. A *fuzzy interpretation* is a mapping $I : A \rightarrow [0, 1]$ that can be extended to arbitrary formulas:

- $[\overline{c}]_I = c$,
- $[\alpha \otimes \beta]_I = \max([\alpha]_I + [\beta]_I - 1, 0)$,
- $[\alpha \oplus \beta]_I = \min([\alpha]_I + [\beta]_I, 1)$,
- $[\max(\alpha, \beta)]_I = \max([\alpha]_I, [\beta]_I)$,
- $[\min(\alpha, \beta)]_I = \min([\alpha]_I, [\beta]_I)$,
- $[\alpha \rightarrow \beta]_I = \min(1 - [\alpha]_I + [\beta]_I, 1)$
- $[\alpha \leftrightarrow \beta]_I = [(\alpha \rightarrow \beta) \otimes (\beta \rightarrow \alpha)]_I = [\min(\alpha \rightarrow \beta, \beta \rightarrow \alpha)]_I$ and
- $[\sim \alpha]_I = 1 - [\alpha]_I$.

for a constant $\overline{c}$ and α and β formulas. Note that for formulas $\alpha_1, \dots, \alpha_n$ we have

$$[\bigotimes_{i=1}^{n} \alpha_i]_I = [\alpha_1 \otimes \dots \otimes \alpha_n]_I = \max(\sum_{i=1}^{n} [\alpha_i]_I - (n-1), 0),$$
$$[\bigoplus_{i=1}^{n} \alpha_i]_I = [\alpha_1 \oplus \dots \oplus \alpha_n]_I = \min(\sum_{i=1}^{n} [\alpha_i]_I, 1)$$

and it can be shown that the conjunction and disjunction operators are associative, although the usual distribution laws do not hold. Moreover remark that

$$[\alpha_1 \rightarrow \alpha_2]_I = 1 \text{ iff } [\alpha_1]_I \leq [\alpha_2]_I.$$

The set of all fuzzy interpretations $A \rightarrow [0, 1] \cap \mathbb{Q}$ will be written as $\mathcal{F}(A)$. We say that $I \in \mathcal{F}(A)$ is a *fuzzy model* of a set of formulas B if $[\alpha]_I = 1$ for each $\alpha \in B$.

2.3. Fuzzy answer set programming (FASP)

We now recall a fuzzy version of ASP based on [29], combining ASP (Section 2.1) and Łukasiewicz logic (Section 2.2).

A *regular FASP program* (under Łukasiewicz semantics) is a finite set of rules of the form

$$r : g(a_1, \ldots, a_n) \leftarrow f(b_1, \ldots, b_m, \text{not}\, c_1, \ldots, \text{not}\, c_k),$$

with a_i, b_j, c_l literals (atoms a or negated atoms $\neg a$) and/or constants $\overline{c}$ (where $c \in [0,1] \cap \mathbb{Q}$) with $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$ and $l \in \{1, \ldots, k\}$ and "not" the negation-as-failure operator. The connectives f and g are compositions of the Łukasiewicz connectives $\otimes$, $\oplus$, max and min. As for ASP, $\neg a$ is essentially seen as a new literal, which has no explicit connection to a, except for the fact that answer sets containing both a and $\neg a$ "to a sufficiently high degree" will be designated as inconsistent. As before, (strongly) negated literals are defined as follows: $\neg l := \neg a$ if $l = a$ and $\neg l := a$ if $l = \neg a$ (with a an atom). An expression of the form $\text{not}\, l$ (with l a literal) will be called a *negation-as-failure literal*.

We refer to the rule by its label r and $g(a_1, \ldots, a_n)$ is called the *head* r_h of r and $f(b_1, \ldots, b_m, \text{not}\, c_1, \ldots, \text{not}\, c_k)$ is called the *body* r_b of r. Each rule not containing negation-as-failure nor strong negation can be seen as an implication $r_b \rightarrow r_h$ in Łukasiewicz logic. This formula is satisfied by a fuzzy interpretation I iff $I(r_b) \leq I(r_h)$. As will be become clear in this section an arbitrary rule has this same intuition: the truth degree of the head must be greater or equal to the truth degree of the body. Rules of the form $\overline{c} \leftarrow \alpha$ with $\overline{c}$ a constant are called *constraints*. Intuitively such a rule implies that the truth value of α cannot be greater than c. On the other hand, a rule $\alpha \leftarrow \overline{c}$ is called a *fact*. The truth value of α must be at least c. As for ASP, we will consider several classes of FASP programs. If a regular FASP program has at most one literal in the head of each rule, it is called a *regular normal* FASP program and if in addition it does not contain negation-as-failure, it is called a *regular definite* FASP program. Finally, if a regular definite FASP program does not contain strong negation and has exactly one atom in the head of each rule, it is called a *regular simple* FASP program. Regular FASP programs only containing rules of the form

$$a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m \otimes \text{not}\, c_1 \otimes \ldots \otimes \text{not}\, c_k$$

are called *strict disjunctive* FASP programs. If a strict disjunctive FASP program has at most one literal in the head of each rule, it is called a *strict* *normal* FASP program and if a strict normal FASP program does not contain negation-as-failure, it is called a *strict definite* FASP program. A strict definite FASP program not containing strong negation and with exactly one atom in the head of each rule is called a *strict simple FASP program*.

A *fuzzy interpretation* I of a regular FASP program P is any consistent element $I \in \mathcal{F}(\mathcal{L}_P)$, where consistent means that $I(l) + I(\neg l) \leq 1$ for each $l \in \mathcal{L}_P$ with

$$\mathcal{L}_P = \{l \mid l \text{ literal in } P\} \cup \{\neg l \mid l \text{ literal in } P\}.$$

We use this definition of consistency because we want to generalize the classical definition of consistency from Section 2.1. Hence we want to have $I(l) \otimes I(\neg l) = 0$ for each answer set I and each literal l. This is equivalent to $I(l) + I(\neg l) \leq 1$. Note that this definition of consistency coincides with the approach in [11]. The set of all atoms in P is denoted by $\mathcal{B}_P$.

A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is extended to rules as follows:

- $[\overline{c}]_I = c$
- $[\text{not}\, l]_I = 1 - I(l)$
- $[f(\alpha, \beta)]_I = \mathtt{f}([\alpha]_I, [\beta]_I)$ where f is a prefix notation for $\otimes$, $\oplus$, max or min and $\mathtt{f}$ is the corresponding function defined on $[0, 1]$ (see Section 2.2)
- $[\alpha \leftarrow \beta]_I = \min(1 - [\beta]_I + [\alpha]_I, 1)$

with $\overline{c}$ a constant, l a literal and α and β relevant expressions. Remark that $[\alpha \leftarrow \beta]_I = 1$ iff $[\alpha]_I \geq [\beta]_I$.

A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is a *fuzzy model* of a regular FASP program P if $[r]_I = 1$ for each rule $r \in P$. For $I_1, I_2 \in \mathcal{F}(\mathcal{L}_P)$ we write $I_1 \leq I_2$ iff $I_1(l) \leq I_2(l)$ for each $l \in \mathcal{L}_P$. A fuzzy model I of P is a *minimal fuzzy model* if there exists no fuzzy model J of P such that $J < I$, i.e. $J \leq I$ and $J \neq I$. A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is called an *answer set* of a regular FASP program P without negation-as-failure if it is a minimal fuzzy model of P. Remark that such a regular FASP program can have none, one or several answer sets [30]. However, similar as for ASP, a regular simple FASP program P has exactly one answer set which coincides with the least fixpoint of the immediate consequence operator Π_P [8]. This operator maps fuzzy interpretations to fuzzy interpretations and is defined as

$$\Pi_P(I)(a) = \sup\{[r_b]_I \mid (a \leftarrow r_b) \in P\},$$

for an atom $a \in \mathcal{B}_P$ and $I \in \mathcal{F}(\mathcal{B}_P)$. Remark that an atom not occurring the head of any rule in a regular simple FASP program will always have truth value 0 in the answer set.

For programs with negation-as-failure, a generalization of the Gelfond-Lifschitz reduct [29] is used. In particular, for a program P and a fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ the reduct P^I of P w.r.t. I is obtained by replacing in each rule $r \in P$ all expressions of the form $\text{not}\, l$ by the interpretation $\overline{[\text{not}\, l]_I}$. For a literal l, we write $l^I = l$ and $(\text{not}\, l)^I = \overline{[\text{not}\, l]_I}$ and inductively we write α^I for a head or a body α of a rule and r^I for a rule r in which all expressions of the form $\text{not}\, l$ have been replaced by the interpretation $\overline{[\text{not}\, l]_I}$. This new program $P^I = \{r^I \mid r \in P\}$ is a regular negation-as-failure free FASP program and I is called an answer set of P if I is an answer set of P^I.

Let us reconsider the example from Section 1.

Example 2. *Consider the strict normal FASP program P*

$$\begin{aligned} b &\leftarrow \text{not}\, a \\ a &\leftarrow \text{not}\, b \end{aligned}$$

with a and b atoms. We show that for each $x \in [0,1] \cap \mathbb{Q}$, M_x with $M_x(a) = x$ and $M_x(b) = 1 - x$, is an answer set of P. We first compute the reduct P^{M_x}:

$$\begin{aligned} b &\leftarrow \overline{1 - x} \\ a &\leftarrow \overline{x} \end{aligned}$$

The minimal model of P^{M_x} is then exactly M_x. Note that there are infinitely many answer sets.

3. Motivating examples

In this section we present some motivating and illustrating examples for FASP. The first example shows how strict disjunctive FASP can be used to model sensor networks. This is followed by an example showing how strict simple FASP can be used to compute transitive closures of proximity relations. A version of the ATM location selection problem and a fuzzy graph coloring problem will be tackled using regular normal FASP.

3.1. Sensor networks

Forest fires cause massive loss of vegetation and animal life. If a fire is detected on time, suppression units are able to reach the fire in its initial stages which is important to avoid huge losses. Moreover suppression costs will be considerably reduced. Wireless sensor networks can be effectively used for this purpose [31]. These networks consist of a number of devices that can sense their environment and communicate wirelessly. Consider such a wireless sensor network consisting of sensors measuring temperature. Since there could be sensors that are defective, one should not blindly draw conclusions based on the measurements of the sensors. We will tackle this as follows. Sensors located near to each other should measure similar temperatures. Hence if such a couple of sensors displays significantly different temperatures, we can assume there must be something wrong with at least one of these sensors. We will use FASP to determine whether there are sensors which are not working optimally and if so, within what range we can assume the real temperature to be.

Suppose we have n sensors. By assuming an appropriate linear rescaling, we can see temperature as a value in $[0, 1] \cap \mathbb{Q}$. Although we might not be able to derive an exact temperature, we will try to find a subinterval of $[0, 1] \cap \mathbb{Q}$ in which we could assume the temperature to be. More specifically, for each sensor $i \in \{1, \ldots, n\}$, we denote the lower bound on the exact temperature as the variable t_i. The temperature measured by sensor i is a fixed value $t'_i \in [0, 1] \cap \mathbb{Q}$. If e_i is the variable representing the error on the measured temperature then the actual temperature must be in the interval $[t_i = t'_i - e_i, t'_i + e_i]$. In our setting, the fixed value t'_i corresponds to a constant $\overline{t'_i}$ in our FASP program and e_i is a variable for which we will infer a value, i.e. the measured temperature is considered given, and we learn a value for the measurement error that is potentially caused by a sensor that is not functioning as it should.

The sensor network defines a weighted graph G as follows. The vertices are the sensors and there is an edge with weight $w_{ij} \in [0, 1] \cap \mathbb{Q}$ between the vertices corresponding to sensor i and sensor j, indicating how near these sensors are to each other. The fixed value $w_{ij} \in [0, 1] \cap \mathbb{Q}$ is such that we can reasonably assume, based on the locations of sensors i and j that the difference of the exact temperature between these locations should be less than w_{ij}. So the degree to which we can assume that there is something wrong with sensors i and/or j is equal to the degree to which $d(t'_i, t'_j) = |t'_i - t'_j|$ is greater or equal to w_{ij}.

We can now write the following program P for given (fixed) values $t'_i, w_{ij} \in [0,1] \cap \mathbb{Q}$ and variables t_i, e_i $(i,j \in \{1,\ldots,n\})$.

$$\begin{array}{rrcl}
r_1: & \neg t_i \oplus \overline{t'_i} & \leftarrow & \text{not}\, e_i \\
r_2: & \overline{1 - t'_i} \oplus t_i & \leftarrow & \text{not}\, e_i \\
r_3: & t_i & \leftarrow & \text{not}\, \neg t_i \\
r_4: & \neg t_i & \leftarrow & \text{not}\, t_i \\
r_5: & e_i \oplus e_j & \leftarrow & d(\overline{t'_i}, \overline{t'_j}) \otimes \overline{1 - w_{ij}}
\end{array}$$

Rules r_1 and r_2 are obtained as follows. We want to model that the lower bound on the actual temperature t_i and the measured temperature t'_i are similar to the degree that we do not know that there is something wrong with sensor i. Hence we want to model the formula $(t_i \leftrightarrow \overline{t'_i}) \leftarrow \text{not}\, e_i$, where $\overline{t'_i}$ is the constant representing the measured temperature in the program. The attentive reader has likely noticed that this "rule" does not adhere to the syntax of (strict) disjunctive FASP. However it can be easily rewritten as the two syntactically correct FASP rules r_1 and r_2. Indeed, notice that in Łukasiewicz logic we have $t_i \leftrightarrow \overline{t'_i} = \min({\sim}\, t_i \oplus \overline{t'_i}, \overline{1 - t'_i} \oplus t_i)$. Hence for a fuzzy interpretation I such that $I(\neg t_i) = I({\sim}\, t_i) = 1 - I(t_i)$, we have that I models $(t_i \leftrightarrow \overline{t'_i}) \leftarrow \text{not}\, e_i$ iff $[\min(\neg t_i \oplus \overline{t'_i}, \overline{1 - t'_i} \oplus t_i)]_I \geq [\text{not}\, e_i]_I$ iff $[\neg t_i \oplus \overline{t'_i}]_I \geq [\text{not}\, e_i]_I$ and $[\overline{1 - t'_i} \oplus t_i]_I \geq [\text{not}\, e_i]_I$ and hence iff I models r_1, r_2, r_3 and r_4 where rules r_3 and r_4 are needed to obtain that $I(t_i) + I(\neg t_i) = 1$. Recall that, contrary to Łukasiewicz logic, this is not automatically valid in FASP.

Rule r_5 is justified by that fact that for a fuzzy interpretation I it holds that $[d(\overline{t'_i}, \overline{t'_j}) \otimes \overline{1 - w_{ij}}]_I = \max(d(t'_i, t'_j) - w_{ij}, 0)$. If $d(t'_i, t'_j) \leq w_{ij}$ and no other rules imply $I(e_1) > 0$ or $I(e_2) > 0$, then for an answer set I we obtain $[e_i \oplus e_j]_I = 0$, i.e. there is nothing wrong with the sensors. Otherwise, if $d(t'_i, t'_j) > w_{ij}$, then $[e_i \oplus e_j]_I \geq d(t'_i, t'_j) - w_{ij}$, i.e. there is something wrong with the sensors at least to the degree to which $d(t'_i, t'_j)$ is greater or equal to w_{ij}.

One can easily show that the solutions to the sensor network problem correspond to the answer sets of this program.

Consider as a concrete example a network with three sensors as depicted in Figure 1. Suppose we have the measurements $t'_1 = 0.4$, $t'_2 = 0.9$ and $t'_3 = 0.5$ and we have $w_{1,2} = w_{1,3} = w_{2,3} = 0.2$, i.e. all the sensors are fairly far apart from each other.

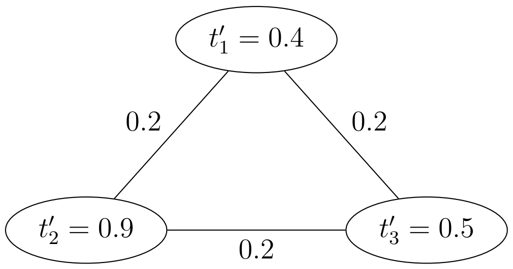

Figure 1: Example of a sensor network

The three rules of type r_5 are the following:

$$\begin{array}{rcl} e_1 \oplus e_2 & \leftarrow & 0.3 \\ e_2 \oplus e_3 & \leftarrow & 0.2 \\ e_1 \oplus e_3 & \leftarrow & 0 \end{array}$$

These rules impose lower bounds on $e_i \oplus e_j$ and by computing reducts P^I w.r.t. fuzzy interpretations I meeting these conditions and verifying that I is a minimal fuzzy model of P^I, we obtain for instance the following answer set I.

$$I(e_1) = 0.3, I(e_2) = 0, I(e_3) = 0.2, I(t_1) = 0.1, I(t_2) = 0.9, I(t_3) = 0.3,$$

$$I(\neg t_1) = 0.9, I(\neg t_2) = 0.1, I(\neg t_3) = 0.7$$

In particular, in this example, P^I is the program containing the following rules.

$$\begin{array}{rclrcl} \neg t_1 \oplus \overline{0.4} & \leftarrow & \overline{0.7} & \overline{0.6} \oplus t_1 & \leftarrow & \overline{0.7} \\ \neg t_2 \oplus \overline{0.9} & \leftarrow & \overline{1} & \overline{0.1} \oplus t_2 & \leftarrow & \overline{1} \\ \neg t_3 \oplus \overline{0.5} & \leftarrow & \overline{0.8} & \overline{0.5} \oplus t_3 & \leftarrow & \overline{0.8} \\ t_1 & \leftarrow & \overline{0.1} & \neg t_1 & \leftarrow & \overline{0.9} \\ t_2 & \leftarrow & \overline{0.9} & \neg t_2 & \leftarrow & \overline{0.1} \\ t_3 & \leftarrow & \overline{0.3} & \neg t_3 & \leftarrow & \overline{0.7} \\ e_1 \oplus e_2 & \leftarrow & \overline{0.3} & e_1 \oplus e_3 & \leftarrow & \overline{0} \\ e_2 \oplus e_3 & \leftarrow & \overline{0.2} & & & \end{array}$$

One can easily check that I is indeed a minimal fuzzy model of P^I.

Another answer set J is defined as follows.

$$J(e_1) = 0, J(e_2) = 0.3, J(e_3) = 0, J(t_1) = 0.4, J(t_2) = 0.6, J(t_3) = 0.5,$$

$$J(\neg t_1) = 0.6, J(\neg t_2) = 0.4, J(\neg t_3) = 0.5$$

Notice that there are several answer sets, each corresponding to a possible explanation.

3.2. Transitive closure

A proximity relation on a universe X is a mapping $R : X \times X \to [0,1] \cap \mathbb{Q}$ that is reflexive ($R(x,x) = 1$ for each $x \in X$) and symmetric ($R(x,y) = R(y,x)$ for all $x, y \in X$). A proximity relation R is not necessarily transitive ($R(x,y) \star R(y,z) \leq R(x,z)$ for all $x, y, z \in X$ where $x \star y = \max(x+y-1, 0)$ denotes the semantics for the Łukasiewicz conjunction). A strict simple FASP program can be used to find the transitive closure of R. This is the minimal mapping $\hat{R} : X \times X \to [0,1] \cap \mathbb{Q}$ that is reflexive, symmetric and transitive such that $R(x,y) \leq \hat{R}(x,y)$ for all $(x,y) \in X \times X$. Finding such "transitive approximations" is useful in many artificial intelligence areas, e.g. in fuzzy clustering [32] and analogue problems need to be solved in fuzzy spatial reasoning [33].

The corresponding FASP program simply consists of the rules

$$\hat{R}(x,z) \leftarrow \hat{R}(x,y) \otimes \hat{R}(y,z)$$

for all $x, y, z \in X$ and facts of the form

$$\hat{R}(x,y) \leftarrow \overline{R(x,y)},$$

where $\overline{R(x,y)}$ is the symbol representing the fixed value $R(x,y)$.

3.3. ATM location selection problem

The FASP program presented below is based on the ATM location selection problem for which a corresponding FASP program is given and discussed in [35]. Here, the problem is slightly modified and the resulting program is more concise. This problem is often referred to as the k-center problem and it is shown to be NP-hard [34].

Suppose we want to place k ATM machines $\{a_1, \ldots, a_k\}$ on roads connecting m towns such that the distance between each town and the closest ATM machine is less than a particular distance. Schematically, this can be seen as an undirected weighted graph $G = (\mathit{Towns}, \mathit{Edges})$ where $\mathit{Towns} = \{t_1, \ldots, t_m\}$ is the set of towns and $e_{t_i t_j}$ is an edge if there is a road connecting towns t_i and t_j. Note that $e_{t_i t_j} \in \mathit{Edges}$ iff $e_{t_j t_i} \in \mathit{Edges}$. A weight is given to an edge $e_{t_i t_j}$ in function of the distance between towns t_i and t_j. To obtain a weight that is an element in $[0,1] \cap \mathbb{Q}$, one can use a normalized distance $d : \mathit{Towns} \times \mathit{Towns} \rightarrow [0,1] \cap \mathbb{Q}$, e.g. the actual distance between two towns divided by the sum of all distances between all possible pairs of towns[2]. Suppose such a normalized distance function d is given, then we can define a normalized nearness function $n = 1 - d$. By using the Łukasiewicz conjunction we can perform summations of degrees of nearness. Indeed, suppose $n_1 = 1 - d_1 \in [0,1] \cap \mathbb{Q}$ and $n_2 = 1 - d_2 \in [0,1] \cap \mathbb{Q}$, then for a fuzzy interpretation I we have

$$\begin{aligned}
[\overline{n_1} \otimes \overline{n_2}]_I &= \max(n_1 + n_2 - 1, 0) \\
&= \max(1 - d_1 + 1 - d_2 - 1, 0) \\
&= \max(1 - (d_1 + d_2), 0) \\
&= 1 - \min(d_1 + d_2, 1) \\
&= 1 - [\overline{d_1} \oplus \overline{d_2}]_I
\end{aligned}$$

where $\overline{n_1}, \overline{n_2}, \overline{d_1}, \overline{d_2}$ are the symbols representing the values $n_1, n_2, d_1, d_2 \in [0,1] \cap \mathbb{Q}$.

We can now specify a program whose answer sets correspond to those configurations of ATMs such that the distance from each town to the nearest ATM is at most a particular degree $d' \in [0,1] \cap \mathbb{Q}$.

The first part of the program consists of the facts that define the graph. Specifically, we have a set of rules denoting which towns are connected by a single road and how near they are to each other:

$$\mathrm{edge}(e_{t_i t_j}) \leftarrow \overline{n(t_i, t_j)}$$

for each edge $e_{t_i t_j} \in \mathit{Edges}$. Secondly, for each edge one can (arbitrarily) designate one of the towns to be the starting point and the other one to be the ending point. This choice has no influence on the outcome of the program:

$$\begin{array}{lcl}
\mathrm{start}(t_i, e_{t_i t_j}) & \leftarrow & \overline{1} \\
\mathrm{end}(t_j, e_{t_i t_j}) & \leftarrow & \overline{1}
\end{array}$$

for each edge $e_{t_i t_j} \in \mathit{Edges}$.

The second part of the program consists of rules generating eligible solutions. For each $a \in \{a_1, \ldots, a_k\}$ and each edge $e \in \mathit{Edges}$ we add the

[2]If two cities are not connected directly, the distance of the shortest path is taken.

following rules

$$\begin{array}{lll}
r_1: & \mathrm{loc}(a,e) & \leftarrow \mathrm{loc}(a,e) \oplus \mathrm{loc}(a,e) \\
r_2: & \mathrm{loc}(a,e) & \leftarrow \otimes\{\mathrm{not}\,\mathrm{loc}(a,e') \mid e' \in \mathit{Edges}, e' \neq e\} \\
r_3: & \mathrm{locnearend}(a,e) & \leftarrow (\mathrm{edge}(e) \oplus \mathrm{not}\,\mathrm{locnearstart}(a,e)) \otimes \mathrm{loc}(a,e) \\
r_4: & \mathrm{locnearstart}(a,e) & \leftarrow (\mathrm{edge}(e) \oplus \mathrm{not}\,\mathrm{locnearend}(a,e)) \otimes \mathrm{loc}(a,e)
\end{array}$$

Rule r_1 is used to ensure that the truth degree of $\mathrm{loc}(a,e)$ is in $\{0,1\}$, i.e. an ATM a is located on an edge e or not. Indeed, a fuzzy interpretation I models this rule if and only if $\min(2I(\mathrm{loc}(a,e)),1) \leq I(\mathrm{loc}(a,e))$, i.e. $I(\mathrm{loc}(a,e)) \leq 0$ or $I(\mathrm{loc}(a,e)) \geq 1$. By using negation-as-failure, rule r_2 then generates all possible configurations (cfr. Example 2). If an ATM a is located on an edge e, then rules r_3 and r_4 determine how near to the start and the end of e it is located. Indeed, in terms of distances we want to model the following. Suppose e is the edge between t_i and t_j on which an ATM a is located, then it should hold that

$$d(t_i,a) + d(a,t_j) = d(t_i,t_j).$$

Hence in terms of nearness we want

$$1 - n(t_i,a) + 1 - n(a,t_j) = 1 - n(t_i,t_j)$$

or

$$n(t_i,a) + n(a,t_j) - 1 = n(t_i,t_j).$$

Thus for a fuzzy interpretation I we want

$$I(\mathrm{locnearend}(a,e)) + I(\mathrm{locnearstart}(a,e)) - 1 = I(\mathrm{edge}(e)).$$

This can be modelled by rules r_3 and r_4 since, assuming that $\mathrm{loc}(a,e)$ has truth value 1, a fuzzy interpretation I models r_3 if

$$I(\mathrm{locnearend}(a,e)) \geq \min(I(\mathrm{edge}(e)) + 1 - I(\mathrm{locnearstart}(a,e)),1).$$

Analogously for rule r_4 we obtain

$$I(\mathrm{locnearstart}(a,e)) \geq \min(I(\mathrm{edge}(e)) + 1 - I(\mathrm{locnearend}(a,e)),1).$$

Hence for an answer set I we obtain by the minimality condition that

$$I(\mathrm{locnearend}(a,e)) + I(\mathrm{locnearstart}(a,e)) - 1 = I(\mathrm{edge}(e)).$$

Note that if $\min(I(\text{edge}(e)) + 1 - I(\text{locnearstart}(a, e)), 1) = 1$, then by rule r_3 we would have $I(\text{locnearend}(a, e)) = 1$ and hence by rule r_4 that $I(\text{locnearstart}(a, e)) = I(\text{edge}(e))$ and we obtain the same result.

The following rules define the maximal nearness and hence the shortest distance to an ATM for a town $t \in \mathit{Towns}$. In particular, r_7 and r_8 define the shortest distance to an ATM if the town is not the start or end point of an edge that contains an ATM.

$$\begin{array}{lrcl} r_5: & \text{ATMnear}(t) & \leftarrow & \text{start}(t, e) \otimes \text{locnearstart}(a, e) \otimes \text{loc}(a, e) \\ r_6: & \text{ATMnear}(t) & \leftarrow & \text{end}(t, e) \otimes \text{locnearend}(a, e) \otimes \text{loc}(a, e) \\ r_7: & \text{ATMnear}(t) & \leftarrow & \text{edge}(e) \otimes \text{ATMnear}(t') \otimes \text{start}(t, e) \otimes \text{end}(t', e) \\ r_8: & \text{ATMnear}(t) & \leftarrow & \text{edge}(e) \otimes \text{ATMnear}(t') \otimes \text{end}(t, e) \otimes \text{start}(t', e) \end{array}$$

for each $t, t' \in \mathit{Towns}$, $e \in \mathit{Edges}$ and $a \in \{a_1, \ldots, a_k\}$.

Finally constraints are needed to indicate the minimal nearness $n' = 1-d'$ allowed in a valid configuration of ATMs.

$$r_9 : \overline{0} \leftarrow \text{not}\,\text{ATMnear}(t) \otimes \overline{n'}$$

for each $t \in \mathit{Towns}$. Indeed, a fuzzy interpretation I models r_9 if

$$I(\text{ATMnear}(t)) \geq n'.$$

The explanations above show that each answer set has the properties the solutions of the original search problem must have. On the other hand, we also have that each solution, seen as a fuzzy interpretation I, corresponds to an answer set. It has to be checked that I is a minimal fuzzy model of P^I. Rules r_1^I and r_2^I in P^I are modelled by I since these rules generate all possible placements of the ATMs on the roads and I corresponds to one particular configuration of ATMs. Rules r_3^I and r_4^I are also modelled since these rules compute the exact location on the edge of each ATM. An explanation similar as above ensures that I is minimal such that these rules are modelled. Rules $r_5^I - r_8^I$ compute the shortest distance to an ATM for each town, hence these rules must be modelled in a minimal way by I. Since I corresponds to a configuration such that for each town the distance to the closest ATM is less then $d' = 1 - n'$, rule r_9^I is also modelled.

Consider as a concrete example the following setting. Suppose there are two ATMs a_1 and a_2 and three towns t_1, t_2 and t_3 such that $n(t_1, t_2) = 0.8$, $n(t_1, t_3) = 0.3$ and $n(t_2, t_3) = 0.1$. Suppose we are interested in placing these

ATMs such that the minimal nearness n' is equal to 0.3. One of the answer sets is given in Figure 2. ATM a_1 is placed in t_3 and hence a_1 is near t_1 with degree 0.3 and near t_3 with degree 1. ATM a_2 is placed on the road between t_2 and t_3 with nearness degree 0.8 to t_2 and nearness degree 0.3 to t_3. Indeed the answer set I corresponding to this setting is a minimal model of P^I.

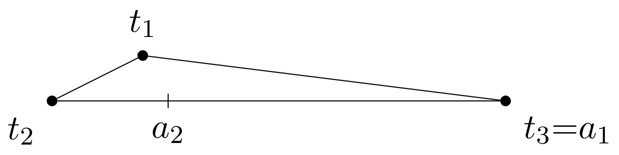

Figure 2: Configuration of ATMs

Another possible solution (Figure 3) would be to place a_1 in town t_3 and a_2 on the road connecting towns t_1 and t_2, for instance such that a_2 is near t_1 and near t_2 with degree 0.9.

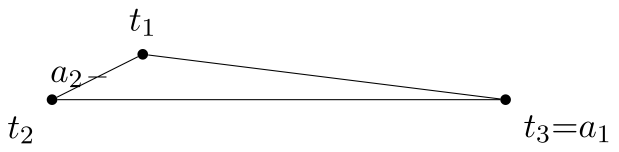

Figure 3: Configuration of ATMs

Note that we can also impose different degrees of nearness for different towns.

3.4. Fuzzy graph coloring problem

Using FASP, a continuous variant of the graph coloring problem can easily be defined. Recall that the classical graph coloring problem consists of coloring the nodes of a graph using a finite number of colors such that two nodes which are connected by an edge have a different color.

Now assume a weighted graph is given, specified by rules

$$\text{edge}(a, b) \leftarrow \overline{c}$$

with $c \in [0, 1] \cap \mathbb{Q}$ and where edge(a, b) represents the weight of the edge between nodes a and b. The problem consists of assigning grey values to each node in the graph such that the difference between the grey values is at least as large as the corresponding edge weight. Besides the above facts, we also need a generating part: in each answer set we want to have that a node is black to the degree that is not white (cfr. Example 2). Hence for each node a we add the rules.

$$\begin{aligned} r_1: \quad \text{black}(a) &\leftarrow \text{not white}(a) \\ r_2: \quad \text{white}(a) &\leftarrow \text{not black}(a) \end{aligned}$$

We also need a connective that denotes how similar the color of two nodes is. An obvious choice would be $\leftrightarrow$. Moreover note that for a fuzzy interpretation I we have $[\alpha \leftrightarrow \beta]_I = 1 - |[\alpha]_I - [\beta]_I|$. By the definitions of the Łukasiewicz connectives, we have that $\alpha \leftrightarrow \beta = (\sim \alpha \oplus \beta) \otimes (\sim \beta \oplus \alpha)$. By rules r_1 and r_2 we have that for each answer set I it must hold that $I(\text{white}(a)) = I(\sim \text{black}(a))$ for each node a. Hence we can write the following rules for each pair of nodes a and b

$$r_3: \quad \text{sim}(a,b) \leftarrow (\text{white}(a) \oplus \text{black}(b)) \otimes (\text{white}(b) \oplus \text{black}(a))$$

Finally, we need constraints to filter out the unwanted assignments:

$$r_4 : \overline{0} \leftarrow \text{edge}(a,b) \otimes \text{sim}(a,b)$$

for each pair of nodes a and b. Indeed, for a fuzzy interpretation I we have that I models r_4 iff $I(\text{edge}(a,b)) + I(\text{sim}(a,b)) - 1 \leq 0$ iff $I(\text{edge}(a,b)) \leq 1 - I(\text{sim}(a,b))$.

Consider as example the graph consisting of four nodes a, b, c and d with edge weights as depicted in Figure 4. Edges with weight 0 have been omitted. One possible colouring, as shown in the picture, is an answer set I such that

$$\begin{array}{llll} I(\text{black}(a)) = 0 & I(\text{black}(b)) = 0.1 & I(\text{black}(c)) = 0.5 & I(\text{black}(d)) = 0.8 \\ I(\text{white}(a)) = 1 & I(\text{white}(b)) = 0.9 & I(\text{white}(c)) = 0.5 & I(\text{white}(d)) = 0.2 \end{array}$$

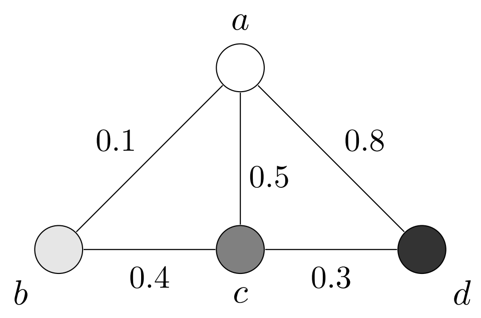

Figure 4: Fuzzy graph coloring

4. Complexity of FASP

In the remainder of the paper, we will study the complexity of the decision problems discussed in the introduction for regular and strict FASP. Note that these decision problems are generalizations of the ones for ASP for which the complexity is given in Table 1. In what follows we will often use the lemmas and remarks presented in this section. The proofs of all lemmas and propositions can be found in the appendix.

Using the fact that $I(a)+I(\neg a) \leq 1$ iff $[\overline{0} \leftarrow a \otimes \neg a]_I = 1$, we will show in Lemma 1 that a regular FASP program can be rewritten as a regular FASP program without strong negation.

Lemma 1. *Let P be a regular FASP program. There exists a regular FASP program P' without strong negation such that a fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is an answer set of P iff there exists an answer set $I' \in \mathcal{F}(\mathcal{L}_{P'})$ of P' such that for each atom $a \in \mathcal{B}_P$ we have $I(a) = I'(a)$ and $I(\neg a) = I'(a')$ for some $a' \in \mathcal{B}_{P'}$.*

Lemma 2. *Let P be a regular FASP program such that $P = P' \cup C$ where C is a set of constraints in P and $I \in \mathcal{F}(\mathcal{L}_P)$. It holds that I is an answer set of P iff I is an answer set of P' and a fuzzy model of C.*

Remark 2. *Recall that a regular simple FASP program, i.e. a regular definite FASP program with exactly one atom in the head of each rule and no strong negation, has a unique answer set. Hence the complexity of the set-membership problem and the set-entailment problem are equal and the complexity of the existence problem is "constant" for regular simple FASP. Moreover, note that by Lemmas 1 and 2, it follows that if the answer set of a certain type of regular simple FASP programs can be determined in polynomial time, then the complexity of the decision problems for the corresponding types of regular definite FASP is polynomial as well. Indeed, each regular definite FASP program P can be rewritten as $P' \cup C$ where P' is a regular simple FASP program and C is a set of constraints such that I is a answer set of P iff I is an answer set of P' and a fuzzy model of C. To check if P has an answer set, we have to compute the answer set of P' and check if it is a fuzzy model of C.*

Without loss of generality, we may assume that in each rule of a general FASP program, the body has exactly two arguments. Indeed, from the lemmas below it follows that a program can be rewritten as a program with only rules of the form $\alpha \leftarrow f(a,b)$ with a and b (negation-as-failure) literals and/or constants, $f(a,b)$ equal to either $a \otimes b$, $a \oplus b$, $\max(a,b)$ or $\min(a,b)$ and α an arbitrary head. The idea is to substitute expressions in the body by adding new rules with fresh atoms in the head of these rules. This can be done since an answer set is a minimal fuzzy model of the program (see e.g. [35] for a proof) and the functions representing the connectives allowed in the bodies of rules are increasing.

For a function $g : C \to D$, we will denote by $g_{|B}$ the restriction of g to the domain $B \subseteq C$, i.e. the function $g_{|B} : B \to D : x \mapsto g(x)$.

Lemma 3. *Let $P = P_1 \cup \{r\}$ be a regular FASP program where*

$$r : \beta \leftarrow f(l_1, \ldots, l_n)$$

with l_i (negation-as-failure) literals and/or constants, β an arbitrary head and $f(l_1, \ldots, l_n)$ denotes either $\otimes_{i=1}^n l_i$, $\oplus_{i=1}^n l_i$, $\max(l_1, \ldots, l_n)$ or $\min(l_1, \ldots, l_n)$.

For a fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$, it holds that I is an answer set of P iff there exists a fuzzy interpretation $I' \in \mathcal{F}(\mathcal{L}_{P'})$ such that $I'_{|\mathcal{L}_P} = I$ and I' is an answer set of P' where $P' = P_1 \cup P_2$ and P_2 is the program consisting of the rules

$$\begin{array}{lcl} b_1 & \leftarrow & f(l_1, l_2) \\ b_2 & \leftarrow & f(b_1, l_3) \\ & \vdots & \\ b_{n-2} & \leftarrow & f(b_{n-3}, l_{n-1}) \\ \beta & \leftarrow & f(b_{n-2}, l_n) \end{array}$$

with $b_1, \ldots, b_{n-2}$ atoms which are not used in P.

Lemma 4. *Let $P = P_1 \cup \{r\}$ be a regular FASP program where*

$$r : \beta \leftarrow f(\alpha_1, \ldots, \alpha_n)$$

with α_i formulas built from (negation-as-failure) literals and/or constants, $\otimes$, $\oplus$, $\max$, $\min$ and β an arbitrary head and $f(\alpha_1, \ldots, \alpha_n)$ denotes either $\otimes_{i=1}^n \alpha_i$, $\oplus_{i=1}^n \alpha_i$, $\max(\alpha_1, \ldots, \alpha_n)$ or $\min(\alpha_1, \ldots, \alpha_n)$.

For a fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$, it holds that I is an answer set of P iff there exists a fuzzy interpretation $I' \in \mathcal{F}(\mathcal{L}_{P'})$ such that $I'_{|\mathcal{L}_P} = I$ and I' is an answer set of P' where $P' = P_1 \cup P_2$ and P_2 is the program consisting *of the rules*

$$\begin{array}{rcl} a_1 & \leftarrow & \alpha_1 \\ a_2 & \leftarrow & \alpha_2 \\ & \vdots & \\ a_n & \leftarrow & \alpha_n \\ \beta & \leftarrow & f(a_1, \ldots, a_n) \end{array}$$

with $a_1, \ldots, a_n$ atoms which are not used in P.

Combining Lemmas 3 and 4, one can prove the following proposition; a FASP program can be rewritten as a set of rules with only two (negation-as-failure) literals or constants in the body such that the answer sets remain the same.

Proposition 1. *Let P be a regular FASP program. P can be reduced (in time polynomial in the size of the program) to a regular FASP program P' such that $\mathcal{L}_P \subseteq \mathcal{L}_{P'}$ and each rule in P' has at most two arguments in the body and I is an answer set of P iff there exists a fuzzy interpretation $I' \in \mathcal{F}(\mathcal{L}_{P'})$ such that $I'_{|\mathcal{L}_P} = I$ and I' is an answer set of P'.*

5. Complexity of strict FASP

In this section we will study the complexity for strict disjunctive FASP, i.e. regular FASP programs with rules of the form

$$a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m \otimes \text{not}\, c_1 \otimes \ldots \otimes \text{not}\, c_k$$

with a_i, b_j, c_k literals and/or constants $\overline{c}$ with $c \in [0,1] \cap \mathbb{Q}$.

We will first show that set-membership for strict disjunctive FASP is NP-complete. We will do this by showing NP-membership in Proposition 2 and by showing in Proposition 3 that it is already NP-hard for a subclass of strict disjunctive FASP; strict normal FASP. We will use these results to derive complexity results for the remaining decision problems for strict disjunctive FASP and strict normal FASP. We will then use these results to show that set-membership remains NP-complete in strict normal (and disjunctive) FASP even if constraints and strong negation are not allowed.

Proposition 2. *Set-membership for strict disjunctive FASP is in NP.*

Next, we show that the set-membership problem is also NP-hard by showing a reduction from 3SAT, which is NP-complete [36], to (a subclass of) strict disjunctive FASP. Recall that instances of the 3SAT problem are Boolean expressions written in conjunctive normal form with 3 variables in each clause:

$$(a_{11} \vee a_{12} \vee a_{13}) \wedge (a_{21} \vee a_{22} \vee a_{23}) \wedge \ldots \wedge (a_{n1} \vee a_{n2} \vee a_{n3}),$$

where each a_{ij} is an atom or a negated atom, i.e. a literal. The problem consists of deciding whether there exists a propositional interpretation that makes the Boolean expression true.

Proposition 3. *Set-membership for strict normal FASP is NP-hard.*

The following corollary follows directly from Propositions 2 and 3.

Corollary 1. 1. *Set-membership for strict normal FASP is NP-complete.*
2. *Set-membership for strict disjunctive FASP is NP-complete.*

The proofs of Propositions 2 and 3 do not exploit the fact that we want to find an answer set I such that $I(l) \geq \lambda_l$ for a particular λ_l. Hence these proofs can also be used to show NP-completeness for the existence problem.

Proposition 4. 1. *Existence for strict normal FASP is NP-complete.*
2. *Existence for strict disjunctive FASP is NP-complete.*

Finally, by the proofs of Propositions 2 and 3 and the results in Proposition 4 we can show the following proposition.

Proposition 5. 1. *Set-entailment for strict normal FASP is coNP-complete.*
2. *Set-entailment for strict disjunctive FASP is coNP-complete.*

We will now show that set-membership remains NP-complete for strict normal and disjunctive FASP even if strong negation and constraints are not allowed. This result is based on Proposition 6 which uses a lemma that enables us to simulate constraints in a regular FASP program. Note that this lemma is valid for more general FASP programs and not only for strict disjunctive FASP programs.

Lemma 5. *Consider a regular FASP program $P = P_1 \cup C$ where P_1 is a regular FASP program and C is a set of constraints of the form $\overline{0} \leftarrow \alpha$. Let $P' = P_1 \cup C' \cup \{z \leftarrow \text{not}\, y\}$ where z and y are fresh atoms and $C' = \{y \leftarrow \alpha \mid (\overline{0} \leftarrow \alpha) \in C\}$. A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is an answer set of P iff there exists an answer set $I' \in \mathcal{F}(\mathcal{L}_{P'})$ of P' such that $I'_{|\mathcal{L}_P} = I$ and $I'(z) \geq 1$.*

Proposition 6. *Set-membership for strict normal FASP is NP-hard even if constraints and strong negation are not allowed.*

Finally, we can derive the following corollaries:

Corollary 2.
1. *Set-membership for strict normal FASP is NP-complete, even if constraints and strong negation are not allowed.*
2. *Set-membership for strict disjunctive FASP is NP-complete, even if constraints and strong negation are not allowed.*

By using a result from [37][3] we can derive that a strict normal FASP program without constraints and without strong negation always has an answer set.

Proposition 7. *A strict normal FASP program without constraints and without strong negation always has an answer set.*

Proposition 7 combined with results from [38] concerning stratified fuzzy description logic programs implies that a strict normal FASP program for which the dependency graph does not contain cycles (see Section 6.1.1 for the exact definitions) has exactly one answer set.

As the following example shows, the result from Proposition 7 is not valid for regular FASP programs in which disjunction is allowed in the body of rules. For these types of FASP programs the existence of an answer set is not guaranteed, even if constraints and strong negation are not allowed.

Example 3. *Consider the following regular normal FASP program P.*

$$\begin{array}{lcll} p & \leftarrow & p \oplus p & (1) \\ q & \leftarrow & q \oplus q & (2) \\ p & \leftarrow & \text{not}\, p \otimes q & (3) \\ q & \leftarrow & \overline{c} & (4) \end{array}$$

[3]We would like to thank the reviewer for making us aware of this result.

with $c > 0$.

Suppose P has an answer set I. By the same reasoning as in Section 3.3, it follows that $I(p), I(q) \in \{0,1\}$. If $I(p) = 0$, then P^I is the regular simple FASP program

$$\begin{array}{rcl} p & \leftarrow & p \oplus p \\ q & \leftarrow & q \oplus q \\ p & \leftarrow & q \\ q & \leftarrow & \overline{c} \end{array}$$

which has as minimal fuzzy model $J(p) = J(q) = 1$. Thus $I \neq J$ is not an answer set of P. If $I(p) = 1$, then P^I is the regular simple FASP program

$$\begin{array}{rcl} p & \leftarrow & p \oplus p \\ q & \leftarrow & q \oplus q \\ p & \leftarrow & \overline{0} \\ q & \leftarrow & \overline{c} \end{array}$$

which has as minimal fuzzy model $J(p) = 0$, $J(q) = 1$. Thus $I \neq J$ is not an answer set of P.

For the class of strict disjunctive FASP where constraints and strong negation are not allowed we can only show NP-membership for the existence problem (follows from the proof of Proposition 2). For set-entailment we can only show coNP-membership for both strict normal and disjunctive FASP where constraints and strong negation are not allowed (see the proof of Proposition 5).

Finally, we will discuss the complexity for strict definite FASP. We will show that the decision problems are in P. To do this, by Remark 2 it is sufficient to prove that the unique answer set of a strict simple FASP program can be determined in polynomial time. In particular, we will show that for such programs, the unique answer set can be found in polynomial time using linear programming, which is known to be in P. Moreover, the complexity remains the same if maximum is allowed in the body of rules. Indeed, rules of the form $a \leftarrow b \otimes c$ are modelled by a fuzzy interpretation I iff $I(b)+I(c)-1 \leq I(a)$. Rules of the form $d \leftarrow \max(e, f)$ are modelled by a fuzzy interpretation I iff $I(e) \leq I(d)$ and $I(e) \leq I(f)$. Hence such a program can be efficiently translated to a linear program.

Example 4. *Consider the following program P.*

$$\begin{array}{rcl} a & \leftarrow & b \otimes \overline{\tfrac{1}{2}} \\ b & \leftarrow & \max(c, \overline{\tfrac{1}{3}}) \end{array}$$

The corresponding linear program contains the following constraints.

$$\begin{array}{rcl} a' & \geq & b' - \frac{1}{2} \\ b' & \geq & c' \\ b' & \geq & \frac{1}{3} \\ 1 & \geq & a', b', c' \\ a', b', c' & \geq & 0 \end{array}$$

and the function to be minimized is $f(a', b', c') = a' + b' + c'$. The solution $a' = 0$, $b' = \frac{1}{3}$, $c' = 0$ given by the linear program then corresponds to the answer set I of P: $I(a) = a'$, $I(b) = b'$ and $I(c) = c'$.

Proposition 8. *The unique answer set of a regular simple FASP program with only conjunction and maximum in the body of the rules can be found in polynomial time.*

Corollary 3. *The complexity of the decision problems for strict simple and strict definite FASP is polynomial.*

6. Complexity of regular FASP

In this section, we will investigate the complexity of the decision problems for regular FASP. From the complexity results for fuzzy equilibrium logic, which is a proper generalization of regular FASP [23], we can derive that existence and set-membership for regular FASP are in Σ_2^P and that set-entailment is in Π_2^P. By reducing the decision problems for disjunctive ASP to regular FASP (see Proposition 9), one can also derive resp. Σ_2^P-hardness and Π_2^P-hardness. Hence for regular FASP without any restrictions, we obtain Σ_2^P-completeness for existence and set-membership and Π_2^P-completeness for set-entailment (see Proposition 10).

Proposition 9. *Let P be a disjunctive ASP program and let $I \in \mathcal{P}(\mathcal{L}_P)$. Define the regular FASP program P' as follows:*

$$P' = \{a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m \otimes \text{not}\, c_1 \otimes \ldots \otimes \text{not}\, c_k \mid$$

$$(a_1 \vee \ldots \vee a_n \leftarrow b_1 \wedge \ldots \wedge b_m \wedge \text{not}\, c_1 \wedge \ldots \wedge \text{not}\, c_k) \in P\}$$

$$\cup \{a \leftarrow a \oplus a \mid a \in \mathcal{L}_P\}.$$

Then I is an answer set of P iff I is an answer set of P'.

We will use Proposition 9 to disjunctive ASP to regular FASP and normal ASP to regular normal FASP.

Proposition 10. 1. *Set-membership and existence for regular FASP are Σ_2^P-complete. Set-entailment is Π_2^P-complete.*

2. *Set-membership and existence for regular normal FASP are NP-hard and in Σ_2^P. Set-entailment is coNP-hard and in Π_2^P.*

In Section 6.1, we will discuss the complexity for regular simple and definite FASP programs. We show that characterizing the complexity for regular simple FASP programs is equivalent to an open problem about integer equations [39]. However, we can provide a pseudo-polynomial algorithm and show P-membership for several subclasses of regular definite FASP. We will then use these results in Section 6.2 to characterize the computational complexity for regular normal FASP in a more fine grained manner than in Proposition 10.

6.1. Complexity of regular definite FASP programs

In this section we will discuss the complexity results for programs consisting of rules of the form

$$a \leftarrow f(b_1, \ldots b_m)$$

with $a, b_1, \ldots b_m$ literals and/or constants $\overline{c}$ with $c \in [0,1] \cap \mathbb{Q}$ and f a composition of $\otimes$, $\oplus$, min and max. By Proposition 1, we can restrict ourselves to programs in which each rule has at most two arguments in the body.

Satisfiability checking in Łukasiewicz logic can be polynomially reduced to checking the feasibility of a mixed integer program [7]. As will be shown in Proposition 11, the NP-completeness of the latter decision problem [7] and the fact that each rule in a regular definite FASP program can be seen as a formula in Łukasiewicz logic, it follows that the decision problems for regular simple and thus also regular definite FASP programs are all in NP. Moreover, since the answer set of a regular simple FASP program is unique, we can also prove coNP-membership for regular definite FASP programs:

Proposition 11. *Set-membership, existence and set-entailment for regular definite FASP is in NP $\cap$ coNP.*

In general, to find the unique minimal fuzzy model of a regular simple FASP program P, one could use the immediate consequence operator Π_P (see

Section 2.3). The minimal fuzzy model of P then equals the least fixpoint of Π_P. This least fixpoint can be found by repeatedly applying the immediate consequence operator starting from the fuzzy interpretation $I_0 : \mathcal{B}_P \to [0,1] : a \mapsto 0$. Unfortunately, the number of iterations that is needed to arrive at the least fixpoint can be exponential in the number of bits needed to represent the rules. Consider for example the program consisting of the following rule, where n is equal to the "size" of the problem.

$$a \leftarrow a \oplus \overline{\left(\frac{1}{2^n}\right)}$$

In that case 2^n iterations of the immediate consequence operator are needed to conclude that a should have truth value 1. Indeed, one starts with the fuzzy interpretation $I_0 : \mathcal{L}_P \to [0,1]$ such that $I_0(a) = 0$. The next applications give us $I_1(a) = \frac{1}{2^n}$, $I_2(a) = \frac{2}{2^n}$, $I_3(a) = \frac{3}{2^n}$, $\ldots$, $I_{2^n}(a) = \frac{2^n}{2^n} = 1$. Hence 2^n iterations are needed. However, the number of iterations of the immediate consequence operator is polynomial in the size of the largest integer occurring in the program. As the following proposition shows, this will always be the case, i.e. we can find the unique answer set of any regular simple FASP program in pseudo-polynomial time.

Proposition 12. *The unique answer set of a regular simple FASP program can be found in pseudo-polynomial time.*

Proposition 13. *The complexity of the decision problems for regular simple and regular definite FASP is polynomial if all constants are polynomially bounded, i.e. all constants $\overline{c}$ in the program are such that $c \in \{0, \frac{1}{k}, \ldots, \frac{k}{k}\}$ with k polynomial in the size of the program.*

For the above program with rule

$$a \leftarrow a \oplus \overline{\left(\frac{1}{2^n}\right)}$$

we could improve the immediate consequence operator by assigning to a immediately truth value 1. It remains unclear, however, whether a general method could be found that always finds the answer set in polynomial time. The connection of this question to a well-known open problem on the feasibility of systems of integer equations suggests that there is not likely to be a straightforward solution. More precisely, the unique minimal fuzzy model

I of a regular simple FASP program P can be found by computing the least solution of a system of equations over the integers in which each equation is of the form $x_i = \alpha_i$ with the variables x_i on the left hand side pairwise distinct, i.e. x_i can only occur once as the left hand side of an equation, and the expressions α_i built from variables, integers, addition, multiplication with positive constants, maximum and minimum. The translation from P to such a system is done as follows. First, create a set $\hat{P}$ of Łukasiewicz formulas:

$$\hat{P} = \{r_b \to r_h \mid (r_h \leftarrow r_b) \in P\} \cup \{\max(a, \overline{0}) \to a \mid a \in \mathcal{B}_P\},$$

where we add tautologies of the form $\max(a, \overline{0}) \to a$ to ensure that each a obtains a positive value after translating to a system of equations over the integers. Next, create a new set $\hat{P}_1$ of Łukasiewicz formulas by replacing for each atom a in $\hat{P}$ the set of formulas with the same "head" a, $\alpha_1 \to a, \ldots \alpha_n \to a$ by the formula $\max(\alpha_1, \ldots, \alpha_n) \to a$. Finally, define the set $\hat{P}_2$ of Łukasiewicz formulas:

$$\hat{P}_2 = \{\alpha \leftrightarrow \beta \mid (\alpha \to \beta) \in \hat{P}_1\}.$$

We can now transform the set $\hat{P}_2$ to a set S of equations over the integers. First, define $\hat{S}$:

$$\hat{S} = \{\alpha = \beta \mid (\alpha \leftrightarrow \beta) \in \hat{P}_2\}$$

This is justified by the fact that $[\alpha \leftrightarrow \beta]_I = 1$ iff $[\alpha]_I = [\beta]_I$. Each constant in some equation in $\hat{S}$ can be assumed to be of the form $\overline{\left(\frac{i}{k}\right)}$ for a fixed k. Each such constant $\overline{\left(\frac{i}{k}\right)}$ is then replaced by i, $a \otimes b$ becomes $\max(a+b-k, 0)$ and $a \oplus b$ becomes $\min(a+b, k)$. This gives us the set S of equations over the integers.

There is a positive integer solution $J(a), J(b), J(c)$ for $a = \max(b+c-k, 0)$ iff the fuzzy interpretation I defined by $I(a) = \frac{J(a)}{k}, I(b) = \frac{J(b)}{k}, I(c) = \frac{J(c)}{k}$ is a fuzzy model of $a \leftrightarrow (b \otimes c)$:

$$\begin{aligned}
[a \leftrightarrow (b \otimes c)]_I = 1 &\Leftrightarrow \frac{J(a)}{k} = \frac{J(b)}{k} \otimes \frac{J(c)}{k} \\
&\Leftrightarrow \frac{J(a)}{k} = \max\left(\frac{J(b)}{k} + \frac{J(c)}{k} - 1, 0\right) \\
&\Leftrightarrow \frac{J(a)}{k} = \max\left(\frac{J(b)}{k} + \frac{J(c)}{k} - \frac{k}{k}, 0\right) \\
&\Leftrightarrow J(a) = \max(J(b) + J(c) - k, 0)
\end{aligned}$$

Similarly, one obtains that there is a positive integer solution $J(a), J(b), J(c)$ for $a = \min(b+c, k)$ iff $I(a) = \frac{J(a)}{k}, I(b) = \frac{J(b)}{k}, I(c) = \frac{J(c)}{k}$ models $a \leftrightarrow (b \oplus c)$. The unique least solution of S then corresponds to the unique minimal model of P in this sense. In [39], an algorithm is presented for computing least solutions of such systems of integer equations. Although in practice it turns out that the algorithm is very efficient, it is still an open problem (e.g. [40]) whether it has polynomial time complexity.

Example 5. *As an illustration, consider the regular simple FASP program consisting of the rules*

$$\begin{aligned} a &\leftarrow b \oplus \overline{\tfrac{1}{4}} \\ b &\leftarrow \min(a, \overline{\tfrac{1}{3}}) \\ a &\leftarrow \overline{\tfrac{1}{2}} \end{aligned}$$

The corresponding set $\hat{P}_2$ is

$$\begin{aligned} a &\leftrightarrow \max(b \oplus \overline{\tfrac{1}{4}}, \overline{\tfrac{1}{2}}, 0) \\ b &\leftrightarrow \max(\min(a, \overline{\tfrac{1}{3}}), 0) \end{aligned}$$

We then have that I is the minimal fuzzy model of P iff I is the minimal fuzzy model of $\hat{P}_2$ iff I' is the least solution of

$$\begin{aligned} a &= \max(\min(b+3, 12), 6, 0) \\ b &= \max(\min(a, 4), 0) \end{aligned}$$

with $I(a) = \frac{I'(a)}{12}$ and $I(b) = \frac{I'(b)}{12}$.

However, as we will show in the following subsections, for several subclasses of regular simple (and definite) FASP programs, we can show P-membership, even if the constants in the program are not polynomially bounded. A summary of the complexity results for regular simple FASP can be found in Table 3 in Section 1.

6.1.1. Directed graphs and cycles

For a regular simple FASP program P we define the dependency graph $G(P)$ as follows. The vertices are the atoms occurring in the program and there is a directed edge from a to b if a occurs in the body of a rule with head b.

Example 6. *Program P*

$$\begin{aligned} a &\leftarrow b \\ a &\leftarrow \overline{0.2} \\ c &\leftarrow a \otimes b \\ c &\leftarrow \overline{0.1} \\ b &\leftarrow \overline{0.6} \end{aligned}$$

has the following dependency graph $G(P)$.

$$\begin{tikzcd} a \arrow[rd] & b \arrow[l] \arrow[d] \\ & c \end{tikzcd}$$

A path in a directed graph is a sequence of vertices such that from each vertex there is an edge to the next vertex in the sequence. A cycle is a path that begins and ends at the same vertex. If there are no cycles in the dependency graph of a regular simple FASP program, the immediate consequence operator will only need a polynomial number of steps to compute the least fixpoint:

Proposition 14. *Consider a regular simple FASP program P such that the dependency graph $G(P)$ has no cycles and the longest path in $G(P)$ has length m. Then, the immediate consequence operator will only need m iterations to compute the answer set of P.*

Example 7. *Reconsider the following FASP program P from Example 6 with a cycle free dependency graph and longest path of length* 3.

$$\begin{aligned} a &\leftarrow b \\ a &\leftarrow \overline{0.2} \\ c &\leftarrow a \otimes b \\ c &\leftarrow \overline{0.1} \\ b &\leftarrow \overline{0.6} \end{aligned}$$

We apply the immediate consequence operator Π_P to find the unique answer set. We start from the fuzzy interpretation $I_0 : \mathcal{B}_\mathcal{P} \to [0,1] : l \mapsto 0$. After one application of Π_P, we obtain the fuzzy interpretation $I_1 = \Pi_P(I_0)$ which is such that $I_1(a) = 0.2$, $I_1(b) = 0.6$ and $I_1(c) = 0.1$. After one more *application, we have $I_2 = \Pi_P(I_1)$ which is such that $I_2(a) = 0.6$, $I_2(b) = 0.6$ and $I_2(c) = 0.1$. After the 3th application, the least fixpoint $I_3 = \Pi_P(I_2)$ of Π_P has been found: $I_3(a) = 0.6$, $I_3(b) = 0.6$ and $I_3(c) = 0.2$.*

6.1.2. Only disjunction in the body

For regular simple FASP programs with only disjunctions in the bodies of rules, we can always find the answer set in a polynomial number of steps, even if the dependency graph contains cycles and the constants are not polynomially bounded. In particular, if there is a cycle in the dependency graph of such a program such that there is a rule $c \leftarrow a \oplus b$ with a and c elements of the cycle and where b must have a truth value that is strictly positive, it follows that the truth values of all atoms in that cycle will saturate to 1.

Proposition 15. *Consider a regular simple FASP program P and its unique answer set I. Suppose P contains the following set of rules*

$$\begin{array}{rcl} a_2 & \leftarrow & a_1 \oplus b_1 \\ a_3 & \leftarrow & a_2 \oplus b_2 \\ & \vdots & \\ a_1 & \leftarrow & a_n \oplus b_n \end{array}$$

and we have $I(b_j) > 0$ for at least one $j \in \{1, \ldots, n\}$. Then for each $i \in \{1, \ldots, n\}$ we have $I(a_i) = 1$.

Remark 3. *If we have that $I(b_i) = 0$ for each $i \in \{1, \ldots, n\}$, then we still have $I(a_1) = \ldots = I(a_n) = c$ for some $c \in [0, 1]$. Moreover, if there are no other rules in P that have an atom a_i in the head of a rule, then $c = 0$.*

We can define an equivalence relation on the set of vertices of an arbitrary directed graph G as follows. Two vertices u and v are equivalent if there is a cycle in G containing both u and v. The corresponding equivalence classes V_i lead to subgraphs G_i which are defined as the restrictions of G to the vertices in V_i and the edges between the vertices in V_i. Each of these subgraphs G_i is strongly connected, i.e. for each two vertices u and v in G_i, there is a path from u to v. Moreover, no G_i is a proper subgraph of another strongly connected subgraph of G. The graphs G_i are called the *strongly connected components* of G and can be seen as generalizations of cycles. Using Proposition 15 we can prove the following proposition.

Proposition 16. *Consider a regular simple FASP program with only disjunctions in the bodies of rules and its unique answer set I. Suppose one of the rules in the program is of the form $a \leftarrow b \oplus d$ such that $I(b) > 0$ and a and d are atoms in the same strongly connected component S in $G(P)$. Then for all $s \in S$ we have $I(s) = 1$.*

Remark 4. *Similar as in Remark 3, we obtain that all the atoms in some strongly connected component must have the same truth value in the answer set.*

Given a regular simple FASP program P with only disjunctions in the bodies of rules, we can find the answer set in polynomial time as follows. Using the algorithm of Tarjan [41] the strongly connected components in $G(P)$ can be identified in polynomial time. Next, for each strongly connected component S a fresh atom a_S is introduced and each atom from S is replaced by a_S. By doing this substitutions it is possible that duplicate rules arise. A program P' is obtained by removing all rules that are already in the program, i.e. such that each rule occurs only once. Finally a program P'' is defined as follows. In this definition we will use formulas of the form $(a > 0)$ for which the semantics is defined as $I(a > 0) = 1$ if $I(a) > 0$ and $I(a > 0) = 0$ otherwise.

- Rules of the form $a \leftarrow b \oplus c$ in P' where a, b and c are different atoms or constants remain unchanged.
- A rule of the form $a \leftarrow a \oplus b$ in P' where $b \neq a$ with b an atom or a constant is replaced by $a \leftarrow (b > 0)$.
- A rule of the form $a \leftarrow a \oplus a$ in P' is replaced by $a \leftarrow (a' > 0)$ with a fresh atom a'. In all other rules every occurrence of a is replaced by a'.

The new program P'' is then a cycle free program. Since the semantics for formulas $(a > 0)$ is characterized by an increasing function, the immediate consequence operator can be used to compute the minimal model I'' of P'' (see [35]) which coincides with the answer set I of P in the sense that for each $a \in S$ we have $I(a) = I''(a_S)$. Since the proof of Proposition 14 does not rely on the fact that the FASP programs adhere to the Łukasiewicz semantics, we can use a very similar proof to show that the answer set of programs containing formulas of the form $(a > 0)$ in the body of rules will be found in polynomial time.

Corollary 4. *The unique answer set of a regular simple FASP program with only disjunction in the body of the rules can be found in polynomial time.*

Example 8. *Consider the following program P*

$$\begin{array}{rcl} a & \leftarrow & b \\ a & \leftarrow & a \oplus \overline{\frac{1}{2^n}} \\ b & \leftarrow & a \oplus c \\ c & \leftarrow & b \end{array}$$

with n a integer larger than the size of the program.

The program P has exactly one strongly connected component $S = \{a, b, c\}$. The corresponding program P' is

$$\begin{array}{rcl} a_S & \leftarrow & a_S \\ a_S & \leftarrow & a_S \oplus \overline{\frac{1}{2^n}} \\ a_S & \leftarrow & a_S \oplus a_S \end{array}$$

From P' we obtain the corresponding cycle free program P''

$$\begin{array}{rcl} a'_S & \leftarrow & (\overline{0} > 0) \\ a'_S & \leftarrow & (\overline{\frac{1}{2^n}} > 0) \\ a_S & \leftarrow & (a'_S > 0) \end{array}$$

To obtain the answer set of P, we then apply the immediate consequence operator to the program P''. We start from a fuzzy interpretation $I_0 : \mathcal{B}_{P''} \to [0,1] : a' \mapsto 0$. After one application of $\Pi_{P''}$ we obtain $I_1 = \Pi_{P''}(I_0)$ which is defined as follows: $I_1(a'_S) = 1$ and $I_1(a_S) = 0$. After one more application we obtain the least fixpoint $I_2 = \Pi_{P''}(I_1)$ where $I_2(a'_S) = I_2(a_S) = 1$. This fixpoint then coincides with the unique answer set I of P: $I(a) = I(b) = I(c) = I_2(a_S) = 1$.

Example 9. *Consider the following program P*

$$\begin{array}{rcl} a & \leftarrow & b \oplus c \\ b & \leftarrow & a \\ d & \leftarrow & c \\ c & \leftarrow & \overline{0.3} \\ c & \leftarrow & d \oplus e \\ a & \leftarrow & c \oplus e \end{array}$$

The program P has three strongly connected components $S_1 = \{a, b\}$, $S_2 = \{c, d\}$ and $S_3 = \{e\}$.Hence the corresponding program P' is

$$\begin{array}{lcl} a_{S_1} & \leftarrow & a_{S_1} \oplus a_{S_2} \\ a_{S_1} & \leftarrow & a_{S_1} \\ a_{S_2} & \leftarrow & a_{S_2} \\ a_{S_2} & \leftarrow & \overline{0.3} \\ a_{S_2} & \leftarrow & a_{S_2} \oplus a_{S_3} \\ a_{S_1} & \leftarrow & a_{S_2} \oplus a_{S_3} \end{array}$$

From P' we obtain the corresponding cycle free program P''

$$\begin{array}{lcl} a_{S_1} & \leftarrow & (a_{S_2} > 0) \\ a_{S_1} & \leftarrow & (\overline{0} > 0) \\ a_{S_2} & \leftarrow & (\overline{0} > 0) \\ a_{S_2} & \leftarrow & \overline{0.3} \\ a_{S_2} & \leftarrow & (a_{S_3} > 0) \\ a_{S_1} & \leftarrow & a_{S_2} \oplus a_{S_3} \end{array}$$

To obtain the answer set of P, we apple the immediate consequence operator to program P''. We start from a fuzzy interpretation $I_0 : \mathcal{B}_{P''} \to [0, 1] : a' \mapsto 0$. After one iteration of $\Pi_{P''}$ we obtain $I_1 = \Pi_{P''}(I_0)$ which is defined as follows: $I_1(a_{S_1}) = I_1(a_{S_3}) = 0$ and $I_1(a_{S_2}) = 0.3$. After one more iteration we obtain the least fixpoint $I_2 = \Pi_{P''}(I_1)$ where $I_2(a_{S_1}) = 1$, $I_2(a_{S_2}) = 0.3$ and $I_2(a_{S_3}) = 0$. This fixpoint then coincides with the unique answer set I of P: $I(a) = I(b) = I_2(a_{S_1}) = 1$, $I(c) = I(d) = I_2(a_{S_2}) = 0.3$ and $I(e) = I_2(a_{S_3}) = 0$.

6.2. Complexity of regular normal FASP programs

For normal programs in classical ASP, NP-membership follows straightforwardly from the fact that we can guess an answer set and verify that the guess is stable in polynomial time. In contrast, due to the infinite number of possible truth values, in FASP not every answer set can be guessed in polynomial time. To address this issue, by analyzing the geometrical structure of fuzzy equilibrium models, [23] shows that whenever there is an answer set I such that $I(l) \geq \lambda$ for a literal l, there always is an answer set J such that $J(l) \geq \lambda$ and such that for each literal l, $J(l)$ can be encoded using a polynomial number of bits. This means that we can verify whether $I(l) \geq \lambda$ for a regular normal FASP program by guessing an answer set in polynomial time and verifying that the guess is stable. As a result, several of the P-membership results for regular definite programs directly translate to NP-membership results for regular normal programs. The only exception is the class of regular normal FASP programs with polynomially bounded constants. Indeed, to check whether I is an answer set of such a program P it has to be verified that I is an answer set of P^I but P^I does not necessarily belong to the class of regular normal FASP programs with polynomially bounded constants. We also obtain the same results for the existence problem since it is the special case of the membership problem with $\lambda = 0$. For set-entailment we obtain coNP-membership if set-membership is in NP. Indeed, the complement of the set-entailment problem is "Given a program P, a literal l and a value λ_l, does there exist an answer set I of P such that $I(l) < \lambda_l$?" By similar results from [23], we know that if such an answer set exists, that there is always one that can be encoded using a polynomial number of bits.

Proposition 17.
1. *The set-membership and the existence problem for the class of regular normal FASP programs with only disjunction in the bodies of rules is in NP. Set-entailment is in coNP.*
2. *The set-membership and the existence problem for the class of regular normal FASP programs with cycle free dependency graphs is in NP. Set-entailment is in coNP.*

Moreover, as shown in Proposition 9, we can reduce the decision problems for normal ASP to regular normal FASP .

Corollary 5. *Existence and set-membership for regular normal FASP with polynomially bounded constants is NP-hard. Set-entailment is coNP-hard.*

7. Reduction to bilevel linear programming

In this section, we will show that we can translate strict disjunctive FASP programs into bilevel linear programs such that there is a one-to-one correspondence between particular solutions of the bilevel linear program and the answer sets of the FASP program. This implementation into bilevel linear programming can then be used as a basis to build solvers for FASP.

Bilevel linear programming problems are optimization problems in which the set of all variables is divided into two disjoint sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. An assignment to the variables will be denoted by a vector

$\mathbf{x} = (x_1, \ldots, x_n)$ for X and by a vector $\mathbf{y} = (y_1, \ldots, y_m)$ for Y. Intuitively, there are two agents, a leader who is responsible for the variables in X and a follower responsible for the variables in Y. Each vector $\mathbf{y}$ has to be chosen by the follower in function of the choice by the leader $\mathbf{x}$ as an optimal solution of the so-called *lower level problem* or the *follower's problem*. Knowing that the follower will react in that way, the leader wants to optimize his objective function in the so-called *upper level problem* or the *leader's problem*.

In a bilevel linear program all objective functions and constraints are linear. In particular, the type of bilevel linear programming problem in which we are interested is given by Bard [42]:

$$\begin{aligned}
&\mathbf{x}^* = \arg\min\nolimits_{\mathbf{x}} c_1\mathbf{x} + d_1\mathbf{y}^* \\
&\text{s.t. } A_1\mathbf{x} + B_1\mathbf{y}^* \leq b_1 \\
&\qquad\quad \mathbf{y}^* = \arg\min\nolimits_{\mathbf{y}} c_2\mathbf{x} + d_2\mathbf{y} \\
&\qquad\quad \text{s.t. } A_2\mathbf{x} + B_2\mathbf{y} \leq b_2
\end{aligned}$$

where $c_1, c_2 \in \mathbb{R}^n$, $d_1, d_2 \in \mathbb{R}^m$, $b_1 \in \mathbb{R}^p$, $b_2 \in \mathbb{R}^q$, $A_1 \in \mathbb{R}^{p\times n}$, $B_1 \in \mathbb{R}^{p\times m}$, $A_2 \in \mathbb{R}^{q\times n}$ and $B_2 \in \mathbb{R}^{q\times m}$.

Now consider a strict disjunctive FASP program P. Without loss of generality we may assume that this program contains no strong negation. We will translate P to a bilevel linear program Q such that the solutions of Q correspond to the answer sets of P. By definition, I is an answer set of P iff I is an answer set of P^I. Informally, a guess I needs to be made first and then it has to be checked whether this guess corresponds to an answer set of P. If $\mathcal{B}_P = \{a_1, \ldots, a_n\}$, then we will define the vector $\tilde{\mathbf{a}} = (\tilde{a}_1, \ldots, \tilde{a}_n)$ and the vector $\tilde{\mathbf{a}}' = (\tilde{a}'_1, \ldots, \tilde{a}'_n)$ where the vector $\tilde{\mathbf{a}}$ represents the truth values of the atoms in $\{a_1, \ldots, a_n\}$ and the vector $\tilde{\mathbf{a}}'$ intuitively represents the truth values of the guesses for the atoms. For each such guess I, represented by $\tilde{\mathbf{a}}'$, we want to check if it is a minimal fuzzy model of P^I. Note that P^I is a positive FASP program in which each rule is of the form

$$r : l_1 \oplus \ldots \oplus l_n \leftarrow x_1 \otimes \ldots \otimes x_m, \tag{1}$$

with l_i, x_j atoms and/or constants. As in the proof of Proposition 2, a fuzzy interpretation $J \in \mathcal{F}(\mathcal{L}_P)$ is a model of r iff

$$J(l_1) + \ldots + J(l_n) \geq J(x_1) + \ldots + J(x_m) - (m-1).$$

Thus for each rule $r \in P^I$ we have a constraint $x_1 + \ldots + x_m - m + 1 \leq l_1 + \ldots + l_n$.

Hence, for each guess $\tilde{\mathbf{a}}'$, i.e. an interpretation I, we check if there is a minimal model J of P^I such that $J(a_i) \leq I(a_i)$ by minimizing all elements in the vector $\tilde{\mathbf{a}}$ subject to the constraints arising from P^I. This is the follower's problem. Finally, the guess is chosen such that the differences between $J(a_i)$ and $I(a_i)$ are as small as possible. This can be done by minimizing the function $\sum_{i=1}^{n}(\tilde{a}'_i - \tilde{a}_i)$. If this sum is equal to 0, we have found an answer set. If this sum is not equal to 0, there cannot be an answer set.

More structured, we have the following proposition.

Proposition 18. *Given a strict disjunctive FASP program P not containing strong negation such that $\mathcal{B}_P = \{a_1, \ldots, a_n\}$. Define the following bilevel linear program Q_P.*

$$\begin{array}{l} \tilde{\mathbf{a}}'^{*} = \arg\min_{\tilde{\mathbf{a}}'} \sum_{i=1}^{n}(\tilde{a}'_i - \tilde{a}^{*}_i) \\ \textit{s.t. } 0 \leq \tilde{a}'_i \leq 1 \\ \qquad\qquad \tilde{\mathbf{a}}^{*} = \arg\min_{\tilde{\mathbf{a}}} \sum_{i=1}^{n} \tilde{a}_i \\ \qquad\qquad \textit{s.t. } \tilde{a}_i \leq \tilde{a}'_i, 0 \leq \tilde{a}_i \leq 1 \textit{ and} \\ \qquad\qquad\quad (\sum_{j=1}^{m} x_j) - m + 1 \leq \sum_{i=1}^{n} l_i \textit{ for each rule } (1) \\ \qquad\qquad\quad \textit{in the reduct of } P \textit{ w.r.t. } \tilde{\mathbf{a}}' \end{array}$$

Then

1. *If Q_P has a solution $\tilde{\mathbf{a}}^{*} = (\tilde{a}_1, \ldots, \tilde{a}_n)$, $\tilde{\mathbf{a}}'^{*} = (\tilde{a}'_1, \ldots, \tilde{a}'_n)$ such that the objective function of the upper level problem is evaluated to 0, then $I : \mathcal{B}_P \to [0,1] : a_i \mapsto \tilde{a}_i$ is an answer set of P.*
2. *If I is an answer set of P, then $\tilde{\mathbf{a}}^{*} = (\tilde{a}_1, \ldots, \tilde{a}_n)$, $\tilde{\mathbf{a}}'^{*} = (\tilde{a}_1, \ldots, \tilde{a}_n)$ where $I(a_i) = \tilde{a}_i$ for each $i \in \{1, \ldots, n\}$ is a solution of Q_P such that the upper level problem is evaluated to 0.*

Example 10. *Reconsider the strict normal FASP program P from Example 2. The corresponding bilevel linear program is*

$$\begin{array}{l} \arg\min_{\tilde{a}',\tilde{b}'}[(\tilde{a}' - \tilde{a}) + (\tilde{b}' - \tilde{b})] \\ \textit{s.t. } 0 \leq \tilde{a}', \tilde{b}' \leq 1 \\ \qquad\qquad \arg\min_{\tilde{a},\tilde{b}}[\tilde{a} + \tilde{b}] \\ \qquad\qquad \textit{s.t. } 0 \leq \tilde{a}, \tilde{b} \leq 1, \tilde{a} \leq \tilde{a}', \tilde{b} \leq \tilde{b}' \\ \qquad\qquad\quad 1 - \tilde{a}' \leq \tilde{b}, 1 - \tilde{b}' \leq \tilde{a} \end{array}$$

The only assignments to the variables such that the objective function of the leader is equal to 0 are the ones with $\tilde{a}' = \tilde{a}$, $\tilde{b}' = \tilde{b}$ and $\tilde{a}' = 1 - \tilde{b}'$.

Remark 5. *A similar construction can be used if ASP is combined with other fuzzy logics, e.g. product logic, but the resulting bilevel program will not necessarily be linear.*

8. Conclusions

In this paper, we presented an overview about the computational complexity of FASP under Łukasiewicz semantics. In particular, when restricting to disjunctions in the head of rules and conjunctions in the bodies of rules, i.e. strict disjunctive FASP programs, NP-completeness was shown, which stands in contrast with the fact that disjunctive ASP is Σ_2^P-complete. This result even holds when restricting to strict disjunctive FASP without strong negation and with exactly one literal in the head of each rule. Hence, allowing disjunctions in the head has no influence on the computational complexity when the only connective in the body is conjunction. Given that we have not been able to show NP-membership for regular normal FASP programs in which both conjunction and disjunction are allowed in the bodies of rules, it is tempting to speculate that, unlike in the classical case, allowing disjunction in the body affects the computational complexity, whereas allowing it in the head does not. For regular simple FASP programs, we showed a correspondence to an open problem which indicates that setting the complexity may not be easy. However, we showed membership in P for several interesting subclasses. Finally, we have proposed an implementation of strict disjunctive FASP using bilevel linear programming which opens the door to practical applications.

Some open problems remain:

- Does there exist a polynomial time algorithm to compute the answer set of a regular simple FASP program?
- Is existence NP-hard for strict disjunctive FASP if constraints and strong negation are not allowed?
- Is regular normal FASP in NP?

Appendix A. Proofs

Proof of Lemma 1

Lemma. *Let P be a regular FASP program. There exists a regular FASP program P' without strong negation such that a fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$*

is an answer set of P iff there exists an answer set $I' \in \mathcal{F}(\mathcal{L}_{P'})$ of P' such that for each atom $a \in \mathcal{B}_P$ we have $I(a) = I'(a)$ and $I(\neg a) = I'(a')$ for some $a' \in \mathcal{B}_{P'}$.

Proof. For each atom a in P, introduce a fresh atom a'. The program P' is then obtained by replacing all negated atoms $\neg a$ in P by their corresponding atom a' and for each couple of atoms a, a' adding the constraint $\overline{0} \leftarrow a \otimes a'$. The program P' then has the required properties since $[\overline{0} \leftarrow a \otimes a']_I = 1$ iff $I(a) + I(a') \leq 1$ for each couple of atoms a, a'. □

Proof of Lemma 2

Lemma. *Let P be a regular FASP program such that $P = P' \cup C$ where C is a set of constraints in P and $I \in \mathcal{F}(\mathcal{L}_P)$. It holds that I is an answer set of P iff I is an answer set of P' and a fuzzy model of C.*

Proof. ($\Rightarrow$) Suppose I is an answer set of P. By assumption, I is a fuzzy model of C. It remains to be shown that I is an answer set of P'. I is a fuzzy model of $(P')^I$ since it is a fuzzy model of P^I. Now suppose there exists a fuzzy model J of $(P')^I$ such that $J \leq I$. We show that $J = I$, from which it then follows that I is a minimal fuzzy model of $(P')^I$ and hence an answer set of P'. Note that J (with $J(a) = 0$ for $a \notin \mathcal{L}_{P'}$) is a fuzzy model of P^I. Indeed, let r be an arbitrary rule in $P^I = (P' \cup C)^I$. If $r^I \in (P')^I$, then it is modelled by J by assumption. If $r^I : \overline{c} \leftarrow \alpha^I \in C^I$, then $[\alpha^I]_J \leq [\alpha^I]_I \leq c$, hence $J \models r^I$. Thus $J \leq I$ is a fuzzy model of P^I. Together with the fact that I is a minimal fuzzy model of P^I, this implies that $J = I$.

($\Leftarrow$) Suppose that I is an answer set of P' and a fuzzy model of C. Then I is a fuzzy model of $P^I = (P' \cup C)^I$ as well. Now suppose there exists a fuzzy model J of P^I such that $J \leq I$, then it follows that J is a model of $(P')^I \subseteq P^I$. Hence, since I is an answer set of P' and thus by definition a minimal fuzzy model of $(P')^I$, it follows that $J = I$

□

Proof of Lemma 3

Lemma. *Let $P = P_1 \cup \{r\}$ be a regular FASP program where*

$$r : \beta \leftarrow f(l_1, \ldots, l_n)$$

with l_i (negation-as-failure) literals and/or constants, β an arbitrary head and $f(l_1, \ldots, l_n)$ denotes either $\otimes_{i=1}^{n} l_i$, $\oplus_{i=1}^{n} l_i$, $\max(l_1, \ldots, l_n)$ or $\min(l_1, \ldots, l_n)$.

For a fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$, it holds that I is an answer set of P iff there exists a fuzzy interpretation $I' \in \mathcal{F}(\mathcal{L}_{P'})$ such that $I'_{|\mathcal{L}_P} = I$ and I' is an answer set of P' where $P' = P_1 \cup P_2$ and P_2 is the program consisting of the rules

$$\begin{array}{lll} b_1 & \leftarrow & f(l_1, l_2) \\ b_2 & \leftarrow & f(b_1, l_3) \\ & \vdots & \\ b_{n-2} & \leftarrow & f(b_{n-3}, l_{n-1}) \\ \beta & \leftarrow & f(b_{n-2}, l_n) \end{array}$$

with $b_1, \ldots, b_{n-2}$ atoms which are not used in P.

Proof. ($\Rightarrow$) Suppose I is an answer set of P. We expand I to a fuzzy interpretation I' on $\mathcal{L}_{P'}$ as follows. Define $I'(b_1) = [f(l_1, l_2)]_{I'}$ and $I'(b_i) = [f(b_{i-1}, l_{i+1})]_{I'}$ for $i \neq 1$. It is easy to see that I' is a fuzzy model of $(P')^{I'}$. Next, we show that I' is a minimal fuzzy model of $(P')^{I'}$. Suppose that $J' \leq I'$ is a fuzzy model of $(P')^{I'}$. One can show that $J = J'_{|\mathcal{L}_P}$ is a fuzzy model of P^I, hence it follows that $J = I$. We prove by induction on $i = 1, \ldots, n-2$ that $J' = I'$:

$$\begin{aligned} I'(b_1) &= [f(l_1, l_2)]_{I'} && \text{(definition } I'\text{)} \\ &= [f(l_1^{I'}, l_2^{I'})]_{I'} && \text{(definition reduct)} \\ &= [f(l_1^{I'}, l_2^{I'})]_{J'} && (J'_{|\mathcal{L}_P} = J = I = I'_{|\mathcal{L}_P}) \\ &\leq J'(b_1) && (J' \text{ fuzzy model of } (P')^{I'}) \\ &\leq I'(b_1) && (J' \leq I') \end{aligned}$$

Suppose $I'(b_{i-1}) = J'(b_{i-1})$.

$$\begin{aligned} I'(b_i) &= [f(b_{i-1}, l_{i+1})]_{I'} && \text{(definition } I'\text{)} \\ &= [f(b_{i-1}, l_{i+1}^{I'})]_{I'} && \text{(definition reduct)} \\ &= [f(b_{i-1}, l_{i+1}^{I'})]_{J'} && (\text{induction and } J'_{|\mathcal{L}_P} = I'_{|\mathcal{L}_P}) \\ &\leq J'(b_i) && (J' \text{ fuzzy model of } (P')^{I'}) \\ &\leq I'(b_i) && (J' \leq I') \end{aligned}$$

($\Leftarrow$) Suppose there is a fuzzy interpretation $I' \in \mathcal{L}_{P'}$ such that I' is an answer set of P' and $I'_{|\mathcal{L}_P} = I$. We show that I is an answer set of P. First note that since I' is a minimal fuzzy model of $(P')^{I'}$, it must hold that $I'(b_1) = [f(l_1, l_2)]_{I'}$ and $I'(b_i) = [f(b_{i-1}, l_{i+1})]_{I'}$ for $i \neq 1$. A straightforward proof then shows that I is a fuzzy model of P^I. Now suppose that there exists a fuzzy model $J \leq I$ of P^I. We show that there exists $J' \in \mathcal{L}_{P'}$ which is a fuzzy model of $(P')^{I'}$ such that $J' \leq I'$ and $J'_{|\mathcal{L}_P} = J$. Since I' is a minimal fuzzy model of $(P')^{I'}$, it then follows that $J' = I'$ and hence $J = I$. Define J' as follows: for $l \in \mathcal{L}_P$ define $J'(l) = J(l)$ and $J'(b_1) = [f(l_1^{I'}, l_2^{I'})]_{J'}$ and $J'(b_i) = [f(b_{i-1}, l_{i+1}^{I'})]_{J'}$ for $i \neq 1$. We prove by induction on $i = 1, \ldots, n-2$ that $J' \leq I'$:

$$\begin{aligned}
J'(b_1) &= [f(l_1^{I'}, l_2^{I'})]_{J'} && \text{(definition } J'\text{)}\\
&= [f(l_1^{I}, l_2^{I})]_{J} && (I = I'_{|\mathcal{L}_P}, J = J'_{|\mathcal{L}_P})\\
&\leq [f(l_1^{I}, l_2^{I})]_{I} && (J \leq I)\\
&= [f(l_1^{I'}, l_2^{I'})]_{I'} && (I = I'_{|\mathcal{L}_P})\\
&= I'(b_1)
\end{aligned}$$

If $J'(b_{i-1}) \leq I'(b_{i-1})$, then

$$\begin{aligned}
J'(b_i) &= [f(b_{i-1}, l_{i+1}^{I'})]_{J'} && \text{(definition } J'\text{)}\\
&= \mathtt{f}(J'(b_{i-1}), J'(l_{i+1}^{I'}))\\
&= \mathtt{f}(J'(b_{i-1}), I'(l_{i+1}^{I'})) && (I'_{|\mathcal{L}_P} = J'_{|\mathcal{L}_P})\\
&\leq \mathtt{f}(I'(b_{i-1}), I'(l_{i+1}^{I'})) && \text{(induction and } \mathtt{f} \text{ increasing)}\\
&= I'(b_i)
\end{aligned}$$

It is easy to show that J' is a fuzzy model of $(P')^{I'}$.

$\square$

Proof of Lemma 4

Lemma. *Let $P = P_1 \cup \{r\}$ be a regular FASP program where*

$$r : \beta \leftarrow f(\alpha_1, \ldots, \alpha_n)$$

with α_i formulas built from (negation-as-failure) literals and/or constants, $\otimes$, $\oplus$, max, min *and β an arbitrary head and $f(\alpha_1,\ldots,\alpha_n)$ denotes either $\otimes_{i=1}^n \alpha_i$, $\oplus_{i=1}^n \alpha_i$,* $\max(\alpha_1,\ldots,\alpha_n)$ *or* $\min(\alpha_1,\ldots,\alpha_n)$.

For a fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$, it holds that I is an answer set of P iff there exists a fuzzy interpretation $I' \in \mathcal{F}(\mathcal{L}_{P'})$ such that $I'_{|\mathcal{L}_P} = I$ and I' is an answer set of P' where $P' = P_1 \cup P_2$ and P_2 is the program consisting of the rules

$$\begin{array}{lcl} a_1 & \leftarrow & \alpha_1 \\ a_2 & \leftarrow & \alpha_2 \\ & \vdots & \\ a_n & \leftarrow & \alpha_n \\ \beta & \leftarrow & f(a_1,\ldots,a_n) \end{array}$$

with $a_1,\ldots,a_n$ atoms which are not used in P.

Proof. ($\Rightarrow$) Suppose that I is an answer set of P. We expand I to a fuzzy interpretation $I' \in \mathcal{F}(\mathcal{L}_{P'})$ as follows: $I'(a_i) = I'(\alpha_i) = I(\alpha_i)$ for $i \in \{1,\ldots,n\}$. It is easy to show that I' is a fuzzy model of $(P')^{I'}$. Next, we show that I' is a minimal fuzzy model of $(P')^{I'}$. Suppose there exists a fuzzy model $J' \leq I'$ of $(P')^{I'}$. We show that $J' = I'$. First remark that $J = J'_{|\mathcal{L}_P}$ is a fuzzy model of P^I. Since I is a minimal fuzzy model of P^I, it follows that $J'_{|\mathcal{L}_P} = J = I = I'_{|\mathcal{L}_P}$. It remains to be shown that $J'(a_i) = I'(a_i)$ for $i \in \{1,\ldots,n\}$. But this follows easily:

$$\begin{aligned} J'(a_i) &\leq I'(a_i) && (J' \leq I') \\ &= I'(\alpha_i) \\ &= I'(\alpha_i^{I'}) \\ &= J'(\alpha_i^{I'}) && (J'_{|\mathcal{L}_P} = I'_{|\mathcal{L}_P}) \\ &\leq J'(a_i) && (J' \text{ fuzzy model of } (P')^{I'}) \end{aligned}$$

($\Leftarrow$) Suppose that I' is a minimal fuzzy model of $(P')^{I'}$. We show that $I = I'_{|\mathcal{L}_P}$ is a minimal fuzzy model of P^I. Remark that, since I' is a minimal fuzzy model of $(P')^{I'}$, it must hold that $I'(a_i) = I'(\alpha_i) = I(\alpha_i)$. It follows easily that I is a fuzzy model of P^I. Now suppose there exists a fuzzy model $J \leq I$ of P^I. We expand J to a fuzzy interpretation $J' \in \mathcal{L}_{P'}$ as follows: $J'(a_i) = J(\alpha_i^I)$. It can be shown that J' is a fuzzy model of $(P')^{I'}$. Moreover, for each $i \in \{1, \ldots, n\}$, we have that

$$\begin{aligned} J'(a_i) &= J(\alpha_i^I) \\ &\leq I(\alpha_i^I) && (J \leq I) \\ &= I(\alpha_i) \\ &= I'(a_i) \end{aligned}$$

Hence $J' \leq I'$ and since I' is a minimal fuzzy model of $(P')^{I'}$, it follows that $J' = I'$ and thus $J = I$.

□

Proof of Proposition 1

Proposition. *Let P be a regular FASP program. P can be reduced (in time polynomial in the size of the program) to a regular FASP program P' such that $\mathcal{L}_P \subseteq \mathcal{L}_{P'}$ and each rule in P' has at most two arguments in the body and I is an answer set of P iff there exists a fuzzy interpretation $I' \in \mathcal{F}(\mathcal{L}_{P'})$ such that $I'_{|\mathcal{L}_P} = I$ and I' is an answer set of P'.*

Proof. Suppose there exists a rule $r \in P$ with more than 2 arguments. We show by induction on the number of connectives n, written in prefix notation, that this rule can be rewritten as a set of rules with in the bodies of rules maximum two arguments and one connective such that the answer sets remain the same.

If $n = 1$, then r is of the form

$$r : \beta \leftarrow f(l_1, \ldots, l_m).$$

By Lemma 3 the assertion holds. Now suppose the assertion holds for $n < k$. We prove that it also holds for $n = k$. Rule r is now of the form

$$r : \beta \leftarrow f(\alpha_1, \ldots, \alpha_n),$$

where the number of connectives in α_i is strictly smaller than k. By Lemma 4, the assertion follows. □

Proof of Proposition 2

Proposition. *Set-membership for strict disjunctive FASP is in NP.*

Proof. From the analysis of the geometrical structure underlying fuzzy equilibrium models which is a proper generalization of regular FASP [23], it follows that a FASP program P has an answer set I such that $I(l) \geq \lambda_l$ for some $l \in \mathcal{L}_\mathcal{P}$ and $\lambda_I \in [0,1] \cap \mathbb{Q}$ iff there is such an answer set that can be encoded using a polynomial number of bits.

Given a strict disjunctive program P and an answer set I, we show that we can check in polynomial time that I is an answer set of P. Note that checking if $I(l) \geq \lambda_l$ for a literal l can be done in constant time. By definition, we need to check that I is a minimal fuzzy model of P^I and that for each $l \in \mathcal{L}_P$ we have $I(l) + I(\neg l) \leq 1$. The latter is straightforward. To check whether I is a minimal fuzzy model of P^I, we can use linear programming. Indeed a rule $r : a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m$ from P^I is satisfied iff

$$\begin{aligned}
&[b_1 \otimes \ldots \otimes b_m \rightarrow a_1 \oplus \ldots \oplus a_n]_I = 1 \\
&\Leftrightarrow [(\sim b_1) \oplus \ldots \oplus (\sim b_m) \oplus a_1 \oplus \ldots \oplus a_n]_I = 1 \\
&\Leftrightarrow I(\sim b_1) + \ldots + I(\sim b_m) + I(a_1) + \ldots + I(a_n) \geq 1 \\
&\Leftrightarrow 1 - I(b_1) + \ldots + 1 - I(b_m) + I(a_1) + \ldots + I(a_n) \geq 1
\end{aligned}$$

Hence, to check whether I is a minimal fuzzy model of P^I we use the following linear program M. The function to be minimized is the sum $\sum_{a \in \mathcal{L}_{P^I}} a'$ where for each literal $a \in \mathcal{L}_\mathcal{P}$ we introduce a variable a' and the constraints in M are the following. For each literal $a \in \mathcal{L}_{P^I}$ we have $0 \leq a' \leq 1$ and for each rule

$$r : a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m$$

in P^I we have

$$1 \leq 1 - b'_1 + \ldots + 1 - b'_m + a'_1 + \ldots + a'_n$$

or equivalently

$$1 - m \leq -b'_1 - \ldots - b'_m + a'_1 + \ldots + a'_n.$$

If $a' = I(a)$ for each literal a is a solution of M, then I is a minimal fuzzy model of P^I. Indeed, since $I(a) = a'$ fulfills the constraints of M, it is a fuzzy model of P^I. Now suppose there exists a fuzzy model J such that $J < I$. Since it is a fuzzy model of P^I, the assignments $a'' = J(a)$ for each literal a satisfy the constraints of M but $\sum_{a \in \mathcal{L}_{P^I}} a'' < \sum_{a \in \mathcal{L}_{P^I}} a'$, a contradiction. Hence I is a minimal fuzzy model of P^I. □

Proof of Proposition 3

Proposition. *Set-membership for strict normal FASP is NP-hard.*

Proof. Consider an arbitrary instance of the 3SAT problem

$$\alpha = (a_{11} \vee a_{12} \vee a_{13}) \wedge (a_{21} \vee a_{22} \vee a_{23}) \wedge \ldots \wedge (a_{n1} \vee a_{n2} \vee a_{n3})$$

We translate each clause $a_{i1} \vee a_{i2} \vee a_{i3}$ to the rule

$$\overline{0} \leftarrow \neg a_{i1} \otimes \neg a_{i2} \otimes \neg a_{i3} \tag{A.1}$$

and for each literal x in α we add the rules

$$\neg x \leftarrow \text{not}\, x \tag{A.2}$$
$$x \leftarrow \text{not}(\neg x) \tag{A.3}$$
$$x' \leftarrow x \tag{A.4}$$
$$x' \leftarrow \neg x \tag{A.5}$$
$$\overline{0} \leftarrow \text{not}(x') \tag{A.6}$$

where x' is a fresh atom not used in α. We denote the resulting strict normal FASP program by P.

1. First suppose that I is an answer set of P. By Lemma 2 we know that I is an answer set of P_1 and a fuzzy model of C where P_1 is the set of all rules in P of the form (A.2)-(A.5) and C is the set of all constraints of the form (A.1) and (A.6).
   Since I is a minimal fuzzy model of $(P_1)^I$ we know that for each literal x it holds that $I(x) = 1 - I(\neg x)$ by rules (A.2) and (A.3) and $I(x') = \max(I(x), I(\neg x))$ by rules (A.4) and (A.5). Since I must be a fuzzy model of the constraints in C, it follows that $1 - I(x') = 0$ by rule (A.6). If $I(x') = I(x)$, then $I(x) = 1$ and $I(\neg x) = 0$. Otherwise, if $I(x') = I(\neg x)$, then $I(\neg x) = 1$ and $I(x) = 0$. Hence, I is a consistent Boolean interpretation.
   Let us define the propositional interpretation G as follows. For each literal x in α we have $G(x) =$ "true" if $I(x) = 1$ and $G(x) =$ "false" if $I(x) = 0$. We check that this assignment evaluates α to "true". This follows easily by (A.1) and the following equations:

$$\begin{aligned}
&[\neg a_{i1} \otimes \neg a_{i2} \otimes \neg a_{i3} \to \overline{0}]_I = 1\\
&\Leftrightarrow [\overline{0} \oplus \sim (\neg a_{i1} \otimes \neg a_{i2} \otimes \neg a_{i3})]_I = 1\\
&\Leftrightarrow [\overline{0} \oplus \sim (\neg a_{i1}) \oplus \sim (\neg a_{i2}) \oplus \sim (\neg a_{i3})]_I = 1\\
&\Leftrightarrow 0 + 1 - I(\neg a_{i1}) + 1 - I(\neg a_{i2}) + 1 - I(\neg a_{i3}) \geq 1
\end{aligned}$$

Since for I it holds that $I(x) = 1 - I(\neg x)$ for each literal x, we obtain that

$$\begin{aligned}&[\neg a_{i1} \otimes \neg a_{i2} \otimes \neg a_{i3} \rightarrow \overline{0}]_I = 1\\ \Leftrightarrow\ &I(a_{i1}) + I(a_{i2}) + I(a_{i3}) \geq 1\end{aligned}$$

Because I is a Boolean interpretation, it must hold that $I(a_{ij}) = 1$ for at least one literal a_{ij} in each clause. Hence, G is an assignment that evaluates each clause $a_{i1} \vee a_{i2} \vee a_{i3}$, and thus the whole expression α, to "true".

2. Consider a propositional interpretation G such that each clause $a_{i1} \vee a_{i2} \vee a_{i3}$ evaluates to "true". We define a fuzzy interpretation in $\mathcal{F}(\mathcal{L}_P)$ by $I(x) = 1$ if $G(x) =$ "true", $I(x) = 0$ if $G(x) =$ "false", $I(x') = \max(I(x), I(\neg x))$. Note that $I(\neg x) = 1 - I(x)$ since G is a propositional interpretation . We show that I is an answer set of P, or by Lemma 2 that it is a minimal fuzzy model of $(P_1)^I$ and a fuzzy model of C. It is clear that I is a fuzzy model of $(P_1)^I$. Now suppose there exists a fuzzy model J of $(P_1)^I$ such that $J < I$. Since I is such that $I(\neg x) + I(x) = 1$, by rules (A.2) and (A.3) in P_1 it follows that

$$J(\neg x) \geq [\text{not}\, x]_I = 1 - I(x) = I(\neg x) \geq J(\neg x)$$

and

$$J(x) \geq [\text{not}(\neg x)]_I = 1 - I(\neg x) = I(x) \geq J(x).$$

Hence we have for each literal x that $J(x) = I(x)$ and $J(\neg x) = I(\neg x)$. Since $J < I$, there must exist a literal x such that $J(x') < I(x')$ which implies by rules (A.4) and (A.5) in P_1 that

$$I(x') > J(x') \geq J(x) = I(x) \text{ and } I(x') > J(x') \geq J(\neg x) = I(\neg x).$$

This is impossible since either $I(x) = 1$ or $I(\neg x) = 1$ and then $I(x') > 1$.

It remains to be shown that I is a fuzzy model of C. Since $I(x') = \max(I(x), I(\neg x)) = 1$ we have that I models the rule $\overline{0} \leftarrow \text{not}(x')$ for each literal x. As before, we obtain

$$\begin{aligned}&[\overline{0} \leftarrow \neg(a_{i1}) \otimes \neg(a_{i2}) \otimes \neg(a_{i3})]_I = 1\\ \Leftrightarrow\quad &I(a_{i1}) + I(a_{i2}) + I(a_{i3}) \geq 1\end{aligned}$$

Since each clause $a_{i1} \vee a_{i2} \vee a_{i3}$ is satisfied by G, we know that for least one a_{ij} it must hold that $I(a_{ij}) = 1$. Hence $I(a_{i1}) + I(a_{i2}) + I(a_{i3}) \geq 1$.

□

Proof of Proposition 5

Proposition. 1. *Set-entailment for strict normal FASP is coNP-complete.*

2. *Set-entailment for strict disjunctive FASP is coNP-complete.*

Proof. To show coNP-membership for set-entailment in strict normal (disjunctive) FASP, we show that the complementary decision problem, i.e. "Given a strict normal (disjunctive) FASP program P, a literal $l \in \mathcal{L}_P$ and a value $\lambda_l \in [0,1] \cap \mathbb{Q}$; is there an answer set I of P such that $I(l) < \lambda_l$?" is in NP by a straightforward adaption of the proof of Proposition 2.

To show coNP-hardness, we reduce the NP-hard problem "existence" to the complement of the set-entailment problem. Consider a strict normal (disjunctive) FASP program P. Define $P' = P \cup \{a \leftarrow a\}$ with a a fresh atom. We show that P has an answer set iff it is not the case that all answer sets I' of P' are such that $I'(a) \geq 0.5$. First suppose that P has an answer set I. Then there exists an answer set I' of P' with $I'(a) < 0.5$. Indeed, define $I'(a) = 0$ and $I'(x) = I(x)$ otherwise. Next, suppose that there exists an answer set I' of P' such that $I'(a) < 0.5$. Then $I = I'_{|\mathcal{L}_P}$ is an answer set of P.

□

Proof of Lemma 5

Lemma. *Consider a regular FASP program $P = P_1 \cup C$ where P_1 is a regular FASP program and C is a set of constraints of the form $\overline{0} \leftarrow \alpha$. Let $P' = P_1 \cup C' \cup \{z \leftarrow \text{not}\, y\}$ where z and y are fresh atoms and $C' = \{y \leftarrow \alpha \mid (\overline{0} \leftarrow \alpha) \in C\}$. A fuzzy interpretation $I \in \mathcal{F}(\mathcal{L}_P)$ is an answer set of P iff there exists an answer set $I' \in \mathcal{F}(\mathcal{L}_{P'})$ of P' such that $I'_{|\mathcal{L}_P} = I$ and $I'(z) \geq 1$.*

Proof. ($\Rightarrow$) Suppose that $I \in \mathcal{F}(\mathcal{L}_P)$ is an answer set of P. Define $I' \in \mathcal{F}(\mathcal{L}_{P'})$ as $I'(a) = I(a)$ if $a \in \mathcal{L}_P$, $I'(z) = 1$ and $I'(y) = 0$. We show that I' is an answer set of P'.

First, we prove that I' is a fuzzy model of P' and thus of $(P')^{I'}$. Clearly, I' is a fuzzy model of P_1 and it models the rule $z \leftarrow \text{not}\, y$. If $y \leftarrow \alpha$ is a rule in C', then by assumption we have that $I = I'_{|\mathcal{L}_P}$ models the rule $\overline{0} \leftarrow \alpha$. Thus $[\overline{0} \leftarrow \alpha]_{I'} = 1$ and $[\alpha]_{I'} = 0 = I'(y)$. Hence I' models $y \leftarrow \alpha$.

Next, we show that I' is a minimal fuzzy model of $(P')^{I'}$. Suppose there exists a fuzzy model $J' \in \mathcal{F}(\mathcal{L}_{P'})$ of $(P')^{I'}$ such that $J' \leq I'$. We show that $J = J'_{|\mathcal{L}_P}$ is a fuzzy model of P^I. Clearly, J is a fuzzy model of $(P_1)^I$. Since $J' \leq I'$ we have that $J'(y) \leq I'(y) = 0$, thus given a rule $r : \overline{0} \leftarrow \alpha$ in C we have that for the corresponding rule $y \leftarrow \alpha$ in C' it holds that $0 = J'(y) \geq [\alpha^I]_J$, with α^I the reduct of the expression α w.r.t. I. Hence $[r^I]_J = 1$. Because I is a minimal fuzzy model of P^I, it follows that $I = J$. As mentioned before, we have $J'(y) = I'(y)$ and since $[z \leftarrow [\text{not}\, y]_{I'}]_{J'} = 1$, we also have $J'(z) \geq 1 - I'(y) = I'(z) \geq J'(z)$. Hence $I' = J'$, which shows that I' is a minimal fuzzy model of $(P')^{I'}$.

($\Leftarrow$) Suppose that $I' \in \mathcal{F}(\mathcal{L}_{P'})$ is an answer set of P' such that $I'(z) = 1$. We show that $I = I'_{|\mathcal{L}_P}$ is an answer set of P. By Lemma 2 it is sufficient to show that I is an answer set of P_1 and a fuzzy model of C.
First, we show that I is a fuzzy model of C. Since I' is a minimal fuzzy model of $(P')^{I'}$, it must hold that $I'(z) = 1 - I'(y)$ and thus that $I'(y) = 0$. Given a rule $r : \overline{0} \leftarrow \alpha$ in C we have that for the corresponding rule $y \leftarrow \alpha$ in C' it holds that $0 = I'(y) \geq [\alpha]_{I'}$, and thus $[r]_I = 1$.
Next, note that I is a fuzzy model of $(P_1)^I$ since I' is a fuzzy model of $(P_1)^{I'}$. Now suppose there exists a fuzzy model $J \in \mathcal{F}(\mathcal{L}_{P_1})$ of $(P_1)^I$ such that $J \leq I$. Define $J' \in \mathcal{F}(\mathcal{L}_{P'})$ as follows: $J'(a) = J(a)$ if $a \in \mathcal{L}_P$, $J'(y) = 0$ and $J'(z) = 1$. We show that J' is a fuzzy model of $(P')^{I'}$. By assumption, J' is a fuzzy model of $(P_1)^{I'}$. For the rule $r : z \leftarrow \text{not}\, y$ in P' we have $J'(z) = 1 = I'(z) \geq [\text{not}\, y]_{I'}$, hence J' models $r^{I'}$. Finally, given a rule $r : y \leftarrow \alpha$ in C' we have for the corresponding rule $\overline{0} \leftarrow \alpha$ in C that $J'(y) = 0 \geq [\alpha^{I'}]_{J'}$. Hence J' models $r^{I'}$. Since $J' \leq I'$ and I' is a minimal fuzzy model of $(P')^{I'}$ it follows that $J' = I'$ and thus $J = I$.

□

Proof of Proposition 6

Proposition. *Set-membership for strict normal FASP is NP-hard, even if constraints and strong negation are not allowed.*

Proof. Consider an arbitrary instance of the 3SAT problem

$$\alpha = (a_{11} \vee a_{12} \vee a_{13}) \wedge (a_{21} \vee a_{22} \vee a_{23}) \wedge \ldots \wedge (a_{n1} \vee a_{n2} \vee a_{n3})$$

As shown in the proof of Proposition 3, α is satisfied by an assigment G iff the propositional interpretation I, with $I(x) = 1$ if $G(x) =$ "true" and $I(x) = 0$ if $G(x) =$ "false" is an answer set of P with P the program obtained in the proof of Proposition 3.

By Lemma 1 it follows that P can be rewritten to a strict normal FASP program P' without strong negation and in which the head contains exactly one atom or the constant $\overline{0}$ such that there is a one-on-one correspondence between the answer sets. By Lemma 5, it follows that we can define a strict normal FASP program P'' without constraints and without strong negation such that the answer sets of P' correspond to the answer sets of P'' for which a certain atom has at least truth value 1. □

Proof of Corollary 2

Corollary. 1. *Set-membership for strict normal FASP is NP-complete, even if constraints and strong negation are not allowed.*
2. *Set-membership for strict disjunctive FASP is NP-complete, even if constraints and strong negation are not allowed.*

Proof. Follows by the reduction in the proof of Proposition 6 and by Proposition 2. □

Proof of Proposition 7

Proposition. *A strict normal FASP program without constraints and without strong negation always has an answer set.*

Proof. From Theorem 3.1 in [37] it follows that such a strict normal FASP program has at least one answer set. □

Proof of Proposition 8

Proposition. *The unique answer set of a regular simple FASP program with only conjunction and maximum in the body of the rules can be found in polynomial time.*

Proof. Consider a regular simple FASP program P with only rules of the form $a \leftarrow b \otimes c$ and $d \leftarrow \max(e, f)$. The answer set P can be found by solving the following linear program L_P. The function to be minimized is

$f(a'_1, \ldots a'_n) = \sum_{i=1}^{n} a'_i$ with $\mathcal{B}_P = \{a_1, \ldots, a_n\}$ and a'_i is the corresponding variable for a_i and for each rule $a \leftarrow b \otimes c$ we add the constraints

$$\begin{aligned} a' &\geq b' + c' - 1 \\ 1 &\geq a', b', c' \geq 0 \end{aligned}$$

and for each rule $d \leftarrow \max(e, f)$, we add the constraints

$$\begin{aligned} d' &\geq e' \\ d' &\geq f' \\ 1 &\geq d', e', f' \geq 0. \end{aligned}$$

Suppose that $I : \mathcal{B}_P \rightarrow [0,1]$ is the answer set of P, i.e. I is the unique minimal fuzzy model of P. We show that if L_P has a solution $J' : \{a'_1, \ldots, a'_n\} \rightarrow \mathbb{R}$, that $J : \mathcal{B}_P \rightarrow [0,1] : a_i \mapsto J'(a'_i)$ is a minimal fuzzy model of P. Since P has a unique minimal fuzzy model, we then obtain $J = I$ and J' is the unique solution of L_P. Clearly, since J' satisfies the constraints in L_P we obtain that J is a fuzzy model of P. Suppose J is not a minimal fuzzy model of P, i.e. there exists a fuzzy model $M : \mathcal{B}_P \rightarrow [0,1]$ of P such that $M < J$, then $M' : \{a'_1, \ldots, a'_n\} \rightarrow \mathbb{R} : a'_i \mapsto M(a_i)$ satisfies the constraints of L_P and it holds that $\sum_{i=1}^{n} M'(a'_i) < \sum_{i=1}^{n} J'(a'_i)$, a contradiction.

□

Proof of Corollary 3

Corollary. *The complexity of the decision problems for strict simple and strict definite FASP is polynomial.*

Proof. Follows from Remark 2 and Proposition 8. □

Proof of Proposition 9

Proposition. *Let P be a disjunctive ASP program and let $I \in \mathcal{P}(\mathcal{L}_P)$. Define the regular FASP program P' as follows:*

$$\begin{gathered} P' = \{a_1 \oplus \ldots \oplus a_n \leftarrow b_1 \otimes \ldots \otimes b_m \otimes \text{not}\, c_1 \otimes \ldots \otimes \text{not}\, c_k \mid \\ (a_1 \vee \ldots \vee a_n \leftarrow b_1 \wedge \ldots \wedge b_m \wedge \text{not}\, c_1 \wedge \ldots \wedge \text{not}\, c_k) \in P\} \\ \cup \{a \leftarrow a \oplus a \mid a \in \mathcal{L}_P\}. \end{gathered}$$

Then I is an answer set of P iff I is an answer set of P'.

Proof. First note that $I \in \mathcal{F}(\mathcal{L}_P)$ models a rule of the form $a \leftarrow a \oplus a$ iff $\min(2I(a), 1) \leq I(a)$. This is only possible if $I(a) = 0$ or $I(a) = 1$. The proposition then follows from the fact that the Łukasiewicz connectives restricted to values in $\{0, 1\}$ agree with the corresponding classical connectives, and the semantics of ASP and FASP coincide in such a case. □

Proof of Proposition 10

Proposition.
1. *Set-membership and existence for regular FASP are Σ_2^P-complete. Set-entailment is Π_2^P-complete.*
2. *Set-membership and existence for regular normal FASP are NP-hard and in Σ_2^P. Set-entailment is coNP-hard and in Π_2^P.*

Proof. Since fuzzy equilibrium logic is a proper generalization of FASP, we can use its complexity results [23] to obtain Σ_2^P-membership for set-membership and existence and Π_2^P-membership for set-entailment. This result holds for regular FASP as well as for regular normal FASP. The hardness results are obtained by reducing the decision problems for normal resp. disjunctive ASP (see Table 1) to regular normal resp. regular FASP which is possible due to Proposition 9. □

Proof of Proposition 11

Proposition. *Set-membership, existence and set-entailment for regular definite FASP is in NP $\cap$ coNP.*

Proof. Each regular simple FASP program can be seen as a set of formulas in Łukasiewicz logic. Checking if such a set of formulas has a minimal fuzzy model can be polynomially reduced to checking the feasibility of a mixed integer program which is an NP-complete problem. Since the answer set is unique we obtain NP $\cap$ coNP for regular simple FASP for all decision problems. By Lemmas 1 and 2, it follows that it can be checked whether a fuzzy interpretation is an answer set of a regular definite FASP program by checking if it is the unique answer set of a particular regular simple FASP program and checking if a set of constraints is satisfied, hence we obtain the same results for regular definite FASP. □

Proof of Proposition 12

Proposition. *The unique answer set of a regular simple FASP program can be found in pseudo-polynomial time.*

Proof. Suppose m is the largest integer occurring in the program and n is equal to the size of the program. Then all constants $\overline{c}$ in the program are such that $c \in T = \{0, \frac{1}{k}, \ldots, \frac{k}{k}\}$ with k polynomial in m. After each application of Π_P, either the least fixpoint is found and the procedure terminates or the truth value of at least one atom is increased to a new value in T; hence there are at most $n \cdot k$ such iterations and the number of iterations is polynomial in m and n. □

Proof of Proposition 13

Proposition. *The complexity of the decision problems for regular simple and regular definite FASP is polynomial if all constants are polynomially bounded, i.e. all constants $\overline{c}$ in the program are such that $c \in \{0, \frac{1}{k}, \ldots, \frac{k}{k}\}$ with k polynomial in the size of the program.*

Proof. Follows from the proof of Proposition 12. □

Proof of Proposition 14

Proposition. *Consider a regular simple FASP program P such that the dependency graph $G(P)$ has no cycles and the longest path in $G(P)$ has length m. Then, the immediate consequence operator will only need m iterations to compute the answer set of P.*

Proof. Start with the fuzzy interpretation I_0 that maps all atoms to 0. Define A_0 as the set of all constants in P. Define A_1 as the set of all atoms a that only depend on constants: each rule with head $a \in A_1$ is of the form $a \leftarrow \overline{c}$. After one application of Π_P, each $a \in A_1$ is given a truth value

$$I_1(a) = \sup\{[r_b]_{I_0} \mid (a \leftarrow r_b) \in P\} = \sup\{c \mid (a \leftarrow \overline{c}) \in P\}.$$

In further applications of Π_P, the truth value of $a \in A_1$ will not increase since it only depends on constants.

Next, define A_2 as the set of all atoms $a \notin A_1$ such that for each rule $a \leftarrow f(b,d)$ in P, we have that $b, d \in A_0 \cup A_1$. After two applications of Π_P, each $a \in A_2$ is assigned a truth value

$$I_2(a) = \sup\{[r_b]_{I_1} \mid (a \leftarrow r_b) \in P\} = \sup\{[f(b,d)]_{I_1} \mid (a \leftarrow f(b,d)) \in P\}.$$

In further applications of Π_P, the truth value of $a \in A_2$ will not increase since it only depends on atoms for which we already know the truth value will not increase anymore.

Continuing this procedure, after k iterations of Π_P, we get fixed truth values for all atoms $a \in A_k$: atoms $a \notin \cup_{i=1}^{k-1} A_i$ such that for each rule $a \leftarrow f(b,d)$ in P we have that $b, d \in \cup_{i=0}^{k-1} A_i$. Another application of Π_P will give fixed values for the atoms in A_{k+1}. Since there are no cycles, we have that $A_k = \emptyset$ for $k > m$. Hence, after m iterations, the least fixpoint of Π_P has been found. □

Proposition 15

Proposition. *Consider a regular simple FASP program P and its unique answer set I. Suppose P contains the following set of rules*

$$\begin{array}{rcl} a_2 & \leftarrow & a_1 \oplus b_1 \\ a_3 & \leftarrow & a_2 \oplus b_2 \\ & \vdots & \\ a_1 & \leftarrow & a_n \oplus b_n \end{array}$$

and we have $I(b_j) > 0$ for at least one $j \in \{1, \ldots, n\}$. Then for each $i \in \{1, \ldots, n\}$ we have $I(a_i) = 1$.

Proof. If for all $i \in \{1, \ldots, n\}$ we have $I(a_i) + I(b_i) \leq 1$ and thus $[a_i \oplus b_i]_I = I(a_i) + I(b_i)$, then we have $I(a_1) < I(a_1)$ a contradiction. Indeed,

$$I(a_1) \leq I(a_1) + I(b_1) \leq I(a_2) \leq I(a_2) + I(b_2) \leq \ldots \leq I(a_j) < I(a_j) + I(b_j)$$

$$\leq I(a_{j+1}) \leq \ldots \leq I(a_n) + I(b_n) \leq I(a_1).$$

Thus, there has to be some $k \in \{1, \ldots, n\}$ such that $I(a_k) + I(b_k) > 1$, but then $I(a_{k+1}) = 1$ (or $I(a_1) = 1$ if $k = n$). This implies that $I(a_i) = 1$ for each $i \in \{1, \ldots, n\}$. □

Proof of Proposition 16

Proposition. *Consider a regular simple FASP program with only disjunctions in the bodies of rules and its unique answer set I. Suppose one of the rules in the program is of the form $a \leftarrow b \oplus d$ such that $I(b) > 0$ and a and d are atoms in the same strongly connected component S in $G(P)$. Then for all $s \in S$ we have $I(s) = 1$.*

Proof. Suppose $S = \{a_1, \ldots, a_n\}$ with $a = a_1$ and $d = a_2$. By the definition of a strongly connected component there must be path from a_1 to a_2 and so on until we reach a_n. Similary, one can also find a path from a_n to a_1. If we consider all corresponding rules in P, we have a cycle consisting of the elements in S that contains a with $a \leftarrow b \oplus d$ and $I(b) > 0$. By Proposition 15, we can conclude that $I(a_i) = 1$ for all $i \in \{1, \ldots, n\}$. □

Proof of Corollary 4

Corollary. *The unique answer set of a regular simple FASP program with only disjunction in the body of the rules can be found in polynomial time.*

Proof. Consider a regular simple FASP program P with only disjunctions in the bodies of rules. We can find the answer set I in polynomial time as follows. Using the algorithm of Tarjan [41] the strongly connected components in $G(P)$ can be identified in polynomial time. Next, for each strongly connected component S we introduce a fresh atom a_S. This is followed by defining a cycle free program P' that is obtained from P by replacing each atom from S by a_S and this for each strongly connected component S. Moreover, superfluous rules are removed in the sense that no rule appears more than once in P'. From P', we then obtain a program P'' as follows.

- Rules of the form $a \leftarrow b \oplus c$ where a, b and c are different atoms or constants remain unchanged.
- A rule of the form $a \leftarrow a \oplus b$ where $b \neq a$ with b an atom or a constant is replaced by $a \leftarrow (b > 0)$.
- A rule of the form $a \leftarrow a \oplus a$ is replaced by $a \leftarrow (a' > 0)$ with a fresh atom a'. In all other rules every occurrence of a is replaced by a'.

Note that these are the only possible types of rules in P'. The semantics for formulas of the form $(a > 0)$ are defined by increasing functions and hence the immediate consequence operator can be used to compute the minimal model of P'', see [35] for details. Since Proposition 14 does not exploit the fact that we have FASP programs under Łukasiewicz semantics, we can use the same proof to show that if formulas $(a > 0)$ are allowed in the bodies of rules that the immediate consequence operator will only need a polynomial number of steps to compute the minimal model I'' of P''. Finally we put $I(a) = I''(a_S)$ for each $a \in S$. By Remark 4 and Proposition 16 it follows that I is the answer set of P.

□

Proof of Proposition 17

Proposition. 1. *The set-membership and the existence problem for the class of regular normal FASP programs with only disjunction in the bodies of rules is in NP. Set-entailment is in coNP.*

2. *The set-membership and the existence problem for the class of regular normal FASP programs with cycle free dependency graphs is in NP. Set-entailment is in coNP.*

Proof. Let us first show that set-membership and existence are in NP. Suppose P is a regular normal FASP program in one of the subclasses subscribed in the statement of the proposition. From the analysis of the geometrical structure underlying fuzzy equilibrium models which is a proper generalization of regular FASP [23], it follows that a FASP program P has an answer set I such that $I(l) \geq \lambda_l$ for some $l \in \mathcal{L}_\mathcal{P}$ and $\lambda_I \in [0,1] \cap \mathbb{Q}$ iff there is such an answer set that can be encoded using a polynomial number of bits. So we can guess such an answer set I in polynomial time. The reduct P^I then belongs to the corresponding subclass of regular definite FASP programs. For set-membership and existence (special case with $\lambda_l = 0$) it then remains to be verified that I is an answer set of P^I. But this follows easy from the fact that the answer set of P^I can be determined in polynomial time. To show that set-entailment is in coNP, we need to show that the complement of the decision problem "Given a FASP program P, a literal l and a value λ_l, does there exist an answer set I of P such that $I(l) < \lambda_l$?" is in NP. By a similar result from [23], it follows that such an answer set, if it exists, can be guessed in polynomial time. The reduct then belongs to the corresponding subclass of regular definite FASP programs for which the unique answer set can be determined in polynomial time. □

Proof of Corollary 5

Corollary. *Existence and set-membership for regular normal FASP with polynomially bounded constants is NP-hard. Set-entailment is coNP -hard.*

Proof. By Proposition 9, it follows that a normal ASP program can be reduced to a regular normal FASP program with polynomially bounded constants. Since set-membership and existence for normal ASP are NP-complete (Table 1), it then follows that these decision problems are NP-hard for regular normal FASP with polynomially bounded constants. The fact that set-entailment is coNP-hard follows from the coNP-completeness for normal ASP (Table 1). □

Proof of Proposition 18

Proposition. *Given a strict disjunctive FASP program P not containing strong negation such that $\mathcal{B}_P = \{a_1, \ldots, a_n\}$. Define the following bilevel linear program Q_P.*

$$
\begin{aligned}
&\tilde{\mathbf{a}}'^* = \arg\min_{\tilde{\mathbf{a}}'} \textstyle\sum_{i=1}^{n} (\tilde{a}'_i - \tilde{a}^*_i) \\
&\textit{s.t. } 0 \leq \tilde{a}'_i \leq 1 \\
&\qquad\quad \tilde{\mathbf{a}}^* = \arg\min_{\tilde{\mathbf{a}}} \textstyle\sum_{i=1}^{n} \tilde{a}_i \\
&\qquad\quad \textit{s.t. } \tilde{a}_i \leq \tilde{a}'_i, 0 \leq \tilde{a}_i \leq 1 \textit{ and} \\
&\qquad\qquad (\textstyle\sum_{j=1}^{m} x_j) - m + 1 \leq \textstyle\sum_{i=1}^{n} l_i \textit{ for each rule } (1) \\
&\qquad\qquad \textit{in the reduct of } P \textit{ w.r.t. } \tilde{\mathbf{a}}'
\end{aligned}
$$

Then

1. *If Q_P has a solution $\tilde{\mathbf{a}}^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$, $\tilde{\mathbf{a}}'^* = (\tilde{a}'_1, \ldots, \tilde{a}'_n)$ such that the objective function of the upper level problem is evaluated to 0, then $I : \mathcal{B}_P \to [0,1] : a_i \mapsto \tilde{a}_i$ is an answer set of P.*
2. *If I is an answer set of P, then $\tilde{\mathbf{a}}^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$, $\tilde{\mathbf{a}}'^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$ where $I(a_i) = \tilde{a}_i$ for each $i \in \{1, \ldots, n\}$ is a solution of Q_P such that the upper level problem is evaluated to 0.*

Proof. 1. Suppose Q_P has a solution $\tilde{\mathbf{a}}^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$, $\tilde{\mathbf{a}}'^* = (\tilde{a}'_1, \ldots, \tilde{a}'_n)$ such that the objective function of the upper level problem is evaluated to 0. We show that $I : \mathcal{B}_P \to [0, 1] : a_i \mapsto \tilde{a}_i$ is a minimal fuzzy model of P^I. First note that if $\sum_{i=1}^{n}(\tilde{a}'_i - \tilde{a}_i) = 0$ it must hold that $\tilde{a}'_i = \tilde{a}_i$ for all $i \in \{1, \ldots, n\}$ since we have the constraints $\tilde{a}_i \leq \tilde{a}'_i$. By the constraints in the lower level problem it then follows that I is a fuzzy model of P^I. Now suppose there exists a fuzzy model J of P^I such that $J < I$. Then $\tilde{\mathbf{a}}^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$, $\hat{\mathbf{a}}'^* = (\hat{a}'_1, \ldots, \hat{a}'_n)$ where $J(a_i) = \hat{a}'_i$ is a solution of Q_P with $\sum_{i=1}^{n}(\hat{a}'_i - \tilde{a}_i) < \sum_{i=1}^{n}(\tilde{a}'_i - \tilde{a}_i)$, a contradiction.

2. Suppose I is an answer set of P. We need to show that $\tilde{\mathbf{a}}^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$, $\tilde{\mathbf{a}}'^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$ is a solution of Q_P. As in the proof of Proposition 8, we can show that if the leader makes a choice $\tilde{\mathbf{a}}'^* = (a'_1, \ldots, a'_n)$ which can be seen as a fuzzy interpretation $I'(a_i) = a'_i$ of P, that $\tilde{\mathbf{a}}^* = (a^*_1, \ldots, a^*_n)$ where $J(a_i) = a^*_i$ is a minimal fuzzy model of $P^{I'}$ are the possible optimal solutions of the lower level problem. Since $\tilde{\mathbf{a}}^* = \tilde{\mathbf{a}}'^*$ and the fact that if the leader makes the choice $\tilde{\mathbf{a}}'^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$, that $\tilde{\mathbf{a}}^* = (\tilde{a}_1, \ldots, \tilde{a}_n)$ is an optimal solution of the lower level problem, we have found a solution of Q_P.

$\square$

References

[1] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003.

[2] T. Eiter, G. Gottlob, Complexity results for disjunctive logic programming and application to nonmonotonic logics, in: Proceedings of the International Logic Programming Symposium, 1993, pp. 266–278.

[3] J. Janssen, S. Schockaert, D. Vermeir, M. De Cock, A core language for fuzzy answer set programming, International Journal of Approximate Reasoning 53 (2012) 660–692.

[4] D. Van Nieuwenborgh, M. De Cock, D. Vermeir, An introduction to fuzzy answer set programming, Annals of Mathematics and Artificial Intelligence 50 (3-4) (2007) 363–388.

[5] P. Hájek, Metamathematics of Fuzzy Logic, Trends in Logic, 1998.

[6] R. McNaughton, A theorem about infinite-valued sentential logic, The Journal of Symbolic Logic 16 (1) (1951) 1–13.

[7] R. Hähnle, Proof theory of many-valued logic - linear optimization - logic design: connections and interactions, Soft Computing 1 (1997) 107–119.

[8] C. Damásio, L. Pereira, Antitonic logic programs, in: Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning, 2001, pp. 379–392.

[9] Y. Loyer, U. Straccia, Epistemic foundation of stable model semantics, Theory and Practice of Logic Programming 6 (4) (2006) 355–393.

[10] T. Łukasiewicz, U. Straccia, Tightly integrated fuzzy description logic programs under the answer set semantics for the semantic web, in: Proceedings of the 1st International Conference on Web Reasoning and Rule Systems, 2007, pp. 289–298.

[11] N. Madrid, M. Ojeda-Aciego, Measuring inconsistency in fuzzy answer set semantics, IEEE Transactions on Fuzzy Systems 19 (4) (2011) 605–622.

[12] U. Straccia, Query answering in normal logic programs under uncertainty, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 2005, pp. 687–700.

[13] U. Straccia, Annotated answer set programming, in: Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 2006, pp. 1212–1219.

[14] U. Straccia, Query answering under the any-world assumption for normal logic programs, in: Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning, 2006, pp. 329–339.

[15] D. Dubois, H. Prade, Possibility theory, probability theory and multiple-valued logics: a clarification, Annals of Mathematics and Artificial Intelligence 32 (1-4) (2001) 35–66.

[16] K. Bauters, S. Schockaert, M. De Cock, D. Vermeir, Possibilistic answer set programming revisited, in: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, 2010, pp. 48–55.

[17] C. Baral, M. Gelfond, J. Rushton, Probabilistic reasoning in computer science, in: Logic Programming and Nonmonotonic Reasoning, 7th International Conference, 2004, pp. 21–33.

[18] A. Dekhtyar, V. Subrahmanian, Hybrid probabilistic programs, in: Proceedings of the 14th International Conference on Logic Programming, 1997, pp. 391–405.

[19] D. Mundici, Satisfiability in many-valued sentential logic is NP-complete, Theoretical Computer Science 52 (5) (1987) 145–153.

[20] T. Łukasiewicz, Many-valued disjunctive logic programs with probabilistic semantics, in: Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning, 1999, pp. 277–289.

[21] F. Bobillo, F. Bou, U. Straccia, On the failure of the finite model property in some fuzzy description logics, Fuzzy Sets and Systems 172 (23) (2011) 1–12.

[22] M. Cerami, U. Straccia, On the (un)decidability of fuzzy description logics under łukasiewicz t-norm, Information Sciences 227 (2013) 1–21.

[23] S. Schockaert, J. Janssen, D. Vermeir, Fuzzy equilibrium logic: Declarative problem solving in continuous domains, ACM Transactions on Computational Logic 13 (4) (2012) 111–155.

[24] J. Bard, J. Falk, An explicit solution to the multi-level programming problem, Computers and Operations Research 9 (1982) 77–100.

[25] W. Candler, R. Townsley, A linear two-level programming problem, Computers and Operations Research 9 (1982) 59–76.

[26] C. Shi, J. Lu, G. Zhang, H. Zhou, An extended branch and bound algorithm for linear bilevel programming, Applied Mathematics and Computation 180 (2) (2006) 529–537.

[27] J. Bard, J. Moore, A branch and bound algorithm for the bilevel programming problem, SIAM Journal on Scientific and Statistical Computation 11 (1990) 281–292.

[28] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: Proceedings of the Fifth International Conference and Symposium on Logic Programming, 1988, pp. 1070–1080.

[29] J. Janssen, S. Schockaert, D. Vermeir, M. De Cock, General fuzzy answer set programs, in: Proceedings of the International Workshop on Fuzzy Logic and Applications, 2009, pp. 353–359.

[30] U. Straccia, M. Ojeda-Aciego, C. V. Damásio, On fixed-points of multi-valued functions on complete lattices and their application to generalized logic programs, SIAM Journal on Computing 8 (5) (2009) 1881–1911.

[31] L. Yu, N. Wang, X. Meng, Real-time forest fire detection with wireless sensor networks, in: Proceedings of the Wireless Communication Networking and Mobile Computing International Conference, 2005, pp. 1214–1217.

[32] J. Castro, E. Trillas, J. Zurita, Non-monotonic fuzzy reasoning, Fuzzy Sets and Systems 94 (1998) 217–225.

[33] S. Schockaert, M. De Cock, E. Kerre, Spatial reasoning in a fuzzy region connection calculus, Artificial Intelligence 173 (2009) 258–298.

[34] N. Megiddo, T. A., New results on the complexity of p-center problems, SIAM Journal on Computing 12 (4) (1983) 751–758.

[35] J. Janssen, Foundations of fuzzy answer set programming, Ph.D. thesis, Ghent University and Vrije Universiteit Brussel (2011).

[36] S. Cook, The complexity of theorem-proving procedures, in: Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing, 1971, pp. 151–158.

[37] N. Madrid, M. Ojeda-Aciego, On the existence and unicity of stable models in normal residuated logic programs, International Journal of Computer Mathematics 89 (3) (2012) 310–324.

[38] T. Lukasiewicz, Fuzzy description logic programs under the answer set semantics for the semantic web, Fundamenta Informaticae 82 (3) (2008) 289–310.

[39] T. Gawlitza, H. Seidle, Precise fixpoint computation through strategy iteration, in: Proceedings of the 16th European Conference on Programming, 2007, pp. 300–315.

[40] H. Bjorklund, S. Sandberg, S. Vorobyov, Complexity of model checking by iterative improvement: the pseudo-boolean framework, in: Proceedings of the 5th Andrei Ershov Memorial Conference "Perspectives of System Informatics", 2003, pp. 381–394.

[41] R. Tarjan, Depth-first search and linear graph algorithms, SIAM Journal on Computing 1 (2) (1972) 146–160.

[42] J. Bard, Practical Bilevel Optimization: Algorithms and Applications, Kluwer Academic Publishers: USA, 1998.