Complexity of fuzzy answer set programming under Łukasiewicz semantics: First results

Marjon Blondeel[1] *, Steven Schockaert[2] **, Martine De Cock[2], and Dirk Vermeir[1]

[1] Dept. of Computer Science, Vrije Universiteit Brussel, Belgium
{mblondee,dvermeir}@vub.ac.be
[2] Dept. of Applied Mathematics and Computer Science, Ghent University, Belgium
{steven.schockaert,martine.decock}@ugent.be

Abstract. Fuzzy answer set programming (FASP) has recently been proposed as a generalization of answer set programming in which propositions are allowed to be graded. Little is known about its computational complexity. In this paper we present some results and reveal a connection to an open problem about integer equations, suggesting that characterizing the complexity of FASP may not be straightforward.

1 Introduction

Answer set programming (ASP) is a form of declarative programming. Basically, a problem is translated to a logic program under the stable model semantics [6], also called an ASP program. The answer sets (i.e. the stable models) of the program then correspond to the solutions of the initial problem. A disjunctive ASP program is 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 atoms and "not " the negation-as-failure operator, denoting that "not a" is true if there is no proof to support a. A disjunctive ASP program is called positive if there occurs no negation-as-failure in the program. A normal ASP program is a set of rules with $n = 1$ and a normal program which is positive is called a simple ASP program. For a disjunctive ASP program P and an atom a, deciding whether a occurs in an answer set of P is Σ_2^P-complete [4]. For normal programs this problem is NP-complete and for simple problems it is P-complete [1].

Although ASP has been successfully applied, e.g. to solve combinatorial optimization problems, it is not directly suitable for modeling problems with continuous domains. Fuzzy answer set programming (FASP) [7], a generalization of ASP, allows to model continuous systems by using an infinite number of truth values, which correspond to intensities of properties. To define relationships between the atoms, we will restrict to the connectives of Łukasiewicz logic, one of the most widely-used fuzzy logics. In this setting, FASP relates to Łukasiewicz logic as ASP does to classical logic. For Łukasiewicz logic, satisfiability is an

* Funded by a joint Research Foundation-Flanders (FWO) project
** Postdoctoral fellow of the Research Foundation-Flanders (FWO)

NP-complete problem [10]. Since this problem has the same complexity for classical logic, one would expect ASP and FASP to have the same complexity as well. In the case of probabilistic ASP, the complexity of finding some stable model of a disjunctive program has been shown to be Σ_2^P-complete [8]. In [9], complexity results are shown when restricting to minimum in the body and maximum in the head. In our paper, for disjunctive FASP programs we will show Σ_2^P-completeness by using the complexity of fuzzy equilibrium logic [11], but for simple and normal programs we can only show coNP- and Σ_2^P-membership respectively. We furthermore show that characterizing the complexity of simple programs is equivalent to an existing open problem about integer equations [5], which in turn relates to an open problem about the complexity of optimizing pseudo-boolean functions [2]. This suggests that the problem of characterizing the complexity of FASP is not likely to have an easy solution. However, we introduce several subclasses of simple programs for which P-membership can be shown. Furthermore, if restricting the syntax of disjunctive FASP to only Łukasiewicz conjunction and maximum in the body and Łukasiewicz disjunction and minimum in the head, we will be able to show NP-membership.

2 Background

Formulas in Łukasiewicz logic are built from a set of atoms A, the truth constants in $[0,1] \cap \mathbb{Q}$ and the connectives conjunction $\otimes$, disjunction $\oplus$, max, min, implication $\rightarrow$ and negation $\neg$. An interpretation is a mapping $I : A \rightarrow [0,1]$. We can extend this interpretation to arbitrary formulas as follows.

- $[c]_I = c$, $[\alpha \otimes \beta]_I = \max([\alpha]_I + [\beta]_I - 1, 0)$, $[\alpha \oplus \beta]_I = \min([\alpha]_I + [\beta]_I, 1)$
- $[\min(\alpha,\beta)]_I = \min([\alpha]_I, [\beta]_I)$, $[\max(\alpha,\beta)]_I = \max([\alpha]_I, [\beta]_I)$
- $[\alpha \rightarrow \beta]_I = \min(1 - [\alpha]_I + [\beta]_I, 1)$, $[\neg\alpha]_I = 1 - [\alpha]_I$

for $c \in [0,1]$ and α and β formulas. Let us briefly introduce a fuzzy version of answer set programming based on [7]. A *disjunctive FASP program* is a set of rules of the form $r : \beta \leftarrow \alpha$, with α and β formulas built from atoms in A, constants in $\mathbb{Q} \cap [0,1]$, expressions of the form "not a" with $a \in A$ (with not the negation-as-failure operator), the connectives $\otimes$, $\oplus$, min and max, but restricting to formulas β without "not ". The formula β is called the head of r and α is the body. Although the rules in disjunctive FASP are not restricted to having only disjunctions in the head, we will use the label "disjunctive" since in all rules there is in some sense a disjunctive property: an atom in the body does not necessarily have an equal influence on the truth values of each atom in the head. We denote the set of atoms occurring in a disjunctive program P as $\mathcal{B}_P$. An *interpretation* I of a disjunctive program P is a mapping $I : \mathcal{B}_P \rightarrow [0,1]$, which can be extended to arbitrary rules in the same manner as for formulas in Łukasiewicz logic, and for an atom a we define $[\text{not } a]_I = 1 - [a]_I$. For interpretations I_1 and I_2 we say that $I_1 \leq I_2$ iff $I_1(a) \leq I_2(a)$ for all $a \in \mathcal{B}_P$. An interpretation I is called a *model* of P iff $[r]_I = 1$ for all $r \in P$. If in each rule the head contains exactly one atom, the program is called a *normal program.* A disjunctive program is called *positive* if it contains no expressions of the form "not a". A normal program which is positive is called a *simple program.* An interpretation I is called an *answer set* of a positive disjunctive program P iff it is a minimal model of P. For disjunctive programs which are not positive, answer sets are defined using a generalization of the Gelfond-Lifschitz reduct. Specifically, let P be a disjunctive program and I an interpretation. The reduct P^I of the program P is obtained from P by replacing all expressions "not a" by the interpretation $[\text{not } a]_I$. The reduct is then a positive program. We say that an interpretation I is an answer set of P iff I is an answer set of P^I. Note that without loss of generality, we may assume that in each rule of a disjunctive program, the head and the body have exactly two arguments. Indeed, a disjunctive program can be rewritten to a disjunctive program with the same models and with only rules of the form $g(a_1, a_2) \leftarrow f(l_1, l_2)$ with a_1 and a_2 atoms and/or constants, l_1 and l_2 atoms, constants and/or of the form "not a" with a an atom and f and g prefix notations for $\otimes$, $\oplus$, min or max.

3 Complexity results

In this section, we will consider the following decision problem. Given a disjunctive FASP program P, an atom a and a value $\lambda_a \in [0, 1] \cap \mathbb{Q}$, is there an answer set I of P such that $I(a) \geq \lambda_a$? We will refer to this decision problem as the *existence problem.* Our results are listed in Table 1 and discussed below.

Table 1. Complexity of disjunctive (FASP) programs

	simple (3.1)	normal (3.2)	disjunctive (3.3)
ASP	in P	NP-complete	Σ_2^P-complete
FASP: no restrictions	in coNP	NP-hard, in Σ_2^P	Σ_2^P-complete
FASP: only $\otimes$ and max in body, only $\oplus$ and min in head	in P	in NP	in NP
FASP: only $\oplus$ in body	in P	in NP	Σ_2^P
FASP: cycle free	in P	in NP	n.a.
FASP: polynomially bounded constants	in P	NP-complete	Σ_2^P-complete

3.1 Complexity for simple FASP programs

A rule $\beta \leftarrow \alpha$ in a disjunctive FASP program can be seen as the Łukasiewicz formula $\alpha \rightarrow \beta$, which means that the existence problem can be reduced to entailment checking in Łukasiewicz logic, extended with truth constants (i.e. Rational Pavelka logic). Indeed, a simple program P has a unique minimal model I, thus checking if $I(a) \geq \lambda_a$ is the same as checking wheter $J(a) \geq \lambda_a$ for all models J of P, thus the same as checking whether $a \leftarrow \lambda_a$ can be entailed by P. As entailment checking in Rational Pavelka logic is known to be coNP-complete, we find that the existence problem belongs to coNP. Although for simple programs there is coNP-membership, the connection to an existing open problem suggests that P-membership will be difficult to show. More precisely, the unique minimal model of a simple program P can be found by computing the least solution of a system of equations over the integers with addition, multiplication with positive constants, maximum and minimum. In [5], 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. [2]) whether it has polynomial time complexity. In general, we can use the immediate consequence operator [3] to find the unique minimal model of a simple program. The immediate consequence operator for a simple program P is a function Π_P that maps interpretations to interpretations and is defined as $\Pi_P(I)(a) = \sup\{[\alpha]_I \mid (a \leftarrow \alpha) \in P\}$ for an interpretation I and $a \in \mathcal{B}_P$. The minimal model of P equals the least fixpoint of Π_P [3]. Unfortunately, if for instance the program contains the rule $a \leftarrow a \oplus \frac{1}{2^n}$, with n the total number of atoms occurring in P and a not appearing in any other rule in P, then 2^n iterations will be needed to conclude that a should have truth value 1. In general, there seems to be a problem if there are "loops" in the program. These loops are actually the cycles in a directed graph, the so-called *depency graph* of the program. For each simple (or normal) program P we define the depency graph $G(P)$ as follows. The vertices are the atoms in the program and there is a directed edge from atom a to atom b if a occurs in the body of a rule with head b.

Polynomially bounded constants or cycle free If the dependency graph of a simple program P has no cycles, Π_P will need only a polynomial number of iterations to find the least fixpoint. If cycles are allowed but we demand that all constants in the program are polynomially bounded, i.e. they are elements of $T = \{0, \frac{1}{k}, \ldots, \frac{k}{k}\}$ with k polynomial in the size of the problem n, then the answer set will be found in polytime as well. Indeed, after every 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.

Only disjunction in the body For simple programs with only disjunctions in the bodies, the problem of the cycles can be tackled in another way. A directed graph is called *strongly connected* if for each two vertices u and v there is a path from u to v and a path from v to u. The *strongly connected components* (scc) of a directed graph are its maximal strongly connected subgraphs and can be seen as generalizations of cycles. We identify each strongly connected component with its vertices. Now consider a simple program P and its unique answer set I. Suppose there is a rule $c \leftarrow a \oplus b$ such that c and a are elements in the same strongly connected component S of $G(P)$ and $I(b) > 0$, then we can prove that $I(s) = 1$ for each $s \in S$. This result and the fact that Tarjan's algorithm [12] can be used to efficiently compute the partition of the vertices in the strongly connected components of a directed graph induces a modification of the immediate consequence operator such that it runs in polytime.

$$\Pi_P(I)(a) = \begin{cases} 1 & \text{if } (a \leftarrow b \oplus d) \in P, \\ & I(b) > 0 \text{ (or } I(d) > 0), \\ & a, d \text{ (or } a, b) \text{ in the same scc} \\ \sup\{[\alpha]_I \mid (a \leftarrow \alpha) \in P\} & \text{otherwise} \end{cases}$$

Only conjunction and maximum in the body For simple rules with only conjunction and/or maximum in the body we do not need the immediate consequence operator to find the answer set. Rules of the form $a \leftarrow \max(b, c)$ can be rewritten as two rules $a \leftarrow b$ and $a \leftarrow c$ and an interpretation I models $w \leftarrow u \otimes v$ iff it models the Łukasiewicz formula $\neg u \oplus \neg v \oplus w$. To find the minimal model one can use linear programming, which is known to be in P. Indeed, the function to be minimized is the sum of all atoms in the program. For a rule $a \leftarrow b$ we add the constraints $b \leq a$ and $0 \leq a, b \leq 1$ and for $w \leftarrow u \otimes v$ we add the constraints $u' + v' + w \geq 1$, $u' = 1 - u$, $v' = 1 - v$ and $0 \leq u, u', v, v', w \leq 1$.

3.2 Complexity for normal FASP programs

From the analysis of the geometrical structure underlying fuzzy equilibrium models [11], it follows that a program P has an answer set I such that $I(a) \geq \lambda_a$ iff there is such an answer set that can be encoded using a polynomial number of bits. This means that if the existence problem is in P for a subclass of simple programs, then the existence problem for the corresponding subclass of normal programs is in NP. For normal programs in classical ASP, the existence problem is NP-complete [1]. By reduction, we can show NP-hardness for normal FASP programs. More precisely, classical connectives are replaced by the corresponding connectives in Łukasiewicz logic and for each atom a, the rule $a \leftarrow a \oplus a$ is added to the FASP program, which ensures that the truth value of every atom is in $\{0, 1\}$. Note that we can also reduce normal ASP to normal FASP restricted to polynomially bounded constants. Remark that disjunctions in the body are needed to show NP-hardness.

3.3 Complexity for disjunctive FASP programs

From the complexity of fuzzy equilibrium logic [11], it follows that the existence problem for disjunctive programs is in Σ_2^P. Disjunctive ASP, which is Σ_2^P-hard [4], can be reduced to disjunctive FASP, in the same manner as for normal programs. This can also be done if the program is restricted to polynomially bounded constants. For programs with only $\otimes$ and max in the body and $\oplus$ and min in the head we can prove NP-membership. Recall that an answer set I can be guessed in polynomial time. It has to be checked whether I is a model of P^I. The latter can be done by a linear program M, similar as in Section 3.1. Finally, to prove that it is a minimal model, a linear program M_a has to be solved for each atom a in P^I. This program has the same constraints as M and the constraints $a < I(a)$ and $b \leq I(b)$ for all atoms $b \neq a$. If M_a has a solution,

I is not an answer set of P. Note that the strict inequality can be handled by assuming a weak inequality $a \leq I(a)$, finding the solution which minimizes a and verifying whether in that solution the value of a is different from $I(a)$.

4 Conclusions

We presented some results about the computational complexity of FASP with Łukasiewicz semantics. For disjunctive FASP, this is the same as for disjunctive ASP. However, NP-membership was shown when restricting to disjunction and minimum in the head and conjunction and maximum in the body. For simple 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.

References

1. Baral, C. *Knowledge, Representation Reasoning and Declarative Problem Solving.* Cambridge University Press, 2003.
2. Bjorklund, H., Sandberg, S., and Vorobyov, S. 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.
3. Damásio, C., and Pereira, L. Antitonic logic programs. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning* (2001), pp. 379–392.
4. Eiter, T., and Gottlob, G. Complexity results for disjunctive logic programming and application to nonmonotonic logics. In *Proceedings of the Internationsl Logic Programming Symposium* (1993), pp. 266–278.
5. Gawlitza, T., and Seidle, H. Precise fixpoint computation through strategy iteration. In *Proceedings of the 16th European Conference on Programming* (2007), pp. 300–315.
6. Gelfond, M., and Lifschitz, V. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming* (1988), pp. 1070–1080.
7. Janssen, J., Schockaert, S., Vermeir, D., and De Cock, M. General fuzzy answer set programs. In *Proceedings of the International Workshop on Fuzzy Logic and Applications* (2009), pp. 353–359.
8. Łukasiewicz, T. Many-valued disjunctive logic programs with probabilistic semantics. In *LPNMR* (1999), pp. 277–289.
9. Mateis, C. Extending disjunctive logic programming by t-norms*. In *LPNMR* (1999), pp. 290–304.
10. Mundici, D. Satisfiability in many-valued sentential logic is NP-complete. *Theoretical Computer Science 52*, 5 (1987), 145–153.
11. Schockaert, S., Janssen, J., Vermeir, D., and De Cock, M. Answer sets in a fuzzy equilibrium logic. In *Proceedings of the 3rd International Conference in Web Reasoning and Rule Systems* (2009), pp. 135–149.
12. Tarjan, R. Depth-first search and linear graph algorithms. *SIAM Journal on Computing 1*, 2 (1972), 146–160.