

# *Fuzzy Answer Sets Approximations*

MARIO ALVIANO\*

*Department of Mathematics and Computer Science, University of Calabria, 87036 Rende (CS), Italy  
(e-mail: alviano@mat.unical.it)*

RAFAEL PEÑALOZA†

*Dresden University of Technology, 01062 Dresden, Germany  
Center for Advancing Electronics Dresden  
(e-mail: penaloza@tcs.inf.tu-dresden.de)*

*submitted 10 April 2013; revised 23 May 2013; accepted 23 June 2013*

---

## Abstract

Fuzzy answer set programming (FASP) is a recent formalism for knowledge representation that enriches the declarativity of answer set programming by allowing propositions to be graded. To now, no implementations of FASP solvers are available and all current proposals are based on compilations of logic programs into different paradigms, like mixed integer programs or bilevel programs. These approaches introduce many auxiliary variables which might affect the performance of a solver negatively. To limit this downside, operators for approximating fuzzy answer sets can be introduced: Given a FASP program, these operators compute lower and upper bounds for all atoms in the program such that all answer sets are between these bounds. This paper analyzes several operators of this kind which are based on linear programming, fuzzy unfounded sets and source pointers. Furthermore, the paper reports on a prototypical implementation, also describing strategies for avoiding computations of these operators when they are guaranteed to not improve current bounds. The operators and their implementation can be used to obtain more constrained mixed integer or bilevel programs, or even for providing a basis for implementing a native FASP solver. Interestingly, the semantics of relevant classes of programs with unique answer sets, like positive programs and programs with stratified negation, can be already computed by the prototype without the need for an external tool.

**KEYWORDS:** Fuzzy logic, answer set programming, search-space pruning operators

---

## 1 Introduction

Answer Set Programming (ASP), i.e., logic programming under stable model semantics (Gelfond and Lifschitz 1991), is a declarative language for knowledge representation (Niemelä 1999; Marek and Truszczyński 1999; Lifschitz 2002). In ASP, problems are modeled by specifying a set of requirements that all solutions, called answer sets, have to satisfy. One of the strengths of ASP is its capability to model non-monotonic knowledge, overcoming a well-known limitation of classical logic, which can only deal with monotonic inferences. While monotonicity is desired in mathematics, it is widely considered a weakness for knowledge representation (Baral 2003), where non-monotonicity arises in

\* Partially supported by Regione Calabria within the PIA project KnowRex POR FESR 2007–2013.

† Partially supported by DFG under grant BA 1122/17-1 and within the Cluster of Excellence ‘cfAED’.

common reasoning tasks such as reasoning by default, abductive reasoning and belief revision. ASP can handle these tasks naturally (Marek and Remmel 2004; Lin and You 2002; Delgrande et al. 2008), allowing for modeling and reasoning on incomplete information, and possibly retracting some conclusions as new knowledge on the application domain is acquired. Since complete knowledge can only be achieved in mathematical abstraction, it can be stated that ASP makes logic closer to the real world.

However, ASP is still based on precise information, which cannot always be assumed in the real world. For example, measures provided by any instrument or sensor always come with some degree of tolerance, and information expressed in natural language are often vague: there is no precise way to distinguish persons that are tall from those who are not. Fuzzy logic (Dubois et al. 1991) can handle vague information of this kind by interpreting propositions with a truth degree in the interval of real numbers  $[0, 1]$ . Intuitively, the higher the degree assigned to a proposition, the *more true* it is, with the extreme elements 0 and 1 denoting *totally false* and *totally true*, respectively. Consider for example the *Barber of Seville* paradox: In the small town of Seville, all and only those men who do not shave themselves are shaved by the barber. Classical set theory can neither prove nor disprove that the barber shaves himself, hence a fuzzy interpretation of this proposition should be  $1/2$ , the most undetermined truth degree.

Fuzzy Answer Set Programming (FASP) aims at combining ASP and fuzzy logic. For example, a FASP encoding of the Barber of Seville paradox is

$$\text{shaves}(\text{barber}, X) \leftarrow \text{not shaves}(X, X) \quad \text{shaves}(X, X) \leftarrow \text{not shaves}(\text{barber}, X)$$

from which  $\text{shaves}(\text{barber}, \text{barber})$  gets the expected truth degree  $1/2$ . FASP has been defined for a very general framework (Nieuwenborgh et al. 2007b), allowing several connectors to be combined in the same program. With the aim of providing an indication for implementing a FASP solver, more constrained frameworks have been considered by Lukasiewicz (2006) and Janssen et al. (2012). However, current proposals are based on compilations into different paradigms and introduce many auxiliary variables which could affect performance negatively. The focus of this paper is on operators for approximating fuzzy answer sets. These operators can be used for limiting the search space of an external tool, such as the linear or bilevel program approaches proposed by Lukasiewicz and Janssen et al., or as the basis for implementing a native FASP solver.

More precisely, in this paper we describe operators for computing two fuzzy interpretations, called lower and upper bound, such that every answer set is between these bounds. We introduce fuzzy unfounded sets, which generalize the notion of unfounded sets of classical ASP to deal with fuzzy semantics. We also define a well-founded operator that combines the fuzzy  $T_P$  operator with the complement of the greatest unfounded set to improve on previously computed bounds. We show that these operators yield answer sets for positive and stratified FASP programs in polynomial time, while in general produce the well-founded semantics by Damásio and Pereira (2001). For dealing with normal FASP programs, we introduce the new minimal satisfiability operator  $S_P$ . For semantics based on the Lukasiewicz t-norm, this operator is polynomial-time computable and can further tighten the approximation. Finally, we describe a prototypical implementation which combines optimization ideas from classical ASP solvers (Gebser et al. 2007; Lierler and Maratea 2004; Alviano et al. 2011), and report on an experiment assessing the potential performance gain provided by our operators to a bilevel program solver.

## 2 Syntax and Semantics

Let  $\mathcal{B}$  be a fixed, finite set of propositional atoms. A *fuzzy atom* (or *atom* for short) is either a propositional atom from  $\mathcal{B}$  or a numeric constant in the range  $[0, 1]$ , where numeric constants are overlined, e.g.  $\bar{1}$ , to distinguish them from propositional atoms. A *literal* is either a fuzzy atom or a fuzzy atom preceded by the default negation symbol *not*. A *normal FASP program* is a finite set of rules of the form

$$a \leftarrow b_1 \otimes \cdots \otimes b_m \otimes \text{not } b_{m+1} \otimes \cdots \otimes \text{not } b_n \quad (1)$$

where  $n \geq m \geq 0$ ,  $a, b_1, \dots, b_n$  are atoms, and  $\otimes$  denotes the fuzzy conjunction. For a rule  $r$  of the form (1), the atom  $a$  is called the *head* of  $r$ , denoted  $H(r)$ , and the conjunction  $b_1 \otimes \cdots \otimes b_m \otimes \text{not } b_{m+1} \otimes \cdots \otimes \text{not } b_n$  is called the *body* of  $r$ , denoted  $B(r)$ . The expressions  $B^+(r) = \{b_1, \dots, b_m\}$  and  $B^-(r) = \{b_{m+1}, \dots, b_n\}$  denote the multiset of positive and negative body literals of  $r$ , respectively. Multisets thus represent conjunctions: a multiset  $A = \{a_1, \dots, a_k\}$  ( $k \geq 0$ ) of literals represents the conjunction  $\bigotimes_{i=1}^k a_i$ . Moreover, *not*  $A$  denotes the multiset  $\{\text{not } a_1, \dots, \text{not } a_k\}$ , i.e.,  $\bigotimes_{i=1}^k \text{not } a_i$ . A rule  $r$  is *positive* if  $B^-(r) = \emptyset$ , and a *fact* if  $B^+(r) = B^-(r) = \emptyset$ . Relevant subclasses of normal FASP programs are the positive and stratified programs. A program is *positive* if all of its rules are positive. The notion of stratified program requires the introduction of the dependency graph  $G_P = (\mathcal{B}, A)$  of a program  $P$ , where  $A$  contains arcs  $a \rightarrow^+ b_i$  and  $a \rightarrow^- b_j$  ( $1 \leq i \leq m < j \leq n$ ) for each rule  $r \in P$  of the form (1).  $P$  is *stratified* if no cycle in  $G_P$  contains  $\rightarrow^-$  arcs.

The semantics of FASP programs generalizes that of ASP by interpreting propositional atoms with a truth degree from the interval  $[0, 1]$ . An additional degree of liberty arises from the choice of the operator used to interpret fuzzy conjunctions. We focus on semantics based on t-norms (Klement et al. 2000). A *t-norm* is a binary, associative and commutative operator  $\otimes : [0, 1] \times [0, 1] \rightarrow [0, 1]$  that is monotonic and has unit 1, i.e., for every  $x, y, z \in [0, 1]$ ,  $x \leq y$  implies  $x \otimes z \leq y \otimes z$ , and  $x \otimes 1 = x$ . There are three fundamental t-norms, called the *Gödel*, *Lukasiewicz*, and *product* t-norms, where  $x \otimes y$  is defined as  $\min\{x, y\}$ ,  $\max\{x + y - 1, 0\}$ , and  $x \cdot y$ , respectively. In the following, FASP programs are assumed to be associated with a fixed t-norm computable in polynomial time. A *fuzzy interpretation*  $I$  for a FASP program  $P$  is a fuzzy set in  $\mathcal{B}$ , i.e., a function  $I : \mathcal{B} \rightarrow [0, 1]$  mapping each propositional atom of  $\mathcal{B}$  into a truth degree in  $[0, 1]$ . The interpretation  $I$  is extended to numeric constants, negative literals and multisets as follows. For a constant  $\bar{c}$ ,  $I(\bar{c}) = c$ ; for a negative literal *not*  $b$ ,  $I(\text{not } b) = 1 - I(b)$ ; for a multiset of literals  $A = \{l_1, \dots, l_k\}$ ,  $I(A) = \bigotimes_{i=1}^k I(l_i)$ .

Let  $\mathcal{I}$  be the set of all interpretations and  $I, J \in \mathcal{I}$ .  $I$  is a *subset* of  $J$  ( $I \subseteq J$ ) if  $I(a) \leq J(a)$  for each  $a \in \mathcal{B}$ .  $I$  is a *strict subset* of  $J$  ( $I \subset J$ ) if  $I \subseteq J$  and  $I \neq J$ . Fuzzy set intersection ( $I \cap J$ ), union ( $I \cup J$ ), and difference ( $I \setminus J$ ) are defined as follows: for every  $a \in \mathcal{B}$ ,  $[I \cap J](a) := \min\{I(a), J(a)\}$ ,  $[I \cup J](a) := \max\{I(a), J(a)\}$ , and  $[I \setminus J](a) := \max\{I(a) - J(a), 0\}$ . An interpretation  $I$  *models* a rule  $r$  with head  $a$ , denoted  $I \models r$ , if  $I(a) \geq I(B(r))$ .  $I$  models a FASP program  $P$  ( $I \models P$ ) if  $I \models r$  for each  $r \in P$ . An interpretation  $M$  is an answer set of a positive FASP program  $P$  if  $M$  is a minimal model of  $P$ , i.e.,  $M \models P$  and there is no interpretation  $I \subset M$  such that  $I \models P$ .  $M$  is an answer set of a normal FASP program  $P$  if  $M$  is an answer set of the reduct  $P^M$  obtained from  $P$  by replacing each negative literal *not*  $b$  by the constant  $\bar{I}(\text{not } b)$ .

*Example 1*

The FASP program  $P_{\text{ex1}} = \{a \leftarrow c \otimes \text{not } b, b \leftarrow \text{not } c, c \leftarrow \overline{0.1}\}$  is a stratified program, containing one positive rule. Consider the interpretation  $I$  such that  $I(a) = 0, I(b) = 0.9$ , and  $I(c) = 0.1$ . The reduct  $P_{\text{ex1}}^I$  is  $\{a \leftarrow c \otimes \overline{0.1}, b \leftarrow \overline{0.9}, c \leftarrow \overline{0.1}\}$ . Under the Lukasiewicz t-norm semantics,  $I$  is a minimal model of  $P_{\text{ex1}}^I$ , and hence it is an answer set for  $P_{\text{ex1}}$ . It can be seen that this is in fact the only answer set of  $P_{\text{ex1}}$ .

In general, a FASP program may have infinitely many answer sets. Rather than trying to enumerate them all, we are interested in approximating them by removing interpretations that cannot be answer sets.

### 3 Search Space Pruning Operators

To approximate the answer sets of a normal FASP program  $P$ , we will construct two interpretations  $L, U \in \mathcal{I}$ , called *lower* and *upper bound*, respectively, such that  $L \subseteq U$ . The idea is to find the tightest interpretations  $L$  and  $U$  such that  $L \subseteq M \subseteq U$  holds for every answer set  $M$  of  $P$ . Let  $\mathbf{0}$  and  $\mathbf{1}$  be the interpretations that map every propositional atom to 0 and 1, respectively. Obviously,  $L = \mathbf{0}$  and  $U = \mathbf{1}$  satisfy the desired condition of bounding all answer sets for any program  $P$ . Before defining operators for improving these bounds, we introduce a convenient notion for the *partial evaluation* of a rule  $r$  w.r.t.  $\langle I, J \rangle$ , where  $I, J \in \mathcal{I}$  are fuzzy interpretations:

$$\langle I, J \rangle(r) := I(B^+(r)) \otimes J(\text{not}B^-(r)). \quad (2)$$

Lower bounds can be improved by the well-known immediate consequence operator.

*Definition 1 (Immediate consequence operator)*

The *immediate consequence operator* of a program  $P$  w.r.t. an upper bound  $U$  is the function  $T_P^U : \mathcal{I} \rightarrow \mathcal{I}$  where  $[T_P^U(L)](a) := \max\{\langle L, U \rangle(r) \mid r \in P, H(r) = a\}$  for each atom  $a \in \mathcal{B}$ .

The operator  $T_P^U$  is monotonic, and thus it has a least fixpoint  $T_P^U \uparrow \mathbf{0}$ , i.e., the sequence  $L_0 := \mathbf{0}, L_{i+1} := T_P^U(L_i)$  ( $i \geq 0$ ) converges to  $T_P^U \uparrow \mathbf{0}$ .

*Theorem 1*

Let  $U \in \mathcal{I}$ , and  $P$  be a program. The fixpoint  $T_P^U \uparrow \mathbf{0}$  is reached after a linear number of iterations, measured on the number of atoms appearing in  $P$ .

If  $P$  is positive,  $T_P^1 \uparrow \mathbf{0}$  coincides with the unique answer set of  $P$  (Lukasiewicz 2006). Hence, by Theorem 1, this answer set is computable in polynomial time. For normal programs, instead, the lower bound can be improved replacing  $L$  by  $T_P^U \uparrow \mathbf{0}$ . To improve the upper bound, we use an idea from classical ASP: the truth of an atom  $a$  in an answer set  $M$  must be supported by some rule, i.e.,  $M(a)$  cannot be larger than the maximum of  $M(B(r))$  over all rules  $r \in P$  with  $H(r) = a$ . Support must also be acyclic, or *founded*.

*Definition 2 (Fuzzy unfounded set)*

Let  $L, U \in \mathcal{I}$ ,  $L \subseteq U$ , and  $P$  be a program. A fuzzy set  $X \in \mathcal{I}$  is a *fuzzy unfounded set* for  $P$  w.r.t.  $(L, U)$  if for each  $r \in P$  such that  $X(H(r)) > 0$ , the following inequality is satisfied:  $[U \cap (\mathbf{1} \setminus X)](H(r)) \geq \langle U \cap (\mathbf{1} \setminus X), L \rangle(r)$ .

Intuitively, fuzzy unfounded sets evidence lack of (acyclic) support.

*Theorem 2*

For FASP programs without numeric constants and crisp sets, Definition 2 coincides with the original notion of unfounded set by Van Gelder et al. (1991).

As in the crisp case, the union of two (fuzzy) unfounded sets is an unfounded set.

*Theorem 3*

Let  $X_1, X_2$  be two fuzzy unfounded sets for  $P$  w.r.t.  $(L, U)$ . Then also  $X_1 \cup X_2$  is a fuzzy unfounded set for  $P$  w.r.t.  $(L, U)$ .

We can thus define the greatest fuzzy unfounded set, denoted  $GUS_P^{L,U}$ , as the union of all fuzzy unfounded sets. We also highlight a relationship with fuzzy answer sets.

*Theorem 4*

$M$  is a fuzzy answer set of a program  $P$  if and only if  $GUS_P^{M,M} = \mathbf{1} \setminus M$ .

In order to find unfounded sets, we can employ the operator  $R_P$ , defined next.

*Definition 3 (Operator  $R_P$ )*

Let  $L, U \in \mathcal{I}$ . The operator  $R_P$  for a program  $P$  w.r.t.  $(L, U)$  is the function  $R_P^{L,U} : \mathcal{I} \rightarrow \mathcal{I}$  such that  $[R_P^{L,U}(X)](a) := \min\{X(a), 1 - \max\{\langle U \cap (\mathbf{1} \setminus X), L \rangle(r) \mid r \in P, H(r) = a\}\}$  for every  $a \in \mathcal{B}$ .

The operator  $R_P^{L,U}$  is antitonic; thus it has a greatest fixpoint  $R_P^{L,U} \Downarrow \mathbf{1}$ , which is the limit of the sequence  $X_0 := \mathbf{1}, X_{i+1} := R_P^{L,U}(X_i), i \geq 0$ . There is a strong relationship between fuzzy unfounded sets and the operator  $R_P$ , which allows for replacing the upper bound  $U$  by  $\mathbf{1} \setminus (R_P^{L,U} \Downarrow \mathbf{1})$ .

*Theorem 5*

Let  $L, U \in \mathcal{I}, L \subseteq U$ , and  $P$  be a program. If  $\mathbf{1} \setminus R_P^{L,U} \Downarrow \mathbf{1} \subseteq U$ , then  $R_P^{L,U} \Downarrow \mathbf{1} = GUS_P^{L,U}$ .

It is easy to see that one application of the  $R_P$  operator requires linear time in the number of rules  $P$ . Moreover, as in Theorem 1, the greatest fixpoint  $R_P^{L,U} \Downarrow \mathbf{1}$  is obtained after at most as many applications of this operator as there are atoms in  $P$ . In total, this means that this fixpoint can be computed in polynomial time on the size of  $P$ .

*Theorem 6*

Let  $L, U \in \mathcal{I}$  and  $P$  be a program. The fixpoint  $R_P^{L,U} \Downarrow \mathbf{1}$  can be computed in polynomial time on the size of  $P$ .

To sum up, the  $T_P$  operator can be used to improve the lower bound  $L$ , while  $R_P$  is useful for decreasing the upper bound  $U$ . These two operators complement each other, as a tighter lower bound may help to further decrease the upper bound, and dually, a tighter upper bound may increase the lower bound. In fact, if  $L \subseteq L'$  and  $U' \subseteq U$ , then (i)  $T_P^U(I) \subseteq T_P^{U'}(I)$  and (ii)  $R_P^{L',U}(I) \subseteq R_P^{L,U}(I)$ , thus  $\mathbf{1} \setminus R_P^{L,U}(I) \subseteq \mathbf{1} \setminus R_P^{L',U}(I)$  hold for every interpretation  $I$ . We can then combine  $T_P$  and  $R_P$  to obtain a new operator.

*Definition 4 (Well-founded operator)*

The well-founded operator is the function  $W_P : \mathcal{I} \times \mathcal{I} \rightarrow \mathcal{I} \times \mathcal{I}$  defined as

$$W_P(L, U) := \left( T_P^U(L), \mathbf{1} \setminus R_P^{L,U} \Downarrow \mathbf{1} \right). \quad (3)$$

The well-founded operator is monotonic in the lattice  $(\mathcal{I} \times \mathcal{I}, \leq)$ , where  $(L, U) \leq (L', U')$  if and only if  $L \subseteq L'$  and  $U' \subseteq U$ . Hence,  $W_P$  has a least fixpoint  $W_P \Uparrow (\mathbf{0}, \mathbf{1})$ . Moreover, every pair  $W_i = (L_i, U_i)$  in the sequence  $W_0 := (\mathbf{0}, \mathbf{1}), W_{i+1} := W_P(W_i)$  ( $i \geq 0$ ) satisfies that  $L_i \subseteq M \subseteq U_i$  for every answer set  $M$  of  $P$ . We show a stronger result.

*Theorem 7*

Let  $P$  be a program,  $L, U$  two interpretations,  $(L', U') = W_P(L, U)$ , and  $M$  an answer set for  $P$ . If  $L \subseteq M \subseteq U$ , then  $L' \subseteq M \subseteq U'$ .

Just as  $T_P$  does for positive programs, an iterative application of the well-founded operator yields the unique answer set of stratified programs in polynomial time.

*Theorem 8*

Let  $P$  be a stratified program. The least fixpoint of  $W_P$  coincides with the unique answer set of  $P$  and is computable in polynomial time.

*Example 2*

Consider the stratified program  $P_{\text{ex1}}$  from Example 1. The application of the well-founded operator to  $P_{\text{ex1}}$  are shown in Table 1. After three iterations, a fixpoint is reached, stating that every answer set  $M$  for  $P_{\text{ex1}}$  must be such that  $M(a) = 0$ ,  $M(b) = 0.9$ , and  $M(c) = 0.1$ . As seen before, this is indeed the only answer set of  $P_{\text{ex1}}$ .

Although it is only possible to guarantee that the iterative application of the combined operator computes an answer set if the program is stratified, they can also produce the answer sets of cyclic programs, as shown by the following example.

*Example 3*

For the program  $P_{\text{ex2}} = P_{\text{ex1}} \cup \{b \leftarrow \text{not } a\}$ , Table 1 shows the applications of  $W_P$ . A fixpoint is reached after four iterations, stating that the only candidate for an answer set is the interpretation  $M$  with  $M(a) = 0$ ,  $M(b) = 1$ , and  $M(c) = 0.1$ . The reduct of  $P_{\text{ex2}}$  w.r.t.  $M$  is  $\{a \leftarrow c \otimes \bar{0}, b \leftarrow \bar{0.9}, c \leftarrow \bar{0.1}, b \leftarrow \bar{1}\}$ , for which  $M$  is a minimal model. However, the iterative application of  $W_P$  might not terminate, as for example for program  $P_{\text{inf}} = \{a \leftarrow \bar{0.9} \otimes \text{not } a\}$  over the product t-norm. The least fixpoint of  $W_{P_{\text{inf}}}$  assigns to  $a$  degree  $9/19$ , but  $\omega$  applications are required.

Even if  $W_P$  often provides good bounds, there is still room for improvement. Consider the program  $P_{\text{odd}} = \{a \leftarrow \text{not } a\}$ .  $P_{\text{odd}}$  has exactly one answer set  $M$  with  $M(a) = 1/2$ . However,  $W_P$  yields the bounds  $L = \mathbf{0}$  and  $U = \mathbf{1}$ . Observe that the rule of  $P_{\text{odd}}$  states the implicit restriction that  $a$  must be evaluated to at least  $1/2$  in every model  $I$  of  $P$  because  $I \models P$  implies  $I(a) \geq I(\text{not } a) = 1 - I(a)$ . Implicit restrictions of this kind might be used to further improve lower bounds.

*Definition 5 (Minimal satisfiability)*

The *minimal satisfiability operator* of a program  $P$  w.r.t. an upper bound  $U$  is the function  $S_P^U : \mathcal{I} \rightarrow \mathcal{I}$  where  $[S_P^U(L)](a) := \inf\{I(a) \mid I \models P, L \subseteq I \subseteq U\}$  for each  $a \in \mathcal{B}$ .

Table 1. Iterations of the combined operator on  $P_{\text{ex1}}$  and  $P_{\text{ex2}}$ 

	$P_{\text{ex1}}$			$P_{\text{ex2}}$		
	a	b	c	a	b	c
$L_0 := \mathbf{0} \quad ; U_0 := \mathbf{1}$	0; 1	0 ; 1	0 ; 1	0 ; 1	0 ; 1	0 ; 1
$L_1 := T_P^{U_0}(L_0); U_1 := \mathbf{1} \setminus R_P^{L_0, U_0} \downarrow \mathbf{1}$	0; 0.1	0 ; 1	0.1; 0.1	0 ; 0.1	0 ; 1	0.1; 0.1
$L_2 := T_P^{U_1}(L_1); U_2 := \mathbf{1} \setminus R_P^{L_1, U_1} \downarrow \mathbf{1}$	0; 0.1	0.9; 0.9	0.1; 0.1	0 ; 0.1	0.9; 1	0.1; 0.1
$L_3 := T_P^{U_2}(L_2); U_3 := \mathbf{1} \setminus R_P^{L_2, U_2} \downarrow \mathbf{1}$	0; 0	0.9; 0.9	0.1; 0.1	0 ; 0	0.9; 1	0.1; 0.1
$L_4 := T_P^{U_3}(L_3); U_4 := \mathbf{1} \setminus R_P^{L_3, U_3} \downarrow \mathbf{1}$	0; 0	0.9; 0.9	0.1; 0.1	0 ; 0	1 ; 1	0.1; 0.1
$L_5 := T_P^{U_4}(L_4); U_5 := \mathbf{1} \setminus R_P^{L_4, U_4} \downarrow \mathbf{1}$	0; 0	0.9; 0.9	0.1; 0.1	0 ; 0	1 ; 1	0.1; 0.1

*Theorem 9*

Let  $L, U, M \in \mathcal{I}$  and  $P$  be a program. If  $M \models P$  and  $L \subseteq M \subseteq U$ , then  $S_P^U(L) \subseteq M \subseteq U$ .

It is thus possible to improve the bounds by an iterative application of the  $S_P$  and  $R_P$  operators. It is also easy to see that  $T_P^U(L) \subseteq S_P^U(L)$ . This in particular means that the lower bound obtained by  $S_P$  is always at least as good as the one given by  $T_P$ , and in some cases strictly better. This translates not only in better bounds being computed, but also in a lower number of iterations needed to obtain them. Unfortunately, in general it is not clear how to compute the minimal satisfiability operator, as it requires finding optimal values for a possibly complex system of constraints, depending on the t-norm. If we restrict to the Łukasiewicz t-norm, then  $S_P$  reduces to solving a series of linear programming problems. More precisely, for a program  $P$  we define a finite system of inequalities  $L_P$  having, for each rule  $r \in P$  of the form (1), one inequality

$$a \geq b_1 + \dots + b_m - b_{m+1} - \dots - b_n + 1 - m \quad (4)$$

such that all variables  $a, b_1, \dots, b_n$  are restricted to the interval  $[0, 1]$ . All models of  $P$  must satisfy  $L_P$  and *vice versa*. Thus, we obtain the following result.

*Theorem 10*

Let  $P$  be a program over the Łukasiewicz t-norm, and  $L, U \in \mathcal{I}$ . For every atom  $a \in \mathcal{B}$  it holds that  $[S_P^U(L)](a) = \min\{I(a) \mid I \text{ satisfies } L_P \cup \{L(b) \leq I(b) \leq U(b) \mid b \in \mathcal{B}\}\}$ .

In this case, the minimal satisfiability operator can be computed efficiently.

*Theorem 11*

Let  $L, U \in \mathcal{I}$ , and  $P$  be a program over the Łukasiewicz t-norm.  $S_P^U(L)$  is computable in polynomial time w.r.t. the number of rules.

Consider again program  $P_{odd}$ .  $S_P$  computes the minimal value for  $a$  with  $a \geq 1 - a$ , i.e.,  $1/2$ . The lower bound is updated to  $1/2$ . Then,  $R_P$  yields  $R_P(L) = 1/2$ , and updates the upper bound to  $1/2$ . Further applications of the operators do not modify  $L$  or  $U$ , hence  $M(a) = 1/2$  is our candidate answer set, which in this case is the correct solution.

## 4 Implementation and Experiment

We developed **fasp**, a prototype handling propositional FASP programs. Programs with variables can be transformed into equivalent propositional FASP programs by means of an almost standard grounding procedure, for example by using **gringo** (Gebser et al. 2007), which we extended to deal with numeric constants. The prototype is available at <https://github.com/alviano/fasp.git>. In the input program, numeric constants are specified by writing a decimal or fractional number preceded by a **#** character. The output of **gringo** is a numeric format which constitutes the input of **fasp**. The output of **fasp** is the fuzzy answer sets approximation obtained by the operators described in Section 3 w.r.t. the t-norm specified by the command-line option `--tnorm=TNORM`, where the currently implemented TNORMs are **lukasiewicz** (default), **godel**, and **product**. For the Łukasiewicz t-norm, **fasp** can also produce the bilevel program defined by Blondeel et al. (2012), which is encoded for the library YALMIP (<http://users.isy.liu.se/johanl/yalmip>) and can be solved by invoking **octave** (<http://www.gnu.org/software/octave/>). The

---

**Algorithm 1:** Fuzzy answer sets approximation

---

**Input** : a FASP program  $P$   
**Output:** lower and upper bounds  $L, U$

```

1 begin
2    $L := \mathbf{0}; \quad U := \mathbf{0};$ 
3   foreach  $r \in P$  do
4     UpperBoundIncrease( $r$ );
5   foreach constant  $\bar{c}$  occurring in  $P$  do
6      $L(\bar{c}) := c; \quad U(\bar{c}) := c;$ 
7     foreach  $r \in P$  such that  $\bar{c} \in B^+(r)$  do
8       UpperBoundIncrease( $r$ );
9       LowerBoundIncrease( $H(r), L(B^+(r)) \otimes U(\text{not } B^-(r))$ );
10  MinimalSatisfiability();
11  return ( $L, U$ )

```

---

**Procedure** UpperBoundIncrease( $r$ : rule)

---

```

1 begin
2   if  $U(B^+(r)) \otimes L(\text{not } B^-(r)) > U(H(r))$  then
3      $U(H(r)) := U(B^+(r)) \otimes L(\text{not } B^-(r));$ 
4      $sp(H(r)) := r;$ 
5     foreach  $r' \in P$  such that  $H(r) \in B^+(r')$  do
6       UpperBoundIncrease( $r'$ );

```

---

bilevel program is produced after computing the approximating operators if `fasp` is run with `--mode=answer-set`, while `--mode=answer-set-unoptimized` can be used for producing the bilevel program without applying any operator. Hence, if the encoding of program  $P_{\text{ex2}}$ :

```
a :- c, #0.1.    b :- #0.9.    c :- #0.1.    b :- not a.
```

is written in a file `test.lp`, the fuzzy answer set `a[0], b[1], c[0.1]` is the output of

```
gringo test.lp | fasp --tnorm=lukasiewicz --mode=answer-set
```

The prototype implements Algorithm 1, where for simplicity we assume that all empty rule bodies are replaced by constant  $\bar{1}$ . Initially,  $L$  and  $U$  are set to  $\mathbf{0}$ . Any numeric constant  $\bar{c}$  is treated as a propositional atom whose lower and upper bounds are set to  $c$  (line 6). These bounds are propagated as described below (lines 5–9), but first `fasp` computes suitable upper bounds for all atoms (line 4). One could argue that processing constants before other rules could be more reasonable; however, Algorithm 1 runs on the stream provided by `gringo`, which first outputs rules where fuzzy atoms are represented by ids, and only at the end are these ids associated to atom names and numeric constants.

Upper bounds are determined by the fixpoint of  $R_P$ . To achieve an efficient implementation of this operator, `fasp` takes advantage of source pointers (Simons et al. 2002), a technique largely used in crisp ASP solvers that we adapted to the fuzzy case. A source pointer for an atom  $a$  is a rule witnessing the upper bound of  $a$ . Numeric constants do not need source pointers, while propositional atoms do. Initially, all source pointers are unset. Procedure `UpperBoundIncrease` is invoked for each rule  $r \in P$  (lines 3–4 of Algorithm 1).

---

**Procedure LowerBoundIncrease( $a$ : atom,  $d$ : degree)**


---

```

1 begin
2   if  $d > L(a)$  then
3      $L(a) := d$ ;
4     foreach  $r \in P$  such that  $a \in B^+(r)$  do
5       LowerBoundIncrease( $H(r)$ ,  $L(B^+(r)) \otimes U(not B^-(r))$ );
6     foreach  $r \in P$  such that  $a \in B^-(r)$  do
7       UpperBoundDecrease( $r$ );

```

---



---

**Procedure UpperBoundDecrease( $r$ : rule)**


---

```

1 begin
2   if  $sp(H(r)) = r$  and  $U(H(r)) - U(B^+(r)) \otimes L(not B^-(r)) > \epsilon$  then
3      $U(H(r)) := U(B^+(r)) \otimes L(not B^-(r))$ ;
4     foreach  $r' \in P$  such that  $H(r) \in B^+(r')$  do
5       UpperBoundDecrease( $r'$ );
6     foreach  $r' \in P$  such that  $H(r) \in B^-(r')$  do
7       LowerBoundIncrease( $H(r')$ ,  $L(B^+(r')) \otimes U(not B^-(r'))$ );
8      $r' := \arg \min_{r'' \in P} U(B^+(r'')) \otimes L(not B^-(r''))$ ;
9     UpperBoundIncrease( $r'$ );

```

---

The procedure computes the upper bound for  $B(r)$  as  $d = U(B^+(r)) \otimes L(not B^-(r))$ . If  $d$  is strictly greater than  $U(H(r))$ , the upper bound of  $H(r)$  is set to  $d$  and the source pointer of  $H(r)$  is set to  $r$  (lines 2–4). This new upper bound for  $H(r)$  is propagated in each rule  $r'$  in which  $H(r)$  occurs as a positive body literal (lines 5–6), possibly increasing the upper bound of  $H(r')$  and changing its source pointer to  $r'$ . At the end of this process, atoms having upper bound different from 0 have source pointers set. Lower bounds, instead, are given by the fixpoint of  $T_P$ , obtained by first processing facts and then numeric constants. Each new lower bound  $d$ , say for atom  $a$ , can increase the lower bound of the head atom of any rule  $r$  in which  $a$  occurs as a positive body literal (lines 2–5 of LowerBoundIncrease). More specifically,  $H(r)$  has a new lower bound set to  $L(B^+(r)) \otimes U(not B^-(r))$  if this degree is strictly greater than  $L(H(r))$ .

Lower and upper bound updates can interact intensively to obtain better bounds, and our system handles these interactions as soon as possible. Whenever the lower bound of an atom  $a$  is increased, the system checks whether decreasing the upper bound of the head atom of any rule  $r$  in which  $a$  occurs as a negative literal is possible (lines 6–7 of LowerBoundIncrease). In particular, this might happen if  $r$  is the source pointer of  $H(r)$ , in which case the upper bound of  $H(r)$  might be decreased to  $U(B^+(r)) \otimes L(not B^-(r))$ . (To ensure termination, the upper bound is updated only if the decrease is greater than a fixed constant  $\epsilon$ .) This propagation is handled by procedure UpperBoundDecrease, which first checks whether other upper bounds have to be decreased (lines 4–5). To this end, only rules in which  $H(r)$  occurs as a positive body literal have to be checked, and source pointers allow to skip most of these rules. Once upper bounds have been decreased, the procedure possibly increases the lower bounds of the head atom of any rule  $r'$  in which

---

```

Procedure MinimalSatisfiability
1 begin
2   while bounds changed do
3      $D := \emptyset$ ;
4     foreach  $a \in \mathcal{B}$  do
5       if  $a \notin D$  then
6          $s :=$  solution for linear program for  $a$ ;
7         LowerBoundIncrease( $a, s(a)$ );
8          $D := D \cup \{b \mid s(b) = L(b)\}$ ;

```

---

$H(r)$  occurs as a negative body literal (lines 6–7). Finally the procedure determines the best source pointer for  $H(r)$  (line 8) and in case increases its upper bound, propagating this information by means of the procedure UpperBoundIncrease (line 9).

When these procedures terminate, all atoms have proper lower and upper bounds. For the Lukasiewicz t-norm, bounds can be further improved by the minimal satisfiability operator  $S_P$  implemented by procedure MinimalSatisfiability, which takes advantage of the GLPK library (<http://www.gnu.org/software/glpk/>) for solving linear programs. However, operator  $S_P$  could be resource demanding, and thus `fasp` limits its computation as follows: The operator is computed w.r.t. an atom  $a$ , thus obtaining a proper lower bound for  $a$  which is possibly propagated (lines 6–7). The procedure then skips all atoms for which the solution provided by the simplex algorithm already coincides with their lower bounds and thus witnesses that no improvement is possible for these bounds (line 8). Moreover, as the system of inequalities is almost fixed for the input program, it is computed after reading the program and updated when bounds are changed.

Table 2 reports the result of an experiment on `fasp`. *Graph Coloring* and *Hamiltonian Path* are variants of well-known NP-complete problems. Instances of Graph Coloring are those used in the third ASP Competition (Calimeri et al. 2011), while random instances were tested for Hamiltonian Path. For both domains, random constants were added in the body of facts. *Stratified* and *Odd Cycle* are very simple programs consisting of rules  $a_{i+1} \leftarrow a_i$ ,  $i = 0, \dots, n - 1$ , where  $n$  is given by the test cases, and atom  $a_0$  is defined by  $a_0 \leftarrow 0.9$  and by  $a_0 \leftarrow \text{not } a_n$ , respectively. The experiment was performed on an Intel Xeon CPU X5365 3.00 GHz with 4 GB of central memory and running Debian 6 with kernel Linux 2.6.32. Memory was limited to 3 GB and execution time to 600 seconds. In this benchmark there is a sensible performance gain due to the approximating operators implemented in `fasp`. In fact, “unoptimized” bilevel programs showed a poor

Table 2. *Experimental result on fasps: solved instances and average execution time*

	Tested instances	Timeouts		Average execution time*		Average perc. gain*
		Unopt.	Optimized	Unopt.	Optimized	
Graph Coloring	60	34	0	247.44	34.45 (2.68)	76.43%
Hamiltonian Path	40	33	9	120.51	6.41 (0.02)	81.49%
Stratified	90	10	0	190.07	1.80 (0.02)	96.71%
Odd Cycle	90	33	0	186.94	1.95 (0.03)	97.18%

\* Computed on the instances solved by both the approaches.

performance, timing out 110 times, while the “optimized” approach timed out only 9 times (on which also the “unoptimized” approach timed out). Even restricting to the 170 test cases solved by the unoptimized approach, there is a significant advantage of the optimized approach, evidenced by an average percentage gain of at least 76%. Note that in Table 2 the time required for computing the approximation operators is reported in parentheses and included in the execution time of the optimized approach.

## 5 Related Work

The study of fuzzy extensions of logic programs can be traced back more than two decades (see e.g. Dubois et al. 1991). Moreover, compared to classical logic, fuzzy logics offer several additional levels of liberty for the definition of their semantics; namely, the choice of the space of truth degrees, the interpretation of the conjunction, the negation, and even the implication. Hence, we describe only the work that is closest related to ours.

The first generalization of the immediate consequence operator to deal with fuzzy semantics was due to Achs and Kiss (1995) and Achs (1997), albeit exclusively for the Gödel t-norm semantics. Fuzzy answer set semantics were introduced by Lukasiewicz (2006) based on a generalization of the Gelfond-Lifschitz transformation (Damásio and Pereira 2001). It was then shown that positive and stratified programs have a unique answer set, and that it can be obtained by a finite iteration of the  $T_P$  operator. However, no further analysis on the number of iterations needed to obtain that answer set was made. It is worth noting that the semantics described by Lukasiewicz are based on a finite set of truth values, rather than the whole interval  $[0, 1]$ , as in our case. Nonetheless, the proof ideas can be generalized to arbitrary t-norms over  $[0, 1]$  without difficulty. General fuzzy answer set programs have been studied in detail in the last years (Janssen 2011), considering not only general t-norm semantics, but also arbitrary connectives to be used in the head and the body of the rules.

While the complexity of finding fuzzy answer sets is now relatively well understood, there were to-date no solvers available. In an effort to compute answer sets for general programs, a completion method was proposed by Janssen et al. (2012). The idea is to transform the program  $P$  into a set of fuzzy logic formulas, whose models correspond precisely to the answer sets of  $P$ . However, due to the lack of (optimized) fuzzy logic reasoners, this reduction does not allow for an effective implementation. A different approach, specifically designed for the Lukasiewicz t-norm, is to reduce the program  $P$  to a bilevel linear programming problem (Blondeel et al. 2012). This method, which is also implemented by our system, has important theoretical repercussions, e.g., it can be used to show that disjunctions on rule heads do not add expressivity under this semantics.

Unfounded sets for FASP programs were first defined by Nieuwenborgh et al. (2007a); these unfounded sets are defined w.r.t. total interpretations and are actually crisp sets used for characterizing fuzzy answer sets. We will refer to this notion as *crisp unfounded sets*. Our definition is more general: it is given w.r.t. partial interpretations and for fuzzy sets; it is suitable for pruning the search space but also characterizes answer sets. A relationship between the two notions follows by Theorem 4.

### Corollary 1

For every fuzzy unfounded set  $X$  w.r.t.  $(I, I)$ , set  $\{a \mid X(a) + I(a) > 1\}$  is a crisp

unfounded set. For every crisp unfounded set  $Y$  there is a fuzzy unfounded set  $X$  such that  $Y = \{a \mid X(a) + I(a) > 1\}$ .

To the best of our knowledge, there had been no previous attempts to generalize the  $R_P$  operator as a complementation of unfounded sets. This operator allows for a better approximation of answer sets without spending too many resources. A similar idea was studied by Loyer and Straccia (2009), where a well-founded semantics is used for querying fuzzy logic programs over the Gödel t-norm. A well-founded semantics was also defined by Damásio and Pereira (2001), for which we can prove the following result.

*Theorem 12*

The fixpoint of  $W_P$  gives the well-founded semantics by Damásio and Pereira (2001).

## 6 Conclusions

We studied the problem of finding answer sets for normal FASP programs with t-norm based semantics. We studied fuzzy variants of the operators  $T_P$  and  $R_P$ , which bound the class of all answer sets of a FASP program  $P$ . These operators, as well as the combined well-founded operator  $W_P$ , extend well-known operators from classical ASP to handle fuzzy semantics. As such, our operators preserve many of the properties that make them suitable for practical implementations. In particular, we have shown that one application of  $W_P$  requires only polynomial time, measured on the size of  $P$ . Moreover, for positive and stratified programs, an iterative application of this operator yields the unique answer set in polynomial time, independently of the t-norm used. For normal FASP programs, which may have none or infinitely many answer sets, this operator is only guaranteed to provide lower and upper approximations for the class of answer sets. Depending on the program and the t-norm used, better bounds can be achieved by combining  $R_P$  with a new operator  $S_P$ . In particular,  $S_P$  is computable in polynomial time for the Łukasiewicz t-norm by solving at most one linear program for each atoms in the program.

We implemented a prototype which applies the operators in an optimized manner, using  $S_P$  only when no information can be obtained from  $W_P$ . In particular, if the program is stratified, then  $S_P$  will never be triggered. The system also keeps track of previous solutions of the set of inequalities introduced by  $S_P$ , to avoid trying to optimize atoms whose current bounds are already known to be optimal. It also takes advantage of other optimizations developed for classical ASP, such as source pointers, to reduce the number of computations needed. The approximation provided by our prototype could aid in the computation of fuzzy answer sets, as evidenced by our experiment.

There are several lines for future work. From the theoretical point of view, we plan to investigate further conditions and operators that allow a precise computation of answer sets. In particular, as finding answer sets for normal FASP programs is NP-hard, we need to develop methods for efficiently dealing with a choice operator. We believe that the completion approach by Janssen et al. (2012) can be improved through the introduction of binary selection variables. From the practical point of view, we intend to improve the prototype, which currently relies on an external tool for computing answer sets when the bounds cannot be further improved. One idea for this point would be to implement a completion-based method extended with learning techniques.

## References

- ACHS, Á. 1997. Evaluation strategies of fuzzy datalog. *Acta Cybernetica* 13, 1, 85–102.
- ACHS, Á. AND KISS, A. 1995. Fuzzy extension of datalog. *Acta Cybernetica* 12, 2, 153–166.
- ALVIANO, M., FABER, W., LEONE, N., PERRI, S., PFEIFER, G., AND TERRACINA, G. 2011. The disjunctive datalog system DLV. In *Datalog 2.0*, G. Gottlob, Ed. Vol. 6702. Springer Berlin/Heidelberg, 282–301.
- BARAL, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.
- BLONDEEL, M., SCHOCKAERT, S., DE COCK, M., AND VERMEIR, D. 2012. NP-completeness of fuzzy answer set programming under Lukasiewicz semantics. In *Working papers of the ECAI-2012 workshop in weighted logics for artificial intelligence WL4AI*, L. Godo and H. Prade, Eds. 43–50.
- CALIMERI, F., IANNI, G., RICCA, F., ALVIANO, M., BRIA, A., CATALANO, G., COZZA, S., FABER, W., FEBBRARO, O., LEONE, N., MANNA, M., MARTELLO, A., PANETTA, C., PERRI, S., REALE, K., SANTORO, M. C., SIRIANNI, M., TERRACINA, G., AND VELTRI, P. 2011. The third answer set programming competition: Preliminary report of the system competition track. In *11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2011)*, J. Delgrande and W. Faber, Eds. Lecture Notes in Computer Science, vol. 6645. Springer Berlin/Heidelberg, 388–403.
- DAMÁSIO, C. V. AND PEREIRA, L. M. 2001. Antitonic logic programs. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning*. LPNMR'01. Springer-Verlag, London, UK, UK, 379–392.
- DELGRANDE, J. P., SCHAUB, T., TOMPITS, H., AND WOLTRAN, S. 2008. Belief revision of logic programs under answer set semantics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, G. Brewka and J. Lang, Eds. 411–421.
- DUBOIS, D., LANG, J., AND PRADE, H. 1991. Fuzzy sets in approximate reasoning, part 2: Logical approaches. *Fuzzy Sets and Systems* 40, 1, 203–244.
- GEBSER, M., KAUFMANN, B., NEUMANN, A., AND SCHAUB, T. 2007. Conflict-driven answer set solving. In *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*. Morgan Kaufmann Publishers, 386–392.
- GEBSER, M., SCHAUB, T., AND THIELE, S. 2007. Gringo : A new grounder for answer set programming. In *Logic Programming and Nonmonotonic Reasoning — 9th International Conference, LPNMR'07*, C. Baral, G. Brewka, and J. Schlipf, Eds. Lecture Notes in Computer Science, vol. 4483. Springer Verlag, Tempe, Arizona, 266–271.
- GELFOND, M. AND LIFSCHITZ, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing* 9, 365–385.
- JANSSEN, J. 2011. Foundations of fuzzy answer set programming. Ph.D. thesis, Ghent University.
- JANSSEN, J., VERMEIR, D., SCHOCKAERT, S., AND COCK, M. D. 2012. Reducing fuzzy answer set programming to model finding in fuzzy logics. *Theory and Practice of Logic Programming* 12, 6, 811–842.
- KLEMENT, E. P., MESIAR, R., AND PAP, E. 2000. *Triangular Norms*. Trends in Logic, Studia Logica Library. Springer-Verlag.
- LIERLER, Y. AND MARATEA, M. 2004. Cmodels-2: SAT-based Answer Set Solver Enhanced to Non-tight Programs. In *Proceedings of the 7th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR-7)*, V. Lifschitz and I. Niemelä, Eds. LNAI, vol. 2923. Springer, 346–350.
- LIFSCHITZ, V. 2002. Answer Set Programming and Plan Generation. *Artificial Intelligence* 138, 39–54.
- LIN, F. AND YOU, J.-H. 2002. Abduction in logic programming: A new definition and an abductive procedure based on rewriting. *Artificial Intelligence* 140, 1/2, 175–205.

- LOYER, Y. AND STRACCIA, U. 2009. Approximate well-founded semantics, query answering and generalized normal logic programs over lattices. *Annals Mathematics and Artificial Intelligence* 55, 3-4, 389–417.
- LUKASIEWICZ, T. 2006. Fuzzy description logic programs under the answer set semantics for the semantic web. In *Proc. 2nd Int. Conf. on Rules and Rule Markup Languages for the Semantic Web (RuleML 2006)*, T. Eiter, E. Franconi, R. Hodgson, and S. Stephens, Eds. IEEE Computer Society, 89–96.
- MAREK, V. W. AND REMMEL, J. B. 2004. Answer set programming with default logic. In *10th International Workshop on Non-Monotonic Reasoning (NMR 2004), Whistler, Canada, June 6-8, 2004, Proceedings*, J. P. Delgrande and T. Schaub, Eds. 276–284.
- MAREK, V. W. AND TRUSZCZYŃSKI, M. 1999. Stable Models and an Alternative Logic Programming Paradigm. In *The Logic Programming Paradigm – A 25-Year Perspective*, K. R. Apt, V. W. Marek, M. Truszczyński, and D. S. Warren, Eds. Springer Verlag, 375–398.
- NIEMELÄ, I. 1999. Logic Programming with Stable Model Semantics as Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence* 25, 3–4, 241–273.
- NIEUWENBORGH, D. V., COCK, M. D., AND VERMEIR, D. 2007a. Computing fuzzy answer sets using dlhex. In *Logic Programming, 23rd International Conference, ICLP 2007, Porto, Portugal, September 8-13, 2007, Proceedings*. Lecture Notes in Computer Science, vol. 4670. 449–450.
- NIEUWENBORGH, D. V., COCK, M. D., AND VERMEIR, D. 2007b. An introduction to fuzzy answer set programming. *Annals Mathematics and Artificial Intelligence* 50, 3-4, 363–388.
- SIMONS, P., NIEMELÄ, I., AND SOININEN, T. 2002. Extending and Implementing the Stable Model Semantics. *Artificial Intelligence* 138, 181–234.
- VAN GELDER, A., ROSS, K. A., AND SCHLIPF, J. S. 1991. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM* 38, 3, 620–650.

## Appendix A Proofs

### Theorem 1

Let  $U \in \mathcal{I}$ , and  $P$  be a program. The fixpoint  $T_P^U \uparrow \mathbf{0}$  is reached after a linear number of iterations, measured on the number of atoms appearing in  $P$ .

### Proof

W.l.o.g. let us assume that all empty rule bodies are replaced by constant  $\bar{1}$ . Let  $L_0 := \mathbf{0}$  and  $L_{i+1} := T_P^U(L_i)$ ,  $i \geq 0$ . For every  $i \geq 0$  and  $a \in \mathcal{B}$  such that (i)  $L_i(a) < L_{i+1}(a)$ , there are a rule  $r$  and a literal  $b \in B^+(r)$  such that  $H(r) = a$  and  $L_{i+1}(a) = \langle L_i, U \rangle(r)$ . In particular, note that (ii)  $L_{i+1}(a) \leq L_i(b)$ . In this case we say that  $a$  is *inferred* by  $b$ .

Let  $n$  be the number of propositional atoms in  $P$ . We prove that any chain of inferred atoms has length at most  $n + 1$ , which implies that  $n$  applications of  $T_P$  give the fixpoint  $T_P^U \uparrow \mathbf{0}$ . Suppose on the contrary that there are  $a_0, \dots, a_{n+1}$  such that  $a_0$  is a numeric constant and  $a_{i+1} \in \mathcal{B}$  is inferred by  $a_i \in \mathcal{B}$ ,  $0 \leq i \leq n$ . As  $P$  contains  $n$  propositional atoms, there exist  $1 \leq j < k \leq n + 1$  such that  $a_j = a_k$ . Hence, from (i) we have  $L_{i+1}(a_{i+1}) > L_i(a_{i+1})$  for  $i = 0, \dots, n$ , and thus  $L_k(a_k) > L_{k-1}(a_k) \geq L_j(a_k)$  (where the last inequality is due to the monotonicity of  $T_P$ ). From (ii) we have  $L_{i+1}(a_{i+1}) \leq L_i(a_i)$  for  $i = 0, \dots, n$ , and thus  $L_k(a_k) \leq L_j(a_j) = L_j(a_k)$ . Therefore, we have  $L_k(a_k) > L_j(a_k)$  and  $L_k(a_k) \leq L_j(a_k)$ , that is, a contradiction.  $\square$

*Theorem 2*

For FASP programs without numeric constants and crisp sets, Definition 2 coincides with the original notion of unfounded set by Van Gelder et al. (1991).

*Proof*

Let  $L, U$  be crisp sets,  $L \subseteq U \subseteq \mathcal{B}$ , and  $P$  be an ASP program. According to Van Gelder et al. (1991), a crisp set  $Y \subseteq \mathcal{B}$  is an unfounded set for  $P$  w.r.t.  $(L, U)$  if for each  $r \in P$  such that  $H(r) \in Y$ , (1)  $B^+(r) \not\subseteq U$ , or (2)  $B^-(r) \cap L \neq \emptyset$ , or (3)  $B^+ \cap Y \neq \emptyset$ . Let  $X \in \mathcal{I}$  be such that  $X(a) = 1$  if  $a \in Y$ , and  $X(a) = 0$  otherwise. We have to show that  $Y$  is an unfounded set for  $P$  w.r.t.  $(L, U)$  (according to Van Gelder et al.) if and only if  $X$  is a fuzzy unfounded set for  $P$  w.r.t.  $(L, U)$  (according to Definition 2).

( $\Rightarrow$ ) Consider a rule  $r \in P$  such that  $X(H(r)) > 0$ . We have  $H(r) \in Y$ . Any of (1), (2) and (3) implies  $\langle U \cap (\mathbf{1} \setminus X), L \rangle(r) = 0$ .

( $\Leftarrow$ ) Consider a rule  $r \in P$  such that  $H(r) \in Y$ . We have  $X(H(r)) = 1$ , and hence  $0 = [U \cap (\mathbf{1} \setminus X)](H(r)) \geq \langle U \cap (\mathbf{1} \setminus X), L \rangle(r) = 0$ . We prove that if (1) and (3) do not hold, then (2) holds. Falsity of (3) implies  $\langle U \cap (\mathbf{1} \setminus X), L \rangle(r) = \langle U, L \rangle(r)$ , and falsity of (1) implies  $\langle U, L \rangle(r) = L(\text{not } B^-(r))$ . Therefore, there is an element  $b$  of  $B^-(r)$  such that  $L(b) = 1$ , and hence  $B^-(r) \cap L \neq \emptyset$ , i.e., condition (2) is satisfied.  $\square$

*Theorem 3*

Let  $X_1, X_2$  be two fuzzy unfounded sets for  $P$  w.r.t.  $(L, U)$ . Then also  $X_1 \cup X_2$  is an unfounded set for  $P$  w.r.t.  $(L, U)$ .

*Proof*

Let  $X = X_1 \cup X_2$  and  $r \in P$  such that  $X(H(r)) > 0$  holds. We have to show that  $[U \cap (\mathbf{1} \setminus X)](H(r)) \geq \langle U \cap (\mathbf{1} \setminus X), L \rangle(r)$ . Assume w.l.o.g. that  $X(H(r)) = X_1(H(r))$ . Since  $X_1$  is unfounded, it follows that

$$[U \cap (\mathbf{1} \setminus X)](H(r)) = [U \cap (\mathbf{1} \setminus X_1)](H(r)) \geq \langle U \cap (\mathbf{1} \setminus X_1), L \rangle(r) \geq \langle U \cap (\mathbf{1} \setminus X), L \rangle(r),$$

which proves the result.  $\square$

*Theorem 4*

$M$  is a fuzzy answer set of a program  $P$  if and only if  $GUS_P^{M,M} = \mathbf{1} \setminus M$ .

*Proof*

( $\Rightarrow$ ) Since  $\mathbf{1} \setminus (\mathbf{1} \setminus M) = M$  and  $M \models P$ ,  $\mathbf{1} \setminus M$  is an unfounded set. Moreover, we can prove that for any unfounded set  $X$  for  $P$  w.r.t.  $(M, M)$ ,  $M \cap (\mathbf{1} \setminus X) \models P^M$  holds, from which  $X \subseteq \mathbf{1} \setminus M$  follows, since  $M$  is a minimal model of  $P^M$ . Consider a rule  $r \in P$ . As the interpretation of  $B^-(r)$  is fixed in the reduct  $P^M$ , we have to show that  $[M \cap (\mathbf{1} \setminus X)](H(r)) \geq \langle M \cap (\mathbf{1} \setminus X), M \rangle(r)$  holds. If  $X(H(r)) = 0$ , then  $[M \cap (\mathbf{1} \setminus X)](H(r)) = M(H(r)) \geq \langle M, M \rangle(r) \geq \langle M \cap (\mathbf{1} \setminus X), M \rangle$ . Otherwise, it follows that  $[M \cap (\mathbf{1} \setminus X)](H(r)) \geq \langle M \cap (\mathbf{1} \setminus X), M \rangle$  by Definition 2. Thus,  $GUS_P^{M,M} = \mathbf{1} \setminus M$ . ( $\Leftarrow$ ) Let  $GUS_P^{M,M} = \mathbf{1} \setminus M$ . We first show that  $M \models P$ . Let  $r \in P$ . If  $[\mathbf{1} \setminus M](H(r)) = 0$ , then  $M \models r$  because  $M(H(r)) = 1$ . If  $[\mathbf{1} \setminus M](H(r)) > 0$ , then  $M \models r$  follows from Definition 2 and the fact  $M \cap (\mathbf{1} \setminus (\mathbf{1} \setminus M)) = M$ . Hence  $M \models P$ , which in turn implies  $M \models P^M$ . We now prove that for any  $M' \subseteq M$  such that  $M' \models P^M$ ,  $X = \mathbf{1} \setminus M'$  is

an unfounded set for  $P$  w.r.t.  $(M, M)$ , from which  $X \subseteq GUS_P^{M,M} = \mathbf{1} \setminus M$  and thus  $M' = M$ . Consider a rule  $r \in P$  such that  $X(H(r)) > 0$ . Since  $M' \models P^M$ , it holds that  $M'(H(r)) \geq \langle M', M \rangle(r)$ . From  $M' = \mathbf{1} \setminus X$  and  $M' \subseteq M$ , it then holds that  $[M \cap (\mathbf{1} \setminus X)](H(r)) = M'(H(r)) \geq \langle M', M \rangle(r) = \langle M \cap (\mathbf{1} \setminus X), M \rangle(r)$ , which shows that  $X$  is an unfounded set.  $\square$

*Theorem 5*

Let  $L, U \in \mathcal{I}$ ,  $L \subseteq U$ , and  $P$  be a program. If  $\mathbf{1} \setminus R_P^{L,U} \Downarrow \mathbf{1} \subseteq U$ , then  $R_P^{L,U} \Downarrow \mathbf{1} = GUS_P^{L,U}$ .

*Proof*

( $\subseteq$ ) We show that  $X = R_P^{L,U} \Downarrow \mathbf{1}$  is an unfounded set. Consider  $r \in P$  such that  $X(H(r)) > 0$ . We have to show that  $[U \cap (\mathbf{1} \setminus X)](H(r)) \geq \langle U \cap (\mathbf{1} \setminus X), L \rangle(r)$ . Since  $\mathbf{1} \setminus X \subseteq U$ , we can equivalently show  $[\mathbf{1} \setminus X](H(r)) \geq \langle U \cap (\mathbf{1} \setminus X), L \rangle(r)$ . As  $X$  is a fixpoint of  $R_P^{L,U}$ , we have  $X(H(r)) \leq 1 - \langle U \cap (\mathbf{1} \setminus X), L \rangle(r)$ , which implies that  $1 - X(H(r)) \geq \langle U \cap (\mathbf{1} \setminus X), L \rangle(r)$ .

( $\supseteq$ ) Let now  $Y$  be an unfounded set. We will show that (i)  $R_P^{L,U}(Y) = Y$  and (ii)  $I \subseteq J$  implies  $R_P^{L,U}(I) \subseteq R_P^{L,U}(J)$ , from which we derive  $Y \subseteq R_P^{L,U} \Downarrow \mathbf{1}$ . To show (i), consider  $a \in \mathcal{B}$ . If  $Y(a) = 0$ , then also  $R_P^{L,U}(Y) = 0$  by definition. Let now  $Y(a) > 0$ , and suppose that there is some  $r \in P$  such that  $Y(a) > 1 - \langle U \cap (\mathbf{1} \setminus Y), L \rangle(r)$ . This is equivalent to  $1 - Y(a) < \langle U \cap (\mathbf{1} \setminus Y), L \rangle(r)$ . Since  $Y$  is unfounded, it must hold that  $[U \cap (\mathbf{1} \setminus Y)](a) \geq \langle U \cap (\mathbf{1} \setminus Y), L \rangle(r)$ , and thus we have that

$$[U \cap (\mathbf{1} \setminus Y)](a) \geq \langle U \cap (\mathbf{1} \setminus Y), L \rangle(r) > 1 - Y(a),$$

which is a contradiction with the fact that  $[U \cap (\mathbf{1} \setminus Y)](a) \leq 1 - Y(a)$  must hold. To show (ii), we just note that  $1 - \langle U \cap (\mathbf{1} \setminus I), L \rangle(r) \leq 1 - \langle U \cap (\mathbf{1} \setminus J), L \rangle(r)$  holds for every  $r \in P$ .  $\square$

*Theorem 6*

Let  $L, U \in \mathcal{I}$  and  $P$  be a program. The fixpoint  $R_P^{L,U} \Downarrow \mathbf{1}$  can be computed in polynomial time on the size of  $P$ .

*Proof*

It is easy to see that one application of the  $R_P$  operator requires linear time in the number of rules  $P$ . Moreover, as in Theorem 1, the greatest fixpoint  $R_P^{L,U} \Downarrow \mathbf{1}$  is obtained after at most as many applications of this operator as there are atoms in  $P$ . In total, this means that this fixpoint can be computed in polynomial time on the size of  $P$ .  $\square$

*Theorem 7*

Let  $P$  be a program,  $L, U$  two interpretations,  $(L', U') = W_P(L, U)$ , and  $M$  an answer set of  $P$ . If  $L \subseteq M \subseteq U$ , then  $L' \subseteq M \subseteq U'$ .

*Proof*

By definition,  $L'(a) = [T_P^U(L)](a) = \max\{\langle L, U \rangle(r) \mid H(r) = a\}$ . By assumption,  $L \subseteq M \subseteq U$  and hence  $\langle L, U \rangle(r) \leq \langle M, M \rangle(r) = M(B(r))$  for every rule  $r$ . Since  $M$  is a model of its reduct, we have  $L'(a) \leq \max\{M(B(r)) \mid H(r) = a\} \leq M(a)$ .

On the other hand, since  $M$  is an answer set of  $P$ , by Theorems 4 and 5 it follows that  $\mathbf{1} \setminus M = GUS_P^{M,M} = R_P^{M,M} \Downarrow \mathbf{1} \supseteq R_P^{L,U} \Downarrow \mathbf{1}$ . This implies that  $M \subseteq \mathbf{1} \setminus R_P^{L,U} \Downarrow \mathbf{1} = U'$   $\square$

*Theorem 8*

Let  $P$  be a stratified program. The least fixpoint of  $W_P$  coincides with the unique answer set of  $P$  and is computable in polynomial time.

*Proof*

The fact that  $W_P \uparrow (\mathbf{0}, \mathbf{1})$  is the unique answer set of  $P$  follows using the same ideas presented in (Lukasiewicz 2006). Each application of  $W_P$  requires a computation of  $T_P$  and one of  $R_P^{L,U} \downarrow \mathbf{1}$ , each of which is polynomial on the number of atoms in  $P$ . An increase of  $L(a)$  in the  $i$ -th iteration of  $W_P$  is caused by an increase of  $L(b)$  or a decrease of  $U(b)$  in the previous iteration, for some  $b \in \mathcal{B}$ . As in the proof of Theorem 1, this implies that at most linearly many iterations (on the number of atoms appearing in  $P$ ) can be applied before a fixpoint is reached.  $\square$

*Theorem 9*

Let  $L, U \in \mathcal{I}$ , and  $P$  be a program. If  $M \models P$  and  $L \subseteq M \subseteq U$ , then  $S_P^U(L) \subseteq M \subseteq U$ .

*Proof*

Since  $M \models P$  and  $L \subseteq M$ , we have that  $[S_P^U(L)](a) \leq M(a)$  for every atom  $a \in \mathcal{B}$ , and hence  $S_P^U(L) \subseteq M$ .  $\square$

*Theorem 10*

Let  $P$  be a program over the Łukasiewicz t-norm, and  $L, U \in \mathcal{I}$ . For every atom  $a \in \mathcal{B}$  it holds that  $[S_P^U(L)](a) = \min\{I(a) \mid I \text{ satisfies } \mathbb{L}_P \cup \{L(b) \leq I(b) \leq U(b) \mid b \in \mathcal{B}\}\}$ .

*Proof*

Let  $r$  be a rule of the form (1), and  $I$  an interpretation.  $I \models r$  if and only if

$$I(a) \geq I(B(r)) = \max\{I(b_1) + \dots + I(b_m) - I(b_{m+1}) - \dots - I(b_n) + 1 - m, 0\}.$$

Since  $I(a) \geq 0$ ,  $I \models P$  if and only if  $I$  satisfies the system  $\mathbb{L}_P$ . Additionally,  $L \subseteq I \subseteq U$  if and only if for every  $b \in \mathcal{B}$  it holds that  $L(b) \leq I(b) \leq U(b)$ . Finally, as the feasible region is closed, the optimal can be reached.  $\square$

*Theorem 11*

Let  $L, U \in \mathcal{I}$ , and  $P$  be a program over the Łukasiewicz t-norm.  $S_P^U(L)$  is computable in polynomial time w.r.t. the number of rules.

*Proof*

The computation of the  $S_P$  operator requires to solve one linear programming problem for each atom  $a$  appearing in  $P$ . Linear programming is well-known to be solvable in polynomial time on the number of restrictions. As the size of  $\mathbb{L}_P$  corresponds to the number of rules in  $P$ , this yields a polynomial complexity upper bound.  $\square$

*Theorem 12*

The fixpoint of  $W_P$  gives the well-founded semantics by Damásio and Pereira (2001).

*Proof*

The well-founded semantics by Damásio and Pereira (2001) is defined as the fixpoint of the following operator (adapted to our notation):

$$\Omega(L, U) := (T_P^U \uparrow \mathbf{0}, T_P^L \uparrow \mathbf{0})$$

Let  $U' := T_P^L \uparrow \mathbf{0}$ . The claim immediately follows by Theorem 5 and the following property (proved below):

$$U' \subseteq U \implies \mathbf{1} \setminus U' = GUS_P^{L,U}.$$

We first prove that  $\mathbf{1} \setminus U'$  is unfounded w.r.t.  $(L, U)$ . Let  $r$  be a rule in  $P$  such that  $U'(H(r)) > 0$ . We have

$$\begin{aligned} [U \cap (\mathbf{1} \setminus (\mathbf{1} \setminus U'))](H(r)) &= [U \cap U'](H(r)) = U'(H(r)) \\ &\geq \max\{\langle U, L \rangle(r') \mid r' \in P, H(r') = H(r)\} \\ &\geq \langle U, L \rangle(r) \\ &\geq \langle U \cap U', L \rangle(r) \\ &= \langle U \cap (\mathbf{1} \setminus (\mathbf{1} \setminus U')), L \rangle(r). \end{aligned}$$

We complete the proof by proving  $X \subseteq \mathbf{1} \setminus U'$ , or equivalently  $U' \subseteq \mathbf{1} \setminus X$ , for every unfounded set  $X$  w.r.t.  $(L, U)$ . To this aim, let  $U_n$  be the  $n$ -th application of  $T_P^U$  to  $\mathbf{0}$ ,  $n \geq 0$ . We prove  $U_n \subseteq \mathbf{1} \setminus X$  by induction on  $n \geq 0$ .

For  $n = 0$ , the result holds trivially, since  $U_0 = \mathbf{0}$ . Suppose now that  $U_n \subseteq \mathbf{1} \setminus X$  holds for  $n \geq 0$  in order to show  $U_{n+1} \subseteq \mathbf{1} \setminus X$ . Since  $T_P$  is monotonic, we know that  $U_n \subseteq U'$ . By combining with the induction hypothesis and the original assumption  $U' \subseteq U$ , we have that  $U_n \subseteq U \cap (\mathbf{1} \setminus X)$ . Consider now an atom  $a$  such that  $X(a) > 0$ . We have

$$\begin{aligned} U_{n+1}(a) &= \max\{\langle U_n, L \rangle(r) \mid r \in P, H(r) = a\} \\ &\leq \max\{\langle U \cap (\mathbf{1} \setminus X), L \rangle(r) \mid r \in P, H(r) = a\} \\ &\leq [U \cap (\mathbf{1} \setminus X)](a) \\ &\leq [\mathbf{1} \setminus X](a), \end{aligned} \tag{A1}$$

where line (A1) follows from the assumption that  $X$  is an unfounded set w.r.t.  $(L, U)$ . Hence,  $U_{n+1}(a) \leq 1 - X(a)$  holds for every  $a \in \mathcal{B}$ , which complete our proof.  $\square$