Annotated Answer Set Programming

Umberto Straccia
Istituto di Scienza e Tecnologie dell'Informazione"A. Faedo", Via G. Moruzzi 1
Pisa, I-56124, ITALY
straccia@isti.cnr.it

Abstract

We present *Annotated Answer Set Programming*, that extends the expressive power of disjunctive logic programming with annotation terms, taken from the generalized annotated logic programming framework.

Keywords: many-valued logic programming

1 Introduction

The management of uncertainty and/or imprecision within logic programming has attracted the attention of many researchers and numerous frameworks have been proposed. Essentially, they differ in the underlying notion of uncertainty theory and imprecision theory (see e.g. for *Probability theory* [21, 27], *Fuzzy set theory* [30, 32, 33], *Multi-valued logic* [4, 5, 6, 7, 8, 13, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 31], *Possibilistic logic* [3, 10, 29]) and how uncertainty/imprecision values, associated to rules and facts, are managed.

Under *uncertainty theory* fall all those approaches in which statements rather than being either true or false, are true or false to some *probability* or *possibility/necessity*, while under *imprecision theory* fall all those approaches in which statements are true to some degree, which is taken from a truth space (see [11] for a clarification between the notions of uncertainty and imprecision). In this work we deal with *imprecision* and, thus, statements have a degree of truth. Current frameworks for managing imprecision in logic programming can roughly be classified into *Annotation Based* (AB) and *Implication Based* (IB). In the AB approach (e.g. [15, 27]), a rule is of the form $A\colon f(\beta_1,\ldots,\beta_n) \leftarrow B_1\colon \beta_1,\ldots,B_n\colon \beta_n$ which asserts "the value of atom A is at least (or is in) $f(\beta_1,\ldots,\beta_n)$, whenever the value of atom B_i is at least (or is in) β_i, $1 \leq i \leq n$". Here f is an n-ary computable function and β_i is either a constant or a variable ranging over an appropriate truth domain. In the IB approach, (e.g. [4, 8, 16, 25, 31, 32, 33] a rule is of the form $A \stackrel{\alpha}{\leftarrow} B_1, ..., B_n$ which says that the value associated with the implication $B_1 \wedge ... \wedge B_n \rightarrow A$ is α. Computationally, given an assignment I of values to the B_i, the value of A is computed by taking the "conjunction" of the values $I(B_i)$ and then somehow "propagating" it to the rule head. The values the atoms may have are taken from a lattice.

However, only a few deal with disjunctive logic programs ([22, 23]) and only [22] deals with (a restricted form of) disjunctive logic programs with non-monotone negation under the IB approach. In this work we will consider disjunctive logic programs with negation. However, the approach we follow in this paper is annotation-based. We deserve more attention to the implication-based approach in future work.

We proceed as follows. In the next section we provide some basic definitions about *disjunctive logic programs* [14]. We then present *an-*

notated disjunctive logic programs in which we extend the rules with annotation terms and conclude with an outlook to some topics for further research.

2 Answer Set Programming

We recall disjunctive logic programs under the answer set semantics.

A *term* t is defined as either a constant or a variable. An *atom* is of the form $P(t_1, \ldots, t_n)$, where all t_i are terms and P is a n-ary predicate symbol. *Ground atoms* are atoms without variables. A *literal* L is either a *positive literal* $L = A$, or a *negative literal* $L = \neg A$, where A is an atom. A *ground literal* is a literal without variables. An *extended literal* is a literal or an expression of the form $\text{not}(L)$ ("L is not provable"), where L is a literal. A *ground extended literal* is an extended literal without variables. For a set $\mathcal{X}$ of extended literals, $\mathcal{X}^- = \{\text{not}(L) \mid \text{not}(L) \in \mathcal{X}\}$, while $\neg\mathcal{X} = \{\neg L \mid L \in \mathcal{X}, L \text{ literal}\}$, where we define $\neg\neg A = A$. A *disjunctive logic program*, also called *answer set program* (asp) $\mathcal{P}$, is a finite set of *rules* of the form $\gamma \leftarrow \delta$ where γ and δ are finite sets of extended literals. For ease, we may omit the graph brackets in a rule. γ is called the *head* of the rules, while δ is called the *body*. The head is interpreted as the disjunction of its components, while the body is interpreted as a conjunction. A *fact* is a rule with empty body, while a *constraint* is a rule with empty head. For ease, we may also write a fact as γ in place of $\gamma \leftarrow$.

We call programs where for each rule $\gamma^- \cup \delta^- = \emptyset$, *programs without negation as failure (naf)*. Programs, where in each rule $|\gamma| = 1$ are called *normal*. Programs without naf, containing positive literals only, are called *positive*. Programs that do not contain variables are called *ground*.

The *Herbrand universe* $H_\mathcal{P}$ of $\mathcal{P}$ is the set of constants appearing in $\mathcal{P}$. If no constant appears in $\mathcal{P}$, then $H_\mathcal{P} = \{a\}$ for an arbitrary constant a. The *grounded program*, $\mathcal{P}^*$, is obtained from $\mathcal{P}$ by substituting every variable in $\mathcal{P}$ by every possible constant in $H_\mathcal{P}$. The *Herbrand base* $\mathcal{B}_\mathcal{P}$ of $\mathcal{P}$ is the set of ground atoms A that can be constructed using the predicate symbols in $\mathcal{P}$ and constants in $H_\mathcal{P}$.

An *interpretation* I of a grounded program $\mathcal{P}$ is any consistent set of literals being a subset of $\mathcal{B}_\mathcal{P} \cup \neg\mathcal{B}_\mathcal{P}$. I *satisfies* a literal L iff $L \in I$. Furthermore, we say that I *satisfies* an extended literal $\text{not}(L)$ iff I does not satisfy L, i.e. $L \notin I$. An interpretation I of a grounded program $\mathcal{P}$ without naf *satisfies* a rule $\gamma \leftarrow \delta$ in $\mathcal{P}$ iff $\gamma \cap I \neq \emptyset$ whenever $\delta \subseteq I$. An interpretation I is a *model* of a grounded program $\mathcal{P}$ without naf iff it satisfies every rule in $\mathcal{P}$. I is a *minimal model* of $\mathcal{P}$ iff I is a model of $\mathcal{P}$ and there is no model $J \subset I$ of $\mathcal{P}$.

For a grounded program $\mathcal{P}$ and an interpretation I, the Gelfond-Lifschitz transformation [14], is the program $\mathcal{P}[I]$ without naf, obtained by deleting in $\mathcal{P}$, (i) each rule that has $\text{not}(L)$ in its body with $L \in I$; (ii) each rule that has $\text{not}(L)$ in its head with $L \notin I$; and (iii) all $\text{not}(L)$ in the bodies and heads of the remaining rules. Finally, an *interpretation* of a program $\mathcal{P}$ (possibly not grounded) is an interpretation I of the grounded program $\mathcal{P}^*$. An interpretation I of a program $\mathcal{P}$ is a *stable model* of $\mathcal{P}$ iff I is a minimal model of $\mathcal{P}^*[I]$. It can easily be shown that, if I is a stable model of $\mathcal{P}$, then I is a model of $\mathcal{P}^*$.

Example 1 *Consider the program $\mathcal{P}$ with rules $A \leftarrow not(B)$ and $B \leftarrow not(A)$. Then $I_1 = \{A\}$ and $I_2 = \{B\}$ are the only stable models of $\mathcal{P}$.*

3 Annotated ASP

Classically, n-ary predicates may be seen as functions from their domain into the complete lattice $\mathcal{L} = \langle\{\mathtt{f}, \mathtt{t}\}, \preceq\rangle$, where $\mathtt{f}$ stands for $false$, while $\mathtt{t}$ stands for $true$ and $\mathtt{f} \preceq \mathtt{t}$. We generalize this by considering arbitrary complete lattices $\mathcal{L} = \langle T, \preceq\rangle$ as our truth-space, with the restriction that the truth-set T is *countable*. With $\bot$ and $\top$ we indicate the top and the bottom element of lattice, respectively. The meet is $\wedge$, while the join is $\vee$.

We map now n-ary predicates into functions from their domain into T. The associated value $c \in T$ indicates the degree of truth of

a predicate. We assume there is a function from T to T, wich is called *negation function* (denoted $\neg$) such that it is anti-monotone w.r.t. $\preceq$.

Typical complete lattices are (just mentioning a few): given a set of rational values T, consider $\mathcal{L}_T = \langle T, \leq \rangle$. Then $\mathcal{L}_{\{0,1\}}$ corresponds to the classical truth-space, where 0 stands for 'false', while 1 stands for 'true', while $\mathcal{L}_{[0,1]_\mathbb{Q}}$, which relies on the unit interval, restricted to the rationals, is quite frequently used as truth-space. In $\mathcal{L}_{[0,1]_\mathbb{Q}}$, $\neg c = 1 - c$ is a frequently used negation function. For any $n \in \mathbb{N}$, $\mathcal{L}_n = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}, 1\}$ together with $\neg c = 1 - c$ is often used in the context of many-valued logics. Another frequent truth-space is Belnap's $\mathcal{FOUR}$ [1], where T is $\{f, t, u, i\}$ with $f \preceq u \preceq t$ and $f \preceq i \preceq t$. Here, u stands for 'unknown', whereas i stands for 'inconsistency'. We denote the lattice as $\mathcal{L}_B$. Additionally, besides $\neg f = t$, we have $\neg u = u$ and $\neg i = i$. Another truth-space is $\mathcal{L}_4$, where T is $\{f, cf, ct, t\}$ with $f \preceq cf \preceq ct \preceq t$. Here, cf stands for 'close to false', whereas ct stands for 'close to true'. Besides $\neg f = t$, we have $\neg cf = ct$.

In annotated ASP, from a syntax point of view, we extend ASP with annotation terms, which are taken from the so-called *Generalized Annotated Logic Programs* (GAP) framework [15] (see also [28]). So, fix a complete lattice $\mathcal{L} = \langle T, \preceq \rangle$. Let us define an *annotation function* of arity n to be a total and computable function[1] $f\colon T^n \to T$. Assume a new alphabet of *annotation variables*, which will denote a value in T and can only appear in so-called *annotation terms*. An *annotation item*, ξ, is defined inductively: (i) as a truth value $c \in T$, or as an annotation variable x, or (ii) is of the form $f(\xi_1, \ldots, \xi_n)$, where f is an n-ary annotation function and all ξ_i are annotation items. An *annotation term*, τ, is of the form $\langle \xi, \xi' \rangle$, where ξ and ξ' are annotation items. Informally, an annotation term $\langle c_1, c_2 \rangle$ $(c_1, c_2 \in T)$ is supposed to denote the set of truth values $\{c \in T\colon c_1 \preceq c \preceq c_2\}$, i.e. an interval. We say that $c \in \langle c_1, c_2 \rangle$ iff $c_1 \preceq c \preceq c_2$.

[1]The result of f is computable in a finite amount of time.

Now, let L be a literal and τ an annotation term. An *annotated literal*, *a-literal* for short, $\bar{L}$, is of the form $L{:}\tau$. The intended meaning is that "the truth-value of L lies in the interval τ". An *extended a-literal* is of the form $\mathrm{not}(\bar{L})$, where $\bar{L}$ is an a-literal. The intended meaning of $\mathrm{not}(L{:}\tau)$ is that "it is not provable that the truth of L lies in the interval τ". An *a-disjunctive logic program*, also called *a-answer set program* (a-asp), or simply a-program $\mathcal{P}$, is a finite set of *a-rules* of the form $\gamma \leftarrow \delta$, where γ and δ are finite sets of extended a-literals.

While the informal semantics of rules of an a-asp may be intuitive, the formal development is more involved as for the classical counterpart. In order to avoid straightforward repetition, if not stated otherwise, definitions related to a-asp, parallels those for asp. Furthermore, in *grounding* an a-literal $L{:}\tau$, we assume that the annotation term τ is grounded as well, i.e. annotation variables are replaced with truth values in T and annotation items of the form $f(\xi_1, \ldots, \xi_n)$ are replaced with the result of the computation of $f(\xi_1, \ldots, \xi_n)$. Note that a grounded a-program $\mathcal{P}$ may contain an infinite, but countable (as T is countable), number of rules due to the grounding of annotation terms. While from a semantics point of view this is unproblematic, from a computational point of view this may lead to undecidability in general. We will see later on some useful cases in which decidability is guaranteed. We denote with $\mathcal{P}^*$ the grounded program obtained from $\mathcal{P}$. For a grounded a-program $\mathcal{P}$, $\mathcal{B}_\mathcal{P}$ is the set of ground atoms A that can be constructed using the predicate symbols in $\mathcal{P}$ and the constants in $H_\mathcal{P}$. Note that (i) the Herbrand base of $\mathcal{P}$ contains the constants appearing in $\mathcal{P}$ while it does not contain ground annotation terms occurring in $\mathcal{P}$; and (ii) $\mathcal{B}_\mathcal{P}$ contains atoms A and not annotated atoms.

An *a-interpretation* I of a grounded a-program $\mathcal{P}$ is any (possibly partial) function $I\colon \mathcal{B}_\mathcal{P} \to T$ (*some ground atoms may be left unspecified*). The set of defined atoms in I is denoted $\mathtt{def}(I)$. In the following, whenever we write $I(A)$, we assume that $A \in \mathtt{def}(I)$.

We extend and interpretation I to literals $L = \neg A$ in the obvious way: $I(L) = \neg I(A)$. An a-interpretation I *satisfies* a ground a-literal $L{:}\tau$ iff $I(L) \in \tau$. Note that we can always assume that a-literals are positive, by replacing $\neg A{:}[\xi_1, \xi_2]$ with $A{:}[\neg\xi_2, \neg\xi_1]$. Like for disjunctive logic programs, we say that I *satisfies* an extended a-literal not$(L{:}\tau)$ iff I does not satisfy $L{:}\tau$, i.e. $I(L) \notin \tau$. Finally, an a-interpretation I of a grounded program $\mathcal{P}$ without naf *satisfies* a rule $\gamma \leftarrow \delta$ iff if I satisfies *every* a-literal in δ then I satisfies *some* a-literal in γ. I is an *a-model* of program $\mathcal{P}$ without naf iff it satisfies every rule in $\mathcal{P}$.

Example 2 *Consider the grounded a-program $\mathcal{P}$ without naf, with the two facts, namly $\{A{:}\,\langle 0.2, 0.7\rangle, B{:}\,\langle 0.3, 0.6\rangle\}$ and $C{:}\langle 0.1, 0.3\rangle$, over $\mathcal{L}_{[0,1]_\mathbb{Q}}$* [2]*. The a-program above has infinitely many models I, which map $I(C) \in \langle 0.1, 0.3\rangle$ and either $I(A) \in \langle 0.2, 0.6\rangle$ or $I(B) \in \langle 0.3, 0.6\rangle$. Note that I must be defined on C, but may not necessarily be defined on both A and B.*

Now, consider the following two partial functions $\mathscr{I}_1$ and $\mathscr{I}_2$, assigning to atoms intervals and defined as follows: $\mathscr{I}_1(A) = \langle 0.2, 0.7\rangle$, $\mathscr{I}_1(C) = \langle 0.1, 0.3\rangle$ and $\mathscr{I}_2(B) = \langle 0.3, 0.6\rangle$, $\mathscr{I}_2(C) = \langle 0.1, 0.3\rangle$. Then for each a-model I of $\mathcal{P}$, we have that for some $i = 1, 2$, $\mathtt{def}(\mathscr{I}_i) \subseteq \mathtt{def}(I)$ and for all ground atoms $D \in \mathtt{def}(\mathscr{I}_i), I(D) \in \mathscr{I}_i(D)$. Essentially, $\mathscr{I}_1$ and $\mathscr{I}_2$ are minimal in terms of defined atoms and the intervals are the 'most precise' intervals that can be inferred from the program. $\mathscr{I}_1$ and $\mathscr{I}_2$ represent concisely all the models of the a-program above. □

In the following we formally define the above concept of "interval interpretation". So, let $\mathcal{C}(T)$ be the set of all "closed sub-intervals of T", i.e. the set of pairs $\langle c_1, c_2\rangle$, for $c_1, c_2 \in T$. Note that we do not impose $c_1 \preceq c_2$. By abuse of terminology, we call a pair $\langle c_1, c_2\rangle$ interval. For two intervals $\langle c_1, c_2\rangle$ and $\langle c_3, c_4\rangle$ in $\mathcal{C}(T)$, we define $\langle c_1, c_2\rangle \preceq_p \langle c_3, c_4\rangle$ iff $c_1 \preceq c_3$ and $c_4 \preceq c_2$. That is, the former interval is less narrow than the latter. For instance, $\langle 0, 1\rangle \preceq_p \langle 0.2, 0.6\rangle \preceq_p \langle 0.3, 0.5\rangle \preceq_p \langle 0.6, 0.2\rangle \preceq_p \langle 1, 0\rangle$. Similarly, for two intervals σ_1 and σ_2 in $\mathcal{C}(T)$, we define $\sigma_1 \prec_p \sigma_2$ iff $\sigma_1 \preceq_p \sigma_2$ and $\sigma_2 \not\preceq_p \sigma_1$. Furthermore, we define $\neg\langle c_1, c_2\rangle = \langle \neg c_2, \neg c_1\rangle$. It follows that the $\preceq_p$-least interval is $\langle \bot, \top\rangle$ (complete ignorance about the atom's truth boundaries), the $\preceq_p$-greatest interval is $\langle \top, \bot\rangle$ (maximal inconsistency about the atom's truth). The attentive reader will notice that $\preceq_p$ corresponds exactly to the so-called *knowledge order* on interval bilattices introduce by Fitting [13]. It models the fact that the more precise intervals are the more knowledge we have about the atom's truth.

An *interval interpretation* $\mathscr{I}$ of a grounded program $\mathcal{P}$ without naf, is a (possibly partial) function $\mathscr{I}\colon \mathcal{B}_\mathcal{P} \to \mathcal{C}(T)$. That is, an interval interpretation $\mathscr{I}$ is a representative of a whole family of a-interpretations I: we write $I \in \mathscr{I}$ iff $\mathtt{def}(I) = \mathtt{def}(\mathscr{I})$ and for all $A \in \mathtt{def}(\mathscr{I})$, $I(A) \in \mathscr{I}(A)$. We extend interval interpretations $\mathscr{I}$ to ground literals $L = \neg A$ in the obvious way: $\mathscr{I}(L) = \neg\mathscr{I}(A)$. Furthermore, for interval interpretations $\mathscr{I}_1$ and $\mathscr{I}_2$, we define $\mathscr{I}_1 \preceq_p \mathscr{I}_2$ iff $\mathtt{def}(\mathscr{I}_1) \subseteq \mathtt{def}(\mathscr{I}_2)$ and for all $A \in \mathtt{def}(\mathscr{I}_1), \mathscr{I}_1(A) \preceq_p \mathscr{I}_2(A)$. Furthermore, $\mathscr{I}_1 \prec_p \mathscr{I}_2$ iff $\mathscr{I}_1 \preceq_p \mathscr{I}_2$ and either $\mathtt{def}(\mathscr{I}_1) \subset \mathtt{def}(\mathscr{I}_2)$ or for some $A \in \mathtt{def}(\mathscr{I}_1), \mathscr{I}_1(A) \prec_p \mathscr{I}_2(A)$. The $\preceq_p$-greatest interval interpretation, $\mathscr{I}_\top$, assigns to all ground atoms in $\mathcal{B}_\mathcal{P}$ the maximal inconsistent interval $\langle \top, \bot\rangle$, while the $\preceq_p$-least interval interpretation, $\mathscr{I}_\bot$ is undefined on all ground atoms in $\mathcal{B}_\mathcal{P}$, i.e. $\mathtt{def}(\mathscr{I}_\bot) = \emptyset$.

An interval interpretation $\mathscr{I}$ *satisfies* a ground a-literal $L{:}\tau$ iff $\tau \preceq_p \mathscr{I}(L)$. That is, $\mathscr{I}$ assigns to L a more precise interval than the constraint imposed by τ. We say that $\mathscr{I}$ *satisfies* a rule $\gamma \leftarrow \delta$ iff if $\mathscr{I}$ satisfies every a-literal in δ then $\mathscr{I}$ satisfies some a-literal in γ. $\mathscr{I}$ is an *interval model* of program $\mathcal{P}$ without naf iff it satisfies every rule in $\mathcal{P}$. Furthermore, $\mathscr{I}$ is *minimal* as well iff there is no interval model $\mathscr{I}' \prec_p \mathscr{I}$ of $\mathcal{P}$. For instance, in Example 2, $\mathscr{I}_1$ and $\mathscr{I}_2$ are the only two minimal interval models of $\mathcal{P}$.

As in[15], there are positive a-programs $\mathcal{P}$, which have an unique minimal model, which

[2] Note that the first one is has a disjunction in the head.

may not be computed in finite time. In fact the grounded a-program without naf, containing all ground instances of the rules over $\mathcal{L}_{[0,1]_\mathbb{Q}}$

$$\begin{array}{rcl} A{:}\langle 0,1\rangle. & & \\ A{:}\langle \frac{x+1}{2}, x'\rangle & \leftarrow & A{:}\langle x, x'\rangle \\ B{:}\langle 1,1\rangle & \leftarrow & A{:}\langle 1,1\rangle \end{array} \quad (1)$$

is not finite, i.e. $\mathcal{P}^*$ is not finite. However, $\mathcal{P}$ has unique minimal interval model $\mathscr{I}(A) = \mathscr{I}(B) = \langle 1,1\rangle$. Note also that the least and unique interval model $\mathscr{I}$ of the a-program with two facts $A{:}\langle 0, 0.3\rangle$ and $A{:}\langle 0.4, 0.5\rangle$ assigns $\mathscr{I}(A) = \langle 0.4, 0.3\rangle$. Therefore, the program has an interval model, but no a-model (indeed there cannot be an interpretation assigning to A a value c such that $c \in \langle 0.4, 0.3\rangle$, i.e. $0.4 \leq c \leq 0.3$), which indicates that on A the a-program is inconsistent [3].

Given a grounded a-program (possibly with naf) $\mathcal{P}$ and an a-interpretation I for $\mathcal{P}$, the *Gelfond-Lifschitz transformation* is the grounded positive a-program $\mathcal{P}[I]$, obtained by deleting in $\mathcal{P}$, (i) each rule that has not$(\bar{L})$ in its body and I satisfies $\bar{L}$; (ii) each rule that has not$(\bar{L})$ in its head and I does not satisfy $\bar{L}$; and (iii) all not$(\bar{L})$ in the bodies and heads of the remaining rules.

An *a-interpretation* I of an a-program $\mathcal{P}$ (possibly not grounded) is an a-interpretation of its grounded version $\mathcal{P}^*$. I is a *stable a-model* of $\mathcal{P}$ iff $I \in \mathscr{I}$ for a minimal interval model $\mathscr{I}$ of $\mathcal{P}^*[I]$. Essentially, in order to check if an a-interpretation I, assigning truth values to atoms, is a stable model of $\mathcal{P}$ we (i) consider its grounded version $\mathcal{P}^*$ in which we replace all variables in atoms with all possible combinations of constants of the Herbrand base and all annotation variables with all possible combinations of truth values of T; (ii) then we compute the Gelfond-Lifschitz transformation $\mathcal{P}^*[I]$ of $\mathcal{P}^*$ with respect to I; (iii) as now $\mathcal{P}^*[I]$ is positive it has a minimal interval model $\mathscr{I}$. $\mathscr{I}$ will provide us the most precise interval we can infer for all the atoms; and finally (iv) we check if indeed the a-interpretation I we want to check satisfies the interval constraints imposed by $\mathscr{I}$.

For instance, in Example 2, we have that any stable a-model I of $\mathcal{P}$ is such that $I \in \mathscr{I}_1$ or $I \in \mathscr{I}_2$. In the following example we will illustrate with an extensive example all the above definitions.

Example 3 *Consider the following a-programs $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ and $\mathcal{P}_4$, where*

$$\mathcal{P}_1 = \{r_1, r_2\}\ , \quad \mathcal{P}_2 = \{r_1, r_3\}$$
$$\mathcal{P}_3 = \{r_1, r_4\}\ , \quad \mathcal{P}_4 = \{r_1, r_5\}$$

and the rules $r_1, \ldots, r_5$ are

$$\begin{array}{lcl} r_1 : A{:}\langle 0.6, 0.8\rangle & & \\ r_2 : B{:}\langle 0.4, 0.5\rangle & \leftarrow & \mathit{not}(A{:}\langle 0.2, 0.3\rangle) \\ r_3 : B{:}\langle 0.4, 0.5\rangle & \leftarrow & \mathit{not}(A{:}\langle 0.2, 0.7\rangle) \\ r_4 : B{:}\langle x, x'\rangle & \leftarrow & A{:}\langle x, x'\rangle \\ r_5 : B{:}\langle x, x'\rangle & \leftarrow & \mathit{not}(A{:}\langle x, x'\rangle) \end{array}$$

Now, let us analyze the behavior of each program $\mathcal{P}_i$.

Case $\mathcal{P}_1$. *$\mathcal{P}_1$ is already grounded, but with not and the Herbrand base is $\mathcal{B}_{\mathcal{P}_1} = \{A, B\}$. In any a-model I of $\mathcal{P}_2$ we have that $I(A) \in \langle 0.6, 0.8\rangle$, $I(B) \in \langle 0.4, 0.5\rangle$ and, thus, I does not satisfy $A{:}\langle 0.2, 0.7\rangle$. Therefore, I does satisfy not$(A{:}\langle 0.2, 0.7\rangle)$. Then ${\mathcal{P}_1}^*[I]$ contains the rule r_1 and the fact $B{:}\langle 0.4, 0.5\rangle$, which is obtained from the rule r_2 by removing not$(A{:}\langle 0.2, 0.7\rangle)$, according to the Gelfond-Lifschitz transformation. ${\mathcal{P}_1}^*[I]$ has an unique minimal interval model $\mathscr{I}$, which is $\mathscr{I}(A) = \langle 0.6, 0.8\rangle$, $\mathscr{I}(B) = \langle 0.4, 0.5\rangle$. Therefore, $I \in \mathscr{I}$ holds and, thus, I is a stable a-model of $\mathcal{P}_1$.*

Case $\mathcal{P}_2$. *$\mathcal{P}_2$ is already grounded, but with not and the Herbrand base is $\mathcal{B}_{\mathcal{P}_2} = \{A, B\}$. In any a-model I of $\mathcal{P}_2$ we have that $I(A) \in \langle 0.6, 0.8\rangle$. If further $I(A) \notin \langle 0.6, 0.7\rangle$ then $I(B) \in \langle 0.4, 0.5\rangle$ else I may or may not be defined on B. Now, assume the case $I(A) \in \langle 0.6, 0.7\rangle$ and I undefined on B. Then ${\mathcal{P}_2}^*[I]$ contains the rule r_1, but does not contain the rule r_3, as I does satisfy $A{:}\langle 0.2, 0.7\rangle$. Hence, the minimal interval model $\mathscr{I}$ of ${\mathcal{P}_2}^*[I]$ is such that $\mathscr{I}(A) = \langle 0.6, 0.8\rangle$, while $\mathscr{I}$ is undefined on B, i.e. $B \notin \mathtt{def}(\mathscr{I})$. Therefore, $I \in \mathscr{I}$ and, thus, I is a stable a-model of $\mathcal{P}_2$. On the other hand, assume the*

[3] We may use this property to manage inconsistencies of this kind, but we will not investigate this feature in this paper.

case $I(A) \not\in \langle 0.6, 0.7\rangle$ *and* $I(B) \in \langle 0.4, 0.5\rangle$. *Then* ${\mathcal{P}_2}^*[I]$ *contains the rule* r_1 *and the rule* $B{:}\langle 0.4, 0.5\rangle$, *which is obtained from the rule* r_3 *by removing* $not(A{:}\langle 0.2, 0.7\rangle)$, *according to the Gelfond-Lifschitz transformation (*I *does not satisfy* $A{:}\langle 0.2, 0.7\rangle$ *and, thus, does satisfy* $not(A{:}\langle 0.2, 0.7\rangle)$*). Therefore,* ${\mathcal{P}_2}^*[I]$ *has an unique minimal interval model* $\mathscr{I}$, *which is* $\mathscr{I}(A) = \langle 0.6, 0.8\rangle$, $\mathscr{I}(B) = \langle 0.4, 0.5\rangle$. *Therefore,* $I \in \mathscr{I}$ *holds and, consequently,* I *is a stable a-model of* $\mathcal{P}_2$. *Therefore, for any stable a-model* I *of* $\mathcal{P}_2$, $I(A) \in \langle 0.6, 0.8\rangle$ *and if* $I(A) \not\in \langle 0.6, 0.7\rangle$ *then* $I(B) \in \langle 0.4, 0.5\rangle$ *else* I *is undefined on* B.

Case $\mathcal{P}_3$**.** $\mathcal{P}_3$ *is not grounded, but without not and the Herbrand base is* $\mathcal{B}_{\mathcal{P}_3} = \{A, B\}$. *Then* ${\mathcal{P}_3}^*$, *besides rule* r_1 *contains all instantiations of rule* r_4, *namely:*

$$B{:}\langle 0.6, 0.6\rangle \leftarrow A{:}\langle 0.6, 0.6\rangle$$
$$B{:}\langle 0.6, 0.8\rangle \leftarrow A{:}\langle 0.6, 0.8\rangle$$
$$B{:}\langle 0.8, 0.6\rangle \leftarrow A{:}\langle 0.8, 0.6\rangle$$
$$B{:}\langle 0.8, 0.8\rangle \leftarrow A{:}\langle 0.8, 0.8\rangle$$

In any a-model I *of* $\mathcal{P}_3$, *i.e. of* ${\mathcal{P}_3}^*$, *we have that* $I(A) \in \langle 0.6, 0.8\rangle$, $I(B) \in \langle 0.6, 0.8\rangle$ *and, as* ${\mathcal{P}_3}^*$ *is not-free,* ${\mathcal{P}_3}^*[I] = {\mathcal{P}_3}^*$. *Now,* ${\mathcal{P}_3}^*[I]$, *i.e.* ${\mathcal{P}_3}^*$, *has an unique minimal interval model* $\mathscr{I}$, *which is* $\mathscr{I}(A) = \langle 0.6, 0.8\rangle$, $\mathscr{I}(B) = \langle 0.6, 0.8\rangle$. *Therefore, for any a-interpretation* I, *with* $I(A) \in \langle 0.6, 0.8\rangle$, $I(B) \in \langle 0.6, 0.8\rangle$, $I \in \mathscr{I}$ *holds and, thus,* I *is a stable a-model of* $\mathcal{P}_3$.

Case $\mathcal{P}_4$**.** $\mathcal{P}_4$ *is not grounded, with not and the Herbrand base is* $\mathcal{B}_{\mathcal{P}_4} = \{A, B\}$. *Then* ${\mathcal{P}_4}^*$, *besides rule* r_1 *contains all instantiations of rule* r_5, *namely:*

$$B{:}\langle 0.6, 0.6\rangle \leftarrow not(A{:}\langle 0.6, 0.6\rangle)$$
$$B{:}\langle 0.6, 0.8\rangle \leftarrow not(A{:}\langle 0.6, 0.8\rangle)$$
$$B{:}\langle 0.8, 0.6\rangle \leftarrow not(A{:}\langle 0.8, 0.6\rangle)$$
$$B{:}\langle 0.8, 0.8\rangle \leftarrow not(A{:}\langle 0.8, 0.8\rangle)$$

For any a-model I *of* $\mathcal{P}_4$, *i.e. of* ${\mathcal{P}_4}^*$, *we have that* $I(A) \in \langle 0.6, 0.8\rangle$. *Therefore, in any interpretation* I *satisfying* $I(A) \in \langle 0.6, 0.8\rangle$,

$$\mathcal{P}_4[I] = \{r_1\} \cup \{B{:}\langle c, c'\rangle \mid I(A) \not\in \langle c, c'\rangle, c, c' \in \{0.6, 0.8\}\}\ .$$

In particular, $B{:}\langle 0.8, 0.6\rangle \in \mathcal{P}_4[I]$, *as* $I(A) \not\in \langle 0.8, 0.6\rangle$. *Also, e.g. if* $I(A) = 0.7$ *then also* $\{B{:}\langle 0.6, 0.6\rangle, B{:}\langle 0.8, 0.8\rangle\} \subseteq {\mathcal{P}_4}^*[I]$, *as neither* $I(A) \in \langle 0.6, 0.6\rangle$ *nor* $I(A) \in \langle 0.8, 0.8\rangle$. *Anyway, the least interval model* $\mathscr{I}$ *of* ${\mathcal{P}_4}^*[I]$ *is such that* $\mathscr{I}(A) = \langle 0.6, 0.8\rangle$ *and* $\mathscr{I}(B) = \langle 0.8, 0.6\rangle$. *While* $I(A) \in \langle 0.6, 0.8\rangle = \mathscr{I}(A)$, *it cannot be* $I(B) \in \langle 0.8, 0.6\rangle = \mathscr{I}(B)$ *and, thus,* $I \not\in \mathscr{I}$. *Therefore,* $\mathcal{P}_4$ *has no stable a-model.* □

3.1 Reasoning

Computing stable models is difficult in general as e.g. the a-program $\mathcal{P}$ (1) shows. Indeed, in the general deciding whether an a-interpretation is a stable model is undecidable [15]. Despite this negative result, we show that there are two interesting cases under which it is decidable whether an a-interpretation is a stable model or not.

In the first case, we require that in the lattice $\mathcal{L} = \langle T, \preceq\rangle$, the set of truth values T is *finite*. From a practical point of view this is a limitation we can live with, especially taking into account that computers have finite resources, and thus, only a finite set of truth values can be represented. In particular, this includes also the case of the the rational numbers in $[0, 1]_{\mathbb{Q}}$ under a given fixed decimal precision p (e.g. $p = 2$). Note that the truth spaces described at the beginning of Section 3 fall into this category.

As in the truth space $\mathcal{L} = \langle T, \preceq\rangle$, T is finite, for any a-program $\mathcal{P}$, its grounding $\mathcal{P}^*$ and its Herbrand base $\mathcal{B}_{\mathcal{P}}$ are finite as well. As, $\mathcal{B}_{\mathcal{P}}$ and T are finite, there are also finitely many a-interpretations I and interval interpretations $\mathscr{I}$ of $\mathcal{P}^*$. [4] Now, let I be an a-interpretation. Then $\mathcal{P}^*[I]$ can be computed from $\mathcal{P}^*$ in finite time. Also, computing the set of minimal interval models $\mathscr{I}$ of $\mathcal{P}^*[I]$ can be done in finite time, though, exponential many interval minimal models may exist (note that classical disjunctive logic programs may have exponential many minimal models). Finally, I is a stable a-model of $\mathcal{P}$ iff $I \in \mathscr{I}$ for a minimal interval model $\mathscr{I}$ of $\mathcal{P}^*[I]$. As $I \in \mathscr{I}$ can be checked

[4] Of course, reasoning in annotated LPs is at least as hard as for classical disjunctive logic programming [9, 12], as annotated LPs subsume classical disjunctive logic programs.

in finite time as well, checking whether I is a stable a-model of $\mathcal{P}$ is decidable.

In the second case, we require that annotation functions f appearing in annotations terms have a *finite generation* (see [2]). This technique has been used also in [31]. Essentially, if $C_\mathcal{P}$ are all the truth constants $c \in T$ appearing in $\mathcal{P}$, then f has a *finite generation* iff the *f-closure of $C_\mathcal{P}$*, i.e. the smallest set that contains $C_\mathcal{P}$ and is closed under f, is finite. For instance, $\min$, $\max$ and $1-x$ are such functions, while e.g. the product is not. As a consequence, if $\mathcal{P}$ is an a-program in which all annotation functions appearing in annotation terms have finite generation, then its grounding $\mathcal{P}^*$ is finite as well. Therefore, checking whether an a-interpretation I is a stable a-model of $\mathcal{P}$ is decidable.

4 Conclusions

We have defined annotated disjunctive logic programs, which are disjunctive logic programs where computable functions are used to annotate the atoms to manage truth degrees. We defined their syntax and semantics and have show cases for which deciding whether an a-interpretation I is a stable a-model of $\mathcal{P}$ is decidable. We also conjecture that in these cases an a-program can be mapped into classical disjunctive logic programs (so to inherit the result thereof [9, 12] [5]), allowing a fast prototyping of our framework. The intuition, given an annotated LPs $\mathcal{P}$, is to consider its grounded version $\mathcal{P}^*$. Then, let $C_\mathcal{P}$ be the set of all truth values appearing in $\mathcal{P}^*$. Each n-ary (ground) a-atom $P(t_1, \ldots, t_n)$:$\langle c_1, c_2\rangle$ of $\mathcal{P}^*$ is replaced with an $n+3$-ary propositional atom $P(t_1, \ldots, t_n, c_1, z, c_2)$, where z is variable and each atom has a different z. The intended meaning is that the degree of truth of $P(t_1, \ldots, t_n)$ is z and $z \in \langle c_1, c_2\rangle$. Then we need additional rules to model appropriately the intervals. For instance, for each (ground) a-atom A:$\langle c_1, c_2\rangle$, we consider the rule

$$z \in \langle c_1, c_2\rangle \leftarrow A(c_1, z, c_2) \ ,$$

where $z \in \langle c_1, c_2\rangle$ is a predicate (with obvious meaning) which can easily be defined as $C_\mathcal{P}$ is finite. We also add rules of the form

$$A(c_3, z, c_4) \leftarrow A(c_1, z, c_2), z \in \langle c_3, c_4\rangle$$

to propagate truth values among intervals. We do not further develop this approach due to lack of space. However, we conjecture that along this line we may obtain a stable model preserving classical disjunctive logic program. We will deserve to this topic more attention in a long version of this work.

Another objective is to extend the IB framework to disjunctive logic programs to a larger extend than [22]. Our starting point is [31] where arbitrary combination functions may be used in normal logic programs (recall that [22] restricts these to be special t-norms and s-norms).

[5] Of course, now the size of the complete truth lattice and the computational complexity of annotation functions in annotations terms are also parameters of the overall computational complexity.

References

[1] N. D. Belnap. A useful four-valued logic. In Gunnar Epstein and J. Michael Dunn, editors, *Modern uses of multiple-valued logic*, pages 5–37. Reidel, Dordrecht, NL, 1977.

[2] E. Böhler, C. Glasser, B. Schwarz, and K. Wagner. Generation problems. In *29th Int. Symp. on Mathematical Foundations of Computer Science (MFCS-04)*, LNCS 3153, pages 392–403. Springer, 2004.

[3] C. Chesnevar, G. Simari, T. Alsinet, and L. Godo. A logic programming framework for possibilistic argumentation with vague knowledge. In *Proc. of the 20th Annual Conf. on Uncertainty in Artificial Intelligence (UAI-04)*, pages 76–84, Arlington, Virginia, 2004. AUAI Press.

[4] C V. Damásio, J. Medina, and M. Ojeda Aciego. Sorted multi-adjoint logic programs: Termination results and applications. In *Proc. of the 9th European Conf. on Logics in Artificial Intelligence (JELIA-04)*, LNCS 3229, pages 252–265. Springer, 2004.

[5] C. V. Damásio, J. Medina, and M. Ojeda Aciego. A tabulation proof procedure for residuated logic programming. In *Proc. of the 6th European Conf. on Artificial Intelligence (ECAI-04)*, 2004.

[6] C. Viegas Damásio and L. M. Pereira. Antitonic logic programs. In *Proc. of the 6th European Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, LNCS 2173. Springer, 2001.

[7] C. V. Damásio and L. M. Pereira. Monotonic and residuated logic programs. In Salem Benferhat and Philippe Besnard, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 6th European Conf., (ECSQARU-01), 2001*, LNCS 2143, pages 748–759. Springer, 2001.

[8] C. V. Damásio and L. M. Pereira. Sorted monotonic logic programs and their embeddings. In *Proc. of the 10th Int. Conf. on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-04)*, pages 807–814, 2004.

[9] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[10] D. Dubois, J. Lang, and H. Prade. Towards possibilistic logic programming. In *Proc. of the 8th Int. Conf. on Logic Programming (ICLP-91)*, pages 581–595. The MIT Press, 1991.

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

[12] T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, 1995.

[13] M. C. Fitting. Fixpoint semantics for logic programming - a survey. *Theoretical Computer Science*, 21(3):25–51, 2002.

[14] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[15] M. Kifer and V.S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12:335–367, 1992.

[16] Laks V.S. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554–570, 2001.

[17] Y. Loyer and U. Straccia. The approximate well-founded semantics for logic programs with uncertainty. In *28th Int. Symp. on Mathematical Foundations of Computer Science (MFCS-2003)*, LNCS 2747, pages 541–550, 2003. Springer.

[18] Y. Loyer and U. Straccia. Default knowledge in logic programs with uncertainty. In *Proc. of the 19th Int. Conf. on Logic Programming (ICLP-03)*, LNCS 2916, pages 466–480, 2003. Springer.

[19] Y. Loyer and U. Straccia. Epistemic foundation of the well-founded semantics over bilattices. In *29th Int. Symp. on Mathematical Foundations of Computer Science (MFCS-2004)*, LNCS 3153, pages 513–524, 2004. Springer.

[20] Y. Loyer and U. Straccia. Any-world assumptions in logic programming. *Theoretical Computer Science*, 342(2-3):351–381, 2005.

[21] T. Lukasiewicz. Probabilistic logic programming. In *Proc. of the 13th European Conf. on Artificial Intelligence (ECAI-98)*, pages 388–392, 1998.

[22] C. Mateis. Extending disjunctive logic programming by t-norms. In *Proc. of the 5th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR-99)*, LNCS 1730, pages 290–304. Springer, 1999.

[23] C. Mateis. Quantitative disjunctive logic programming: Semantics and computation. *AI Communications*, 13:225–248, 2000.

[24] J. Medina and M. Ojeda-Aciego. Multi-adjoint logic programming. In *Proc. of the 10th Int. Conf. on Information Processing and Managment of Uncertainty in Knowledge-Based Systems, (IPMU-04)*, pages 823–830, 2004.

[25] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. In *Proc. of the 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, LNAI 2173, pages 351–364. Springer, 2001.

[26] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy sets and systems*, 1(146):43–62, 2004.

[27] R. Ng and V.S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1993.

[28] R. Ng and V.S. Subrahmanian. Stable model semantics for probabilistic deductive databases. *Information and Computation*, 110(1):42–83, 1994.

[29] P. Nicolas, L. Garcia, and I. Stéphan. Possibilistic stable models. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, pages 248–253. Morgan Kaufmann Publishers, 2005.

[30] E. Y. Shapiro. Logic programs with uncertainties: A tool for implementing rule-based systems. In *Proc. of the 8th Int. Joint Conf. on Artificial Intelligence (IJCAI-83)*, pages 529–532, 1983.

[31] U. Straccia. Query answering in normal logic programs under uncertainty. In *8th European Conf.s on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-05)*, LNCS 3571, pages 687–700, 2005. Springer.

[32] M.H. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 4(1):37–53, 1986.

[33] P. Vojtáš. Fuzzy logic programming. *Fuzzy Sets and Systems*, 124:361–370, 2004.