Trichotomy Results on the Complexity of Reasoning with Disjunctive Logic Programs

Mirosław Truszczyński

Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA

Abstract. We present trichotomy results characterizing the complexity of reasoning with disjunctive logic programs. To this end, we introduce a certain definition schema for classes of programs based on a set of allowed arities of rules. We show that each such class of programs has a finite representation, and for each of the classes definable in the schema we characterize the complexity of the existence of an answer set problem. Next, we derive similar characterizations of the complexity of skeptical and credulous reasoning with disjunctive logic programs. Such results are of potential interest. On the one hand, they reveal some reasons responsible for the hardness of computing answer sets. On the other hand, they identify classes of problem instances, for which the problem is "easy" (in P) or "easier than in general" (in NP).

1 Introduction

It is well known that the problem to decide whether a disjunctive logic program has an answer set (the *EAS problem*, for short) is Σ_2^P-complete [1]. It is also well known that putting restrictions on the input instances may affect the complexity. For example, the EAS problem for normal logic programs is NP-complete [2].

In this paper we study the complexity of the EAS problem for classes of disjunctive logic programs that can be defined by sets of program rule *arities*. We show that in each case the problem is either in P, is NP-complete or is Σ_2^P-complete, and we characterize the classes of programs that fall into each category. We extend this result to establish similar characterizations for the problems of skeptical and credulous reasoning with disjunctive logic programs. Such results are of potential interest. On the one hand, they reveal some reasons responsible for the hardness of computing answer sets; cf. Lemmas 4 and 5. On the other hand, they identify classes of problem instances, for which the problem is "easy" (in P) or "easier than in general" (in NP); cf. Lemmas 1 and 2.

Our results can be regarded as *trichotomy* results for the complexity of reasoning tasks in disjunctive logic programming. Similar results are known for the complexity of reasoning in other formalisms: propositional satisfiability [3–5], reasoning with minimal models [6], default logic [7], and abductive reasoning [8]. There is however, an important distinction between those earlier papers and our approach. The results contained there are concerned with the setting in which formulas are conjunctions of Boolean relations, and the set of models of a formula is the intersection of the sets of models of its constituent relations (that, in particular, implies the monotonicity of inference from such formulas). The basic results concern the complexity of the satisfiability problem

for classes of formulas determined by sets of Boolean relation allowed as formula conjuncts. It turns out that there is a simple characterization of all those classes, for which the problem is in P; moreover for all other classes the problem is NP-complete [3–5]. This result can be exploited to characterize the complexity of reasoning with systems, in which basic reasoning tasks reduce to series of satisfiability tests [6–8]. In the setting of disjunctive logic programs, these earlier results seem of little help. It is well known that the answer-set semantics is *nonmonotone* and so, logic programs under the answer-set semantics are *not* conjunctions of their rules. Thus, it is unclear that defining classes of programs in terms of semantic properties of individual rules could yield any useful insights.

2 Preliminaries

We fix an infinite countable set At of propositional variables. A *disjunctive program* (or simply, program) over the set of atoms At is a collection of *disjunctive logic program rules*, that is, expressions of the form

$$r = a_1 | \ldots | a_k \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n, \tag{1}$$

where a_i, b_i and c_i are atoms from At. The disjunction $a_1 | \ldots | a_k$ is the *head* of r and the conjunction $b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n$ is the *body* of r. Further, we call the triple $[k, m, n]$ the *arity* of r. We denote the arity of r by $a(r)$ and write $a_1(r)$, $a_2(r)$ and $a_3(r)$ for the three components of $a(r)$. Thus, under this notation, $a(r) = [a_1(r), a_2(r), a_3(r)]$. If $a_1(r) \geq 1$, we call r *proper*. Otherwise, $a_1(r) = 0$ and r is a *constraint*. Programs consisting of proper rules are *proper* programs. Similarly, programs consisting of constraints only are *constraint* programs.

We recall that given a program P and a set of atoms M (an interpretation), the *reduct* of P with respect to M, P^M, is the program obtained by removing all rules with a literal $not\ c$, where $c \in M$, in the body and, then, removing all negative literals (negated atoms) from the bodies of all remaining rules. A set of atoms M is an *answer set* of a disjunctive program P if M is a minimal model of P^M [9].

For a program P, we denote by $\overline{P}$ and $\overline{\overline{P}}$ the programs consisting of all proper rules and of all constraints in P, respectively. The following result is well known.

Theorem 1. *A set $M \subseteq At$ is an answer set of a program P if and only if M is an answer set of $\overline{P}$ and a model of $\overline{\overline{P}}$.*

One can define classes of logic programs by specifying arities of rules. For instance, the set $\{[1, 1, 0], [0, 1, 0]\}$ defines the set of all Horn programs such that each rule has at most one atom in the body. Some classes of programs do not have such a finitary representation. For instance, the class of all Horn programs with constraints can be defined by the set $\{[k, m, 0] \mid k \leq 1, 0 \leq m\}$, but there is no *finite* set of arities that could be used instead. To handle such cases, we introduce now a general representation schema for classes of programs defined by sets of arities.

Let $U = \{0, 1, \ldots\} \cup \{\infty\}$. We consider U to be ordered by the relation $\leq$ (the standard $\leq$ ordering relation on non-negative integers, extended by $i \leq \infty$, for every

$i = 0, 1 \ldots$). Next, we define $\mathcal{T} = \{[k, m, n] \mid k, m, n \in U\}$. Thus, $\mathcal{T}$ contains all arities, as well as *additional* triples — those containing at least one occurrence of ∞. We refer to triples of that latter sort as *superarities*. We emphasize that superarities are *not* arities as we do not consider infinitary rules. If $\alpha \in \mathcal{T}$, we write α_1, α_2 and α_3 for the components of α.

Let $\alpha, \beta \in \mathcal{T}$. We define $\alpha \preceq \beta$ if (1) $\alpha_i \leq \beta_i$, for $i = 1, 2, 3$ and, (2) if $\alpha_1 = 0$ then $\beta_1 = 0$. We write $\alpha \prec \beta$ when $\alpha \preceq \beta$ and $\alpha \neq \beta$.

If $\Delta \subseteq \mathcal{T}$, then we define $\mathcal{F}(\Delta)$ to be the set of all *finite* programs P that satisfy the following condition: for every rule $r \in P$ there is $\alpha \in \Delta$ such that $a(r) \preceq \alpha$. The condition (2) in the definition of $\preceq$ allows us to distinguish between classes of proper programs and classes of programs with constraints. Indeed, without the condition (2), every class of programs of the form $\mathcal{F}(\Delta)$ would contain constraints. With the condition (2), we can specify constraint-free classes of programs by means of sets Δ such that for every $\alpha \in \Delta$, $\alpha_1 \geq 1$. Including in Δ elements α with $\alpha_1 = 0$ yields classes of programs with constraints. As there are classes of proper programs that are of interest (Horn programs and normal logic programs are typically defined as consisting of proper rules only), the distinction is needed and motivates the condition (2) in the definition of $\preceq$.

Using this schema we can define several important classes of programs. For instance, the class of proper Horn programs can be described as $\mathcal{F}(\{[1, \infty, 0]\})$ and the class of normal logic programs with constraints as $\mathcal{F}(\{[1, \infty, \infty], [0, \infty, \infty]\})$.

Our main goal in this paper is to determine the complexity of the EAS problem when input programs come from classes $\mathcal{F}(\Delta)$, for $\Delta \subseteq \mathcal{T}$.

3 The Case of Finite Δ

In this section we tackle the case when Δ is finite. We note that given a finite set $\Delta \subseteq \mathcal{T}$, the problem to decide the membership of a program in the class $\mathcal{F}(\Delta)$ is in P.

We start by establishing the upper bounds on the complexity of the EAS problem for classes $\mathcal{F}(\Delta)$ given by some particular finite sets Δ of arities. Our first result is concerned with the following classes: $\mathcal{F}(\{[\infty, \infty, 0]\})$ — the class of proper positive disjunctive programs; $\mathcal{F}(\{[1, \infty, 0], [0, \infty, \infty]\})$ — the class of programs whose every rule is either a proper Horn rule or a constraint; $\mathcal{F}(\{[\infty, 1, 0],\ [0, 1, 0]\})$ — the class of *dual Horn programs*, that is, programs whose every rule, when viewed as a propositional clause, is a dual Horn clause; and $\mathcal{F}(\{[i, j, 0] \in \mathcal{T} \mid i + j \leq 2\})$ — the class of positive programs whose every rule consists of at most two literals. For each of these classes of programs, the EAS problem is easy (that is, in P).

Lemma 1. *If Δ is one of:*

1. $\{[\infty, \infty, 0], [0, 0, 0]\}$
2. $\{[1, \infty, 0], [0, \infty, \infty]\}$
3. $\{[\infty, 1, 0], [0, 1, 0]\}$
4. $\{[i, j, 0] \in \mathcal{T} \mid i + j \leq 2\}$

then the EAS problem for $\mathcal{F}(\Delta)$ is in P.

Proof: We omit the proofs of the first two statements, which are quite straightforward.

Thus, let us assume that $P \in \mathcal{F}(\{[\infty, 1, 0], [0, 1, 0]\})$ or $P \in \mathcal{F}(\{[i, j, 0] \in \mathcal{T} \mid i + j \leq 2\})$. Then P is a dual Horn program, or P is positive and every clause in P consists of two literals. In each case, one can decide in polynomial time whether P has a model. If the answer is "no," then P has no answer sets. Otherwise, P has a model, say M. Since M is a model of $\overline{P}$, there is a subset M' of M such that M' is a minimal model of $\overline{P}$. We have $\overline{P}^{M'} = \overline{P}$. Thus, M' is an answer set of $\overline{P}$. Since M satisfies $\overline{\overline{P}}$ and each rule in $\overline{\overline{P}}$ is of the form $\leftarrow a$ or $\leftarrow a, b$, M' satisfies $\overline{\overline{P}}$, too. Thus, M' is an answer set of P. Again, the assertion follows. □

The second result establishes sufficient conditions for the EAS problem to be in the class NP. It turns out to be the case for the following three classes of programs: $\mathcal{F}(\{[1, \infty, \infty], [0, \infty, \infty]\})$ — the class of normal logic programs with constraints; $\mathcal{F}(\{[\infty, 1, \infty], [0, \infty, \infty]\})$ — the class of programs whose reducts consist of proper dual Horn rules and constraints; and $\mathcal{F}(\{[\infty, \infty, 0], [0, \infty, 0]\})$ — the class of positive programs.

Lemma 2. *If Δ is one of:*

1. $\{[1, \infty, \infty], [0, \infty, \infty]\}$
2. $\{[\infty, 1, \infty], [0, \infty, \infty]\}$
3. $\{[\infty, \infty, 0], [0, \infty, 0]\}$

then the EAS problem for $\mathcal{F}(\Delta)$ is in NP.

Proof: If $\Delta = \{[1, \infty, \infty], [0, \infty, \infty]\}$, $\mathcal{F}(\Delta)$ consists of normal logic programs with constraints. In this case, the result is well known [2].

Next, let $\Delta = \{[\infty, 1, \infty], [0, \infty, \infty]\}$ and $P \in \mathcal{F}(\Delta)$. To prove the assertion it is enough to show that there is a polynomial time algorithm for deciding whether a set of atoms $M \subseteq At(P)$ is an answer set of P. To this end, we note that M is a minimal model of $\overline{P}^M$ if and only if for every $a \in M$, the program $\overline{P}^M \cup \{ \leftarrow a\} \cup \{ \leftarrow b \mid b \in At(P) \setminus M\}$ does not have a model. Since $\overline{P}^M \cup \{ \leftarrow a\} \cup \{ \leftarrow b \mid b \in At(P) \setminus M\}$ is dual Horn, verifying whether M is a minimal model of $\overline{P}^M$ can be accomplished in polynomial time. In addition, checking that M is a model of $\overline{\overline{P}}$ can also be done in polynomial time, too. Thus, in this case, the assertion follows.

Finally, if $\Delta = \{[\infty, \infty, 0], [0, \infty, 0]\}$ and $P \in \mathcal{F}(\Delta)$, then deciding whether P has an answer set is equivalent to deciding whether P has a model. Since the problem to decide whether P has a model is in NP, the assertion follows. □

Next, we will prove several lower-bound results. We will first exhibit classes of programs of the form $\mathcal{F}(\Delta)$ for which the EAS problem is NP-hard. To this end, we need a lemma establishing the NP-hardness of the SAT problem for some simple classes of CNF theories. For the most part, the result is folklore. We sketch an argument for the sake of completeness.

Lemma 3. *The SAT problem restricted to each of the following classes of CNF theories is NP-hard:*

1. *the class of all CNF formulas ψ such that each clause of ψ is a disjunction of two negated atoms, or of at most three atoms*

2. *the class of all CNF formulas ψ such that each clause of ψ consists of at most two literals, or is a disjunction of two atoms and one negated atom*
3. *the class of all CNF formulas ψ such that each clause of ψ consists of at most two atoms, or of one negated atom, or is a disjunction of an atom and two negated atoms*
4. *the class of all CNF formulas ψ such that each clause of ψ is a disjunction of two atoms, or of at most three negated atoms.*

Proof: We will only prove the case (3). The argument in all other cases is similar.

Let φ be a CNF formula whose every clause has three literals, and let X be a set of atoms occurring in φ. For each atom $z \in X$ we introduce a fresh atom z'. Next, in each clause c we replace some of its positive literals a with $\neg a'$, and some of its negative literals $\neg b$ with b' so that the resulting clause, we will denote it by $\hat{c}$, is the disjunction of exactly one atom and two negated atoms. Such replacements can always be found.

Finally, we introduce one more fresh atom, say f, and define $F(\varphi)$ as follows:

$$F(\varphi) = \{z \vee z' \mid z \in X\} \cup \{f \vee \neg z \vee \neg z' \mid z \in X\} \cup \{\neg f\} \cup \{\hat{c} \mid c \text{ is a clause in } \varphi\}$$

It is evident that $F(\varphi)$ is in the class of theories under consideration.

We will show that φ has a model if and only if $F(\varphi)$ has a model. To this end, we note that models of $F(\varphi)$ (if exist) are of the form $M \cup \{z' \mid z \in X \setminus M\}$, where $M \subseteq X$. It is now easy to see that M is a model of φ if and only if $M \cup \{z' \mid z \in X \setminus M\}$ is a model of $F(\varphi)$. Thus, the claim and, consequently, the assertion, follows.

As we noted, the argument for the remaining classes is similar. We only need to change the definition of $\hat{c}$ and use clauses $\neg z \vee \neg z'$ instead of $f \vee \neg z \vee \neg z'$ (there is no need to introduce f, as clauses being the disjunctions of two negated atoms are allowed in formulas in each of the classes considered in (1), (2) and (4)). □

We will now use Lemma 3 to establish the NP-hardness of the EAS problem for $\mathcal{F}(\Delta)$ for several simple sets $\Delta \subseteq \mathcal{T}$.

Lemma 4. *If Δ is any of:*

1. $\{[1, 0, 1]\}$
2. $\{[2, 0, 0], [0, 0, 1]\}$
3. $\{[3, 0, 0], [0, 2, 0]\}$
4. $\{[2, 1, 0], [0, 2, 0]\}$
5. $\{[2, 0, 0], [1, 2, 0], [0, 1, 0]\}$
6. $\{[2, 0, 0], [0, 3, 0]\}$

then the EAS problem for $\mathcal{F}(\Delta)$ is NP-hard.

Proof (sketch): (1) The proof of the NP-completeness of the EAS problem for normal logic programs given by Marek and Truszczyński [2] establishes the assertion (1).

(2) We will construct a reduction from the SAT problem concerning the class considered in Lemma 3(4). Let φ be a CNF of the appropriate form. We denote by $pos(\varphi)$ the set of all clauses in φ that are of the form $a \vee b$, where $a, b \in At$. We denote by $neg(\varphi)$ the

set of all remaining clauses in φ (all of them are disjunctions of at most three negative literals).

For every clause $c = \neg y_1 \vee \ldots \vee \neg y_k$ in $neg(\varphi)$, we introduce a fresh atom x_c. Next, we define

$$\begin{aligned}
P(\varphi) &= \{a|b \leftarrow \mid a \vee b \in pos(\varphi)\} \cup \\
&\quad \{x_c|y_i \leftarrow \mid c \in neg(\varphi), \\
&\qquad\qquad c = \neg y_1 \vee \ldots \vee \neg y_k, 1 \leq i \leq k\}, \\
Q(\varphi) &= \{\leftarrow not\ x_c \mid c \in neg(\varphi)\}, \text{ and} \\
R(\varphi) &= P(\varphi) \cup Q(\varphi).
\end{aligned}$$

One can now show that φ is satisfiable if and only if $R(\varphi)$ has an answer set (we omit the details). Since $R(\varphi) \in \mathcal{F}(\Delta)$, the assertion follows by Lemma 3(4).

(3)-(6) In all the remaining cases, we exploit the fact that $P \in \mathcal{F}(\Delta)$ has an answer set if and only if P has a model (the same argument that we used in the proof of Lemma 2 applies). The latter problem for each of the cases (3)-(6) is equivalent to the satisfiability problem for the classes considered in Lemma 3(1)-(4), respectively. In each of these cases the problem is NP-hard (Lemma 3), and so the assertion follows. □

The next lemma establishes conditions guaranteeing Σ_2^P-hardness of the EAS problem. Eiter and Gottlob [1] proved that given $P \in \mathcal{F}(\{[2,0,0],[1,3,0],[1,0,1]\})$, it is Σ_2^P-hard to decide whether P has an answer set. The proof can be modified to the case when the class of input programs is restricted to $\mathcal{F}(\{[2,0,0],[1,2,0],[1,0,1]\})$, as clauses of the arity $[1,3,0]$ can be simulated by clauses of arity $[1,2,0]$. Moreover, in the construction provided by Eiter and Gottlob, the only rule of the arity $[1,0,1]$ used is a constraint and it can be simulated by a rule of the arity $[0,0,1]$. Thus, the Σ_2^P-hardness holds also for the class $\mathcal{F}(\{[2,0,0],[1,2,0],[0,0,1]\})$ of programs. We omit the details and state the result only.

Lemma 5. *If Δ is any of:*

1. $\{[2,0,0],[1,2,0],[0,0,1]\}$
2. $\{[2,0,0],[1,2,0],[1,0,1]\}$

then the EAS problem for $\mathcal{F}(\Delta)$ is Σ_2^P-hard.

We will now derive the main result of this section. It provides a complete characterization of the complexity of the EAS problem for the class $\mathcal{F}(\Delta)$. To state the result we introduce one more piece of notation. Given $\Delta, \Theta \subseteq \mathcal{T}$, we write $\Delta \preceq \Theta$ if for every $\alpha \in \Delta$ there is $\beta \in \Theta$ such that $\alpha \preceq \beta$.

Theorem 2. *Let $\Delta \subseteq \mathcal{T}$ be finite.*

(A) If

1. $\Delta \preceq \{[\infty,\infty,0],[0,0,0]\}$, *or*
2. $\Delta \preceq \{[1,\infty,0],[0,\infty,\infty]\}$, *or*
3. $\Delta \preceq \{[\infty,1,0],[0,1,0]\}$, *or*
4. $\Delta \preceq \{[i,j,0] \in \mathcal{T} \mid i+j \leq 2\}$,

then the EAS problem for $\mathcal{F}(\Delta)$ is in P.

(B) Otherwise, if

1. $\Delta \preceq \{[1,\infty,\infty],[0,\infty,\infty]\}$, *or*
2. $\Delta \preceq \{[\infty,1,\infty],[0,\infty,\infty]\}$, *or*
3. $\Delta \preceq \{[\infty,\infty,0],[0,\infty,0]\}$,

then the EAS problem for $\mathcal{F}(\Delta)$ is NP*-complete.*

(C) Otherwise, the EAS problem for $\mathcal{F}(\Delta)$ is Σ_2^P-complete.

Proof: The claim (A) follows directly from Lemma 1. Thus, let us assume that Δ does not fall under the scope of (A) and satisfies the assumptions of (B). By Lemma 2, the latter implies that the EAS problem for $\mathcal{F}(\Delta)$ is in NP.

If $\{[1,0,1]\} \preceq \Delta$ or $\{[2,0,0],[0,0,1]\} \preceq \Delta$, the NP-hardness of the EAS problem for $\mathcal{F}(\Delta)$ follows from Lemma 4, parts (1) and (2), respectively. Thus, let us assume that $\{[1,0,1]\} \not\preceq \Delta$ and $\{[2,0,0],[0,0,1]\} \not\preceq \Delta$.

Since $\{[1,0,1]\} \not\preceq \Delta$, we have $\Delta \preceq \{[\infty,\infty,0],\ [0,\infty,\infty]\}$. Since Δ does not satisfy the condition (A2), $\{[2,0,0]\} \preceq \Delta$. Since $\{[2,0,0],[0,0,1]\} \not\preceq \Delta$, $\{[0,0,1]\} \not\preceq \Delta$. Thus, $\Delta \preceq \{[\infty,\infty,0],[0,\infty,0]\}$.

Since Δ does not satisfy the condition (A1), $\{[0,1,0]\} \preceq \Delta$. Similarly, since Δ does not satisfy the condition (A3), $\{[1,2,0]\} \preceq \Delta$ or $\{[0,2,0]\} \preceq \Delta$. We also have that Δ does not satisfy the condition (A4). Thus, there is $\alpha \in \Delta$ such that $\alpha_1 + \alpha_2 \geq 3$. Since we already proved that $\{[2,0,0]\} \preceq \Delta$, it follows that at least one of the following conditions holds: $\{[3,0,0],[0,2,0]\} \preceq \Delta$, $\{[2,1,0],[0,2,0]\} \preceq \Delta$, $\{[2,0,0],[1,2,0],[0,1,0]\} \preceq \Delta$, or $\{[2,0,0],[0,3,0]\} \preceq \Delta$. Thus, the NP-hardness of the EAS problem for $\mathcal{F}(\Delta)$ follows again from Lemma 4 and completes the proof of (B).

To prove (C), we observe that if Δ does not fall under the scope of (B), then $\{[2,0,0],[1,2,0],[0,0,1]\} \preceq \Delta$. Indeed, since Δ does not satisfy (B1), $\{[2,0,0]\} \preceq \Delta$. Similarly, since Δ does not satisfy (B2), $\{[1,2,0]\} \preceq \Delta$. Finally, since Δ does not satisfy (B3), $\{[0,0,1]\} \preceq \Delta$ or $\{[1,0,1]\} \preceq \Delta$. Thus, the Σ_2^P-hardness follows by Lemma 5. Since the EAS problem is in Σ_2^P even without any restrictions on the class of programs, both (C) and the assertion of the lemma follows. □

4 The Case of Infinite Δ

The question we study now is whether there are interesting classes of programs of the form $\mathcal{F}(\Delta)$, when Δ is infinite. The main result of this section is that by allowing Δ to be infinite, we do not obtain *any* new classes of programs. In other words, for every class of programs of the form $\mathcal{F}(\Delta)$ there is a finite set $\Delta' \subseteq \mathcal{T}$ such that $\mathcal{F}(\Delta) = \mathcal{F}(\Delta')$.

A sequence $\{\alpha^k\}_{k=1}^{\infty}$ is *monotone* (*strictly* monotone) if for every k, $\alpha^k \preceq \alpha^{k+1}$ ($\alpha^k \prec \alpha^{k+1}$, respectively). Let $\{\alpha^k\}_{k=1}^{\infty}$ be a *monotone* sequence of elements of $\mathcal{T}$. We define the *limit* of this sequence as $\alpha^{\infty} = [(\alpha^{\infty})_1, (\alpha^{\infty})_2, (\alpha^{\infty})_3]$, where $(\alpha^{\infty})_i = \sup\{(\alpha^k)_i \mid k = 1, 2 \ldots, \}$, for $i = 1, 2, 3$. Let $\Delta \subseteq \mathcal{T}$. A monotone sequence $\{\alpha^k\}_{k=1}^{\infty}$ of elements of Δ is *maximal* if there is *no* $\alpha \in \Delta$ such that $\alpha^{\infty} \prec \alpha$. We define $A(\Delta)$ to be the set of the limits of maximal sequences in Δ.

We have the following two lemmas (we omit the proof of the first one as it is evident).

Lemma 6. *Let $\{\beta^k\}_{k=1}^{\infty}$ be a strictly monotone sequence of elements from $\mathcal{T}$. For every $\alpha \in \mathcal{T}$, if $\alpha \prec \beta^{\infty}$, then there is k such that $\alpha \prec \beta^k$.*

Lemma 7. *Let $\Delta \subseteq \mathcal{T}$ and $\alpha \in \Delta$. Then, there is $\alpha' \in A(\Delta)$ such that $\alpha \preceq \alpha'$.*

Proof: Let $X = \{\beta \in \Delta \mid \alpha \preceq \beta\}$. If X has a maximal element, say γ, a sequence with each term equal to γ is maximal. Its limit, also equal to γ, clearly satisfies $\gamma \in A(\Delta)$ and $\alpha \preceq \gamma$. Thus, the assertion follows.

Otherwise, X has no maximal elements. Let α^1 be any element in X (we note that $X \neq \emptyset$, as $\alpha \in X$). Let $k \geq 1$ and let $\langle \alpha^1, \ldots, \alpha^k \rangle$ be a strictly monotone sequence of k elements in X, for some $k \geq 1$. Since X has no maximal elements, X contains elements that are strictly greater than α^k. Let us select as α^{k+1} an element $\beta \in X$ such that $\alpha^k \prec \beta$ and $\alpha^k_i < \beta_i$ on as many positions $i = 1, 2, 3$ as possible. An infinite sequence we define in that way, we will denote it by α, is strictly monotone. Let us assume that there is $\beta \in \Delta$ such that $\alpha^{\infty} \prec \beta$. It follows that there is j, $1 \leq j \leq 3$, such that $(\alpha^{\infty})_j < \beta_j$. Thus, $(\alpha^{\infty})_j = m$, for some integer m, and there is n such that $(\alpha^n)_j = m$. Since $\alpha^{n+1} \preceq \beta$ and $(\alpha^n) = (\alpha^{n+1})_j = m$, the number of positions i such that $(\alpha^n)_i < (\alpha^{n+1})_i$ is strictly smaller than the number of positions i such that $(\alpha^n)_i < \beta_i$. Since $\beta \in X$, that contradicts the way we constructed the sequence α.

It follows that the sequence $\{\alpha^k\}_{k=1}^{\infty}$ is maximal for Δ and so, the assertion follows in this case, too. □

We now have the following properties. We provide a proof for the first of them and omit proofs, rather direct and technical, of the remaining two.

Proposition 1. *For every $\Delta \subseteq \mathcal{T}$, $\mathcal{F}(\Delta) = \mathcal{F}(A(\Delta))$.*

Proof: To prove the assertion, it is enough to show that for every *arity* α (no occurrence of ∞), $\{\alpha\} \preceq \Delta$ if and only if $\{\alpha\} \preceq A(\Delta)$. Let us first assume that $\{\alpha\} \preceq \Delta$. It follows that there is an element $\alpha' \in \Delta$ such that $\alpha \preceq \alpha'$. By Lemma 7, there is $\alpha'' \in A(\Delta)$ such that $\alpha' \preceq \alpha''$. Thus, $\alpha \preceq \alpha''$ and so, $\{\alpha\} \preceq A(\Delta)$.

Conversely, let $\{\alpha\} \preceq A(\Delta)$. It follows that there is $\beta \in A(\Delta)$ such that $\alpha \preceq \beta$. Since $\beta \in A(\Delta)$, there is a monotone sequence $\{\beta^k\}_{k=1}^{\infty}$ of elements of Δ such that its limit is β. Wlog we can assume that either starting with some k_0, the sequence $\{\beta^k\}_{k=1}^{\infty}$ is constant, or the sequence β is strictly monotone. In the first case, $\alpha \preceq \beta = \beta^{k_0}$. Since $\beta^{k_0} \in \Delta$, $\{\alpha\} \preceq \Delta$. In the second case, Lemma 6 implies that there is k such that $\alpha \prec \beta^k$, and again $\{\alpha\} \preceq \Delta$ follows. □

Proposition 2. *For every Δ, $A(\Delta)$ is an antichain.*

Proposition 3. *Every antichain in the partially ordered set $\langle \mathcal{T}, \preceq \rangle$ is finite.*

These properties imply the main result of this section. It asserts that every class of programs $\mathcal{F}(\Delta)$ can be defined by means of a finite set Δ' that is an antichain in $\langle \mathcal{T}, \preceq \rangle$.

Theorem 3. *For every set $\Delta \subseteq \mathcal{T}$ there is a finite subset $\Delta' \subseteq \mathcal{T}$ such that Δ' is an antichain and $\mathcal{F}(\Delta) = \mathcal{F}(\Delta')$.*

Proof: Let us define $\Delta' = A(\Delta)$. By Propositions 2 and 3, Δ' is a finite antichain in $\langle \mathcal{T}, \preceq \rangle$, and by Proposition 1, $\mathcal{F}(\Delta) = \mathcal{F}(\Delta')$. Thus, the theorem follows. □

5 The Complexity of Skeptical and Credulous Reasoning

The EAS problem is just one example of a reasoning task that arises in the context of disjunctive logic programs with the answer-set semantics. There are several other tasks that are of interest, too. They concern deciding whether a program nonmonotonically entails a literal, that is an atom, say a, or its negation $\neg a$. For a disjunctive logic program P and a literal l we say that

1. P *skeptically entails* l, written $P \models_s l$, if $M \models l$, for every answer set M of P (we recall that if M is a model (a set of atoms) and a is an atom, $M \models a$ if $a \in M$, and $M \models \neg a$ if $a \notin M$)
2. P *credulously entails* l, written $P \models_c l$, if there is an answer set M of P such that $M \models l$.

We note that $P \models_s l$ if and only if $P \not\models_c \bar{l}$, where $\bar{l}$ is l's *dual* literal. Thus, to establish fully the complexity of deciding nonmonotonic entailment it is enough to focus on deciding whether $P \models_c \neg a$ and $P \models_s \neg a$, where a is an atom. These two decision tasks were studied by Eiter and Gottlob [1], who proved that, in general, the first one is Σ_2^P-complete and the second one is Π_2^P-complete.

Reasoning with answer sets is related to circumscription and closed-world reasoning with propositional theories. A detailed study of the complexity of those forms of reasoning was conducted by Cadoli and Lenzerini [10]. Using Theorem 2 and one of the results from that paper (which we state in the proof below), one can characterize in terms of our definition schema the complexity of deciding, given a program P and an atom a, whether $P \models_c \neg a$ and $P \models_s \neg a$. The two problems are addressed in the following two theorems.

Theorem 4. *Let $\Delta \subseteq \mathcal{T}$ be finite.*

(A) If

1. $\Delta \preceq \{[1, \infty, 0], [0, \infty, \infty]\}$, *or*
2. $\Delta \preceq \{[\infty, 1, 0], [0, 1, 0]\}$, *or*
3. $\Delta \preceq \{[i, j, 0] \in \mathcal{T} \mid i + j \leq 2\}$,

then the problem to decide whether $P \models_c \neg a$, where $P \in \mathcal{F}(\Delta)$ and a is an atom, is in P.

(B) Otherwise, if

1. $\Delta \preceq \{[1, \infty, \infty], [0, \infty, \infty]\}$, *or*
2. $\Delta \preceq \{[\infty, 1, \infty], [0, \infty, \infty]\}$, *or*
3. $\Delta \preceq \{[\infty, \infty, 0], [0, \infty, 0]\}$,

then the problem to decide whether $P \models_c \neg a$, where $P \in \mathcal{F}(\Delta)$ and a is an atom, is NP*-complete.*

(C) Otherwise, the problem to decide whether $P \models_c \neg a$, where $P \in \mathcal{F}(\Delta)$ and a is an atom, is Σ_P^2-complete.

Proof: It is well known that P has an answer set M such that $M \models \neg a$ (that is, $a \notin M$) if and only if $P \cup \{ \leftarrow a\}$ has an answer set. Let $\Delta \subseteq \mathcal{T}$ be finite and let us define $\Delta' = \Delta \cup \{[0,1,0]\}$. Clearly, if $P \in \mathcal{F}(\Delta)$, then $P \cup \{ \leftarrow a\} \in \mathcal{F}(\Delta')$. Moreover, if Δ falls under the scope of (A) ((A) or (B), respectively) of this theorem then Δ' falls under the scope of (A) ((A) or (B), respectively) of Theorem 2. Consequently, the upper bound follows by Theorem 2.

The proof of hardness exploits earlier results on the hardness of the EAS problem. The reductions are provided by the following constructions. For a program P we define $P' = P \cup \{a|b\}$, $P'' = P \cup \{a \leftarrow not\ b; b \leftarrow not\ a\}$, and $P''' = \overline{P} \cup \{a \leftarrow bd(r) \mid r \in \overline{\overline{P}}\}$, where a and b are fresh atoms. Clearly, P has an answer set if and only if $P' \models_c \neg a$ ($P'' \models_c \neg a$, $P''' \models_c \neg a$, respectively). We omit the details due to space limits. □.

Theorem 5. *Let $\Delta \subseteq \mathcal{T}$ be finite.*

(A) If $\Delta \preceq \{[1,\infty,0],[0,\infty,\infty]\}$, then the problem to decide whether $P \models_s \neg a$, where $P \in \mathcal{F}(\Delta)$ and a is an atom, is in P.

(B) Otherwise, if

1. *$\Delta \preceq \{[1,\infty,\infty],[0,\infty,\infty]\}$, or*
2. *$\Delta \preceq \{[\infty,1,\infty],[0,\infty,\infty]\}$*

then the problem to decide whether $P \models_s \neg a$, where $P \in \mathcal{F}(\Delta)$ and a is an atom, is coNP*-complete.*

(C) Otherwise, the problem to decide whether $P \models_s \neg a$, where $P \in \mathcal{F}(\Delta)$ and a is an atom, is Π_2^P-complete.

Proof: It is well known that P has an answer set such that $M \not\models \neg a$ (that is, $a \in M$) if and only if $P \cup \{ \leftarrow not\ a\}$ has an answer set. That observation implies all upper bound results (by a similar argument as that used in the proof of the previous theorem).

We will now prove the lower bounds for the cases (B) and (C). Let us assume that Δ does not satisfy (A) but falls under the scope of (B). If Δ satisfies (B1), then $\{[1,0,1]\} \preceq \Delta$. Let $P \in \mathcal{F}(\Delta)$ and let a and a' be fresh atoms. We note that P has an answer set if and only if $P \cup \{a \leftarrow not\ a'\}$ has an answer set M such that $a \in M$ or, equivalently, if and only if $P \cup \{a \leftarrow not\ a'\} \not\models_s \neg a$. Thus, the hardness follows (cf. Lemma 4(1)).

Let us assume then that Δ does not satisfy (B1). Then, we have that $\{[2,0,0]\} \preceq \Delta$. It follows from the results of Cadoli and Lenzerini [10] that it is NP-complete to decide whether a given 2CNF theory, whose every clause is a disjunction of two atoms, has a minimal model that contains a given atom a. As minimal models of such theories are precisely answer sets of the corresponding disjunctive program, it follows that given a program $P \in \mathcal{F}(\Delta)$ and an atom a, it is coNP-complete to decide whether $P \models_s \neg a$.

Finally, let us assume that neither (A) nor (B) apply to Δ. Then $[1,2,0] \in \Delta$ and $[2,0,0] \in \Delta$. Eiter and Gottlob [1] proved that if $P \in \mathcal{F}(\{[2,0,0],[1,3,0]\})$ and a is an atom then it is Π_2^P-hard to decide whether $P \models_s \neg a$. That result can be strengthened to the case when $P \in \mathcal{F}(\{[2,0,0],[1,2,0]\})$, as clauses of the arity $[1,3,0]$ can be simulated by clauses of arity $[1,2,0]$ (cf. the comments preceding Lemma 5). □

We note that credulous reasoning is simple (in P or in NP) for several classes of programs. In contrast, there are fewer classes of programs, for which skeptical reasoning

is simple (in P or coNP). The main reason behind this asymmetry is that in the cases (A2), (A3) and (B3) of Theorem 4 (positive programs) answer sets and minimal models coincide. Thus, in these cases, credulous reasoning asks for the existence of a *minimal* model that does not contain an atom a, which is equivalent to the existence of a model (not necessarily minimal) that does not contain a. In other words, the requirement of minimality becomes immaterial (one source of complexity disappears). This is not so with skeptical reasoning, where not having a in any minimal model is not the same as not having a in any model. A similar comparison of skeptical and credulous reasoning for positive programs was offered by Eiter and Gottlob for the coarser setting of classes of programs they considered [1].

6 Another Representation Schema

Finally, we consider briefly an alternative way, in which classes of programs could be described by means of arities of rules. When defining the class $\mathcal{F}(\Delta)$, we view each element $\alpha \in \Delta$ as a shorthand for the set of all arities β such that $\beta \preceq \alpha$. In other words, Δ is an *implicit* representation of the set of all allowed arities: not only those arities that are explicitly listed in Δ are legal but also those that are "dominated" by them.

There is another, more direct (more explicit), way to use arities to define classes of programs. Let $\Delta \subseteq \mathcal{T}$ be a set of *arities*, that is, we now do not allow superarities in Δ. We define $\mathcal{G}(\Delta)$ to consist of all finite programs P such that for every rule $r \in P$, $a(r) \in \Delta$. Thus, when defining the class $\mathcal{G}(\Delta)$, Δ serves as an *explicit* specification of the set of allowed arities.

One can show that the results of Section 3 can be adapted to the setting of classes of the form $\mathcal{G}(\Delta)$, where Δ is a *finite* set of arities. In particular, we have the following result.

Theorem 6. *Let $\Delta \subseteq \mathcal{T}$ be a finite set of arities. If there are no $k, m \geq 1$ such that $\{[k,0,0]\} \in \Delta$ or $\{[k,0,m]\} \in \Delta$, then the EAS problem for $\mathcal{G}(\Delta)$ is in* P. *Otherwise:*
(A) If

1. $\Delta \preceq \{[\infty,\infty,0],[0,0,0]\}$, *or*
2. $\Delta \preceq \{[1,\infty,0],[0,\infty,\infty]\}$, *or*
3. $\Delta \preceq \{[\infty,1,0],[0,1,0]\}$, *or*
4. $\Delta \preceq \{[i,j,0] \in \mathcal{T} \mid i+j \leq 2\}$,

then the EAS problem for $\mathcal{G}(\Delta)$ is in P.
(B) Otherwise, if

1. $\Delta \preceq \{[1,\infty,\infty],[0,\infty,\infty]\}$, *or*
2. $\Delta \preceq \{[\infty,1,\infty],[0,\infty,\infty]\}$, *or*
3. $\Delta \preceq \{[\infty,\infty,0],[0,\infty,0]\}$,

then the EAS problem for $\mathcal{G}(\Delta)$ is NP*-complete.*
(C) Otherwise, the EAS problem for $\mathcal{G}(\Delta)$ is Σ^2_P-complete.

Proof (sketch): Let us first assume that there are no $k, m \geq 1$ such that $\{[k, 0, 0]\} \in \Delta$ or $\{[k, 0, m]\} \in \Delta$, and let $P \in \mathcal{G}(\Delta)$. Then every rule in $\overline{P}$ has at least one positive atom in the body and so, $M = \emptyset$ is the unique answer set of $\overline{P}$. It can be verified in polynomial time whether $M = \emptyset$ is a model of $\overline{\overline{P}}$. Thus, the EAS problem for programs in $\mathcal{G}(\Delta)$ can be decided in polynomial time.

To prove the remaining part of the assertion, we note that the upper bound is implied directly by Theorem 2 (as $\mathcal{G}(\Delta) \subseteq \mathcal{F}(\Delta)$). To prove the lower bounds, we observe that if there are $k, m \geq 1$ such that $\{[k, 0, 0]\} \in \Delta$ or $\{[k, 0, m]\} \in \Delta$, then the EAS problem for $\mathcal{F}(\Delta)$ can be reduced to the EAS problem for $\mathcal{G}(\Delta)$. Indeed, let $P \in \mathcal{F}(\Delta)$ and let $r \in P$. Then there is $\alpha \in \Delta$ such that $a(r) \preceq \alpha$. Having $\{[k, 0, 0]\} \in \Delta$ or $\{[k, 0, m]\} \in \Delta$, where $k, m \geq 1$, allows us to "simulate" the effect of r with a rule r' of arity α obtained by repeating atoms in the head of r, and by inserting an atom a and a negated atom $not\ b$, where a and b are fresh, as many times as necessary in the body of r to "reach" the arity α. We also add the rule $a| \ldots |a \leftarrow$ or $a| \ldots |a \leftarrow not\ a', \ldots, not\ a'$, where a' is another fresh atom and a and $not\ a'$ are repeated k, or k and m times, respectively. □

Thus, as long as Δ is finite, our approach and results obtained earlier apply to the classes $\mathcal{G}(\Delta)$, as well. In particular, given a finite Δ, there is a polynomial-time (in the size of Δ) algorithm to decide which of the cases of Theorem 6 applies. In addition, there is a polynomial-time algorithm to decide whether $P \in \mathcal{G}(\Delta)$.

If Δ is infinite, the situation is different. For some infinite sets Δ there is no problem. For instance, when Δ is specified by means of a finite set Δ' of arities and superarities and consists of all arities α such that $\{\alpha\} \preceq \Delta$, then we have $\mathcal{G}(\Delta) = \mathcal{F}(\Delta')$. Thus, Theorem 2 can be used to determine the complexity of the EAS problem for programs from the class $\mathcal{G}(\Delta)$. However, in general, each finitary representation schema for the classes $\mathcal{G}(\Delta)$ would need to be studied separately and, possibly, it might not lend itself easily to reductions to Theorem 2.

7 Discussion

In the paper, we studied classes of programs defined in terms of "legal" arities of rules. Specifically, we focused on classes of programs of the form $\mathcal{F}(\Delta)$, where $\Delta \subseteq \mathcal{T}$. We proved that each such class has a finite representation and, for each finite set Δ, we determined the complexity of reasoning tasks for programs from $\mathcal{F}(\Delta)$. We also considered a related family of classes of programs, namely those of the form $\mathcal{G}(\Delta)$, where $\Delta \subseteq \mathcal{T}$ consists of arities only. For classes of programs of that form we established the complexity of the EAS problem, as well as that of the credulous and skeptical reasoning. In each case, the complexity is given by one of three complexity classes (P, NP-complete, and Σ_2^P-complete; or P, coNP-complete, and Π_2^P-complete, depending on the type of the reasoning task).

As we noted, our trichotomy results have some similarity to the dichotomy result by Schaefer, and its corollaries for other logic formalisms: the abductive reasoning [8], reasoning with minimal models [6] and, reasoning in default logic [7]. The classes of theories and formulas considered in those papers are defined in terms of boolean relations that are allowed in the language [3–5]. That definition schema satisfies the di-

chotomy property: for every class of formulas definable in that schema, the satisfiability problem is in P, or is NP-complete. The monotonicity of the propositional logic (the set of models of the conjunction of two formulas is the intersection of the sets of models of the conjuncts) is a fundamental property required by that result. Since logic programs with the answer-set semantics do not satisfy the monotonicity property, it is unclear how to extend that formalism the approach originated by Schaefer. Thus, we based our approach on a different definition schema developed specifically for programs, and related to the "complexity" of rules as measured by the numbers of atoms in the head, and positive and negative literals in the body.

It turns out though, that some classes of programs/theories appear prominently in both settings (for instance: Horn programs and Horn theories; positive programs with no more than two literals per rule and 2CNF theories). It is then an interesting problem whether a result based on the classification in terms of types of boolean relations can be obtained for disjunctive logic programs. One possibility might be to consider a more general setting of answer-set programs in the language of propositional logic under the semantics of equilibrium models [11].

Acknowledgments

This work was partially supported by the NSF grant IIS-0325063 and the KSEF grant KSEF-1036-RDE-008.

References

1. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: propositional case. Annals of Mathematics and Artificial Intelligence 15, 289–323 (1995)
2. Marek, W., Truszczyński, M.: Autoepistemic logic. Journal of the ACM 38, pp. 588-619 (1991)
3. Schaefer, T.: The complexity of satisfiability problems. In: Proceedings of the 10th Annual ACM Symposium on Theory of Computing, STOC 1978, pp. 216–226 (1978)
4. Bulatov, A.A., Jeavons, P., Krokhin, A.A.: Classifying the complexity of constraints using finite algebras. SIAM J. Comput. 34, 720–742 (2005)
5. Creignou, N., Khanna, S., Sudan, M.: Complexity Classifications of Boolean Constraint Satisfaction Problems. SIAM (2001)
6. Cadoli, M.: The complexity of model checking for circumscriptive formulae. Information Processing Letters 44, pp. 113–118 (1992)
7. Chapdelaine, P., Hermann, M., Schnoor, I.: Complexity of default logic on generalized conjunctive queries. In: Proceedings of LPNMR 2007. LNCS, vol. 4483, pp. 58-70. Springer (2007)
8. Nordh, G., Zanuttini, B.: What makes propositional abduction tractable. Artificial Intelligence 172, pp. 1245–1284 (2008)
9. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9** (1991) 365–385
10. Cadoli, M., Lenzerini, M.: The complexity of propositional closed world reasoning and circumscription. Journal of Computer and System Sciences 48, pp. 255–310 (1994)
11. Ferraris, P., Lifschitz, V.: Mathematical foundations of answer set programming. In: We Will Show Them! Essays in Honour of Dov Gabbay, College Publications, pp. 615–664 (2005)