Dual-normal logic programs – the forgotten class

JOHANNES K. FICHTE

TU Wien, Austria
University of Potsdam, Germany
(*e-mail:* `fichte@kr.tuwien.ac.at`)

MIROSŁAW TRUSZCZYŃSKI

University of Kentucky, Lexington, KY, USA
(*e-mail:* `mirek@cs.engr.uky.edu`)

STEFAN WOLTRAN

TU Wien, Austria
(*e-mail:* `woltran@dbai.tuwien.ac.at`)

Abstract

Disjunctive Answer Set Programming is a powerful declarative programming paradigm with complexity beyond NP. Identifying classes of programs for which the consistency problem is in NP is of interest from the theoretical standpoint and can potentially lead to improvements in the design of answer set programming solvers. One of such classes consists of *dual-normal programs*, where the number of positive body atoms in proper rules is at most one. Unlike other classes of programs, dual-normal programs have received little attention so far. In this paper we study this class. We relate dual-normal programs to propositional theories and to normal programs by presenting several inter-translations. With the translation from dual-normal to normal programs at hand, we introduce the novel class of *body-cycle* free programs, which are in many respects dual to head-cycle free programs. We establish the expressive power of dual-normal programs in terms of SE- and UE-models, and compare them to normal programs. We also discuss the complexity of deciding whether dual-normal programs are strongly and uniformly equivalent.

KEYWORDS: answer set programming, classes of logic programs, strong and uniform equivalence, propositional satisfiability

1 Introduction

Disjunctive Answer Set Programming (ASP) (Brewka *et al.* 2011) is a vibrant area of AI providing a declarative formalism for solving hard computational problems. Thanks to the power of modern ASP technology (Gebser *et al.* 2012), ASP was successfully used in many application areas, including product configuration (Soininen and Niemelä 1998), decision support for space shuttle flight controllers (Nogueira

et al. 2001; Balduccini *et al.* 2006), team scheduling (Ricca *et al.* 2012), and bio-informatics (Guziolowski *et al.* 2013).

With its main decision problems located at the second level of the polynomial hierarchy, full disjunctive ASP is necessarily computationally involved. But some fragments of ASP have lower complexity. Two prominent examples are the class of *normal* programs and the class of *head-cycle free* (HCF) programs (Ben-Eliyahu and Dechter 1994). In each case, the problem of the existence of an answer set is NP-complete. Identifying and understanding such fragments is of theoretical importance and can also help to make ASP solvers more efficient. A solver can detect whether a program is from an easier class (e.g., is normal or head-cycle free) and, if so, use a dedicated more lightweight machinery to process it.

HCF programs are defined by a *global* condition taking into account all rules in a program. On the other hand, interesting classes of programs can also be obtained by imposing conditions on *individual* rules. Examples include the classes of Horn, normal, negation-free, and purely negative programs. For instance, Horn programs consist of rules with at most one atom in the head and no negated atoms in the body, and purely negative programs consist of rules with no atoms in the positive body. A general schema to define classes of programs in terms of the numbers of atoms in the head and in the positive and negative bodies of their rules was proposed by Truszczyński (2011). In the resulting space of classes of programs, the complexity of the *consistency* problem (that is, the problem of the existence of an answer set) ranges from P to NP-complete to Σ_2^P-complete. The three main classes of programs in that space that fall into the NP-complete category are the classes of normal and negation-free programs (possibly with constraints), mentioned above, and the class of programs whose non-constraint rules have at most one positive atom in the body (Truszczyński 2011). While the former two classes have been thoroughly investigated, the third class has received little attention so far. In particular, the paper by Truszczyński (2011) only identified the class and established the complexity of the main reasoning tasks (deciding the consistency, and skeptical and credulous reasoning).

In this paper, we study this "forgotten" class in more detail. We call its programs *dual-normal*, since the reducts of their non-constraint part are *dual*-Horn. In fact, this is the reason why for dual-normal programs the consistency problem is in NP. Lower complexity is not the only reason why dual-normal programs are of interest. Let us consider a slight modification of the celebrated translation of a $(2, \exists)$-QBF $F = \exists X \forall Y D$ into a disjunctive program $P[F]$ devised by Eiter and Gottlob (1995). The translation assumes that D is a 3-DNF formula, say $D = \bigvee_{i=1}^{n}(l_{i,1} \wedge l_{i,2} \wedge l_{i,3})$, where $l_{i,j}$'s are literals over $X \cup Y$. To define $P[F]$ we introduce mutually distinct fresh atoms $w, \overline{x}$, for $x \in X$, $\overline{y}$, for $y \in Y$, and set

$$P[F] = \{x \vee \overline{x} \leftarrow \ | \ x \in X\} \cup \{y \vee \overline{y} \leftarrow; \ y \leftarrow w; \ \overline{y} \leftarrow w \ | \ y \in Y\} \cup$$
$$\{w \leftarrow l_{i,1}^*, l_{i,2}^*, l_{i,3}^* \ | \ 1 \leqslant i \leqslant n\} \cup \{\bot \leftarrow \neg \ w\}$$

where $l_{i,j}^* = \neg \overline{x}$ if $l_{i,j} = x$, $l_{i,j}^* = \neg x$ if $l_{i,j} = \neg x$, $l_{i,j}^* = y$ for $l_{i,j} = y$, $l_{i,j}^* = \overline{y}$ for $l_{i,j} = \neg y$. It can be shown that $P[F]$ has at least one answer set if and only if F is true. Let

us consider the subclass of $(2, \exists)$-QBFs where each term $l_{i,1} \wedge l_{i,2} \wedge l_{i,3}$ in F contains at most one universally quantified atom from Y. This restriction makes the Σ_2^P-complete problem of the validity of a $(2, \exists)$-QBF NP-complete, only. Moreover, it is easy to check that under that restriction, $P[F]$ is a dual-normal program. Since, the consistency problem for dual-normal programs is NP-complete (Truszczyński 2011) as well, dual-normal programs thus allow here for a straightforward *complexity-sensitive* reduction with respect to the subclass of the $(2, \exists)$-QBF problem mentioned above. Janhunen *et al.* (2006) proposed another translation of QBFs into programs that, with slight modifications, is similarly complexity-sensitive.

Main Contributions. Our first group of results concerns connections between dual-normal programs, propositional theories and normal programs. They are motivated by practical considerations of processing dual-normal programs. First, we give an efficient translation from dual-normal programs to SAT such that the models of the resulting formula encode the answer sets of the original program. While similar in spirit to translations to SAT developed for other classes of programs, our translation requires additional techniques to correctly deal with the dual nature of the programs under consideration. Second, in order to stay within the ASP framework we give a novel translation capable to express dual-normal programs as normal ones, *and* also vice versa, in each case producing polynomial-size encodings. In addition, this translation allows us to properly extend the class of dual-normal programs to the novel class of *body-cycle free* programs, a class for which the consistency problem is still located in NP.

In the next group of results, we investigate dual-normal programs from a different angle: their ability to express concepts modeled by classes of SE- and UE-models (Turner 2001; Eiter *et al.* 2013) and, in particular, to express programs under the notions of equivalence defined in terms of SE- and UE-models (Eiter *et al.* 2007). Among others, we show that the classes of normal and dual-normal programs are incomparable with respect to SE-models, and that dual-normal programs are strictly less expressive than normal ones with respect to UE-models. We also present results concerning the complexity of deciding strong and uniform equivalence between dual-normal programs.

2 Preliminaries

A *rule* r is an expression $H(r) \leftarrow B^+(r), \neg B^-(r)$, where $H(r) = \{a_1, \ldots, a_l\}$, $B^+(r) = \{a_{l+1}, \ldots, a_m\}$, $B^-(r) = \{a_{m+1}, \ldots, a_n\}$, l, m and n are non-negative integers, and a_i, $1 \leqslant i \leqslant n$, are propositional atoms. We omit the braces in $H(r)$, $B^+(r)$, and $B^-(r)$ if the set is a singleton. We occasionally write $\bot$ if $H(r) = \emptyset$. We also use the traditional representation of a rule as an expression $a_1 \vee \cdots \vee a_l \leftarrow a_{l+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n$. We call $H(r)$ the *head* of r and $B(r) = \{a_{l+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n\}$ the *body* of r. A rule r is *normal* if $|H(r)| \leqslant 1$, r is *Horn* if it is normal and $B^-(r) = \emptyset$, r is *dual-Horn* if $|B^+(r)| \leqslant 1$ and $B^-(r) = \emptyset$, r is an *(integrity) constraint* if $H(r) = \emptyset$, r is *positive* if $B^-(r) = \emptyset$, and r is *definite* if $|H(r)| = 1$. If $B^+(r) \cup B^-(r) = \emptyset$, we simply write $H(r)$ instead of $H(r) \leftarrow \emptyset, \emptyset$.

A *disjunctive logic program* (or simply a *program*) is a finite set of rules. We denote the set of atoms occurring in a program P by $\mathrm{at}(P)$. We often lift terminology from rules to programs. For instance, a program is *normal* if all its rules are normal. We also identify the parts of a program P consisting of proper rules as $P_r = \{r \in P \mid H(r) \neq \emptyset\}$ and constraints as $P_c = P \setminus P_r$. In this paper we are particularly interested in the following class.

Definition 1
A program P is called *dual-normal* if each rule r of P is either a constraint or $|B^+(r)| \leqslant 1$. Programs that are both normal and dual-normal are called *singular*.[1]

Note that dual-Horn programs may contain positive constraints with a single body atom but arbitrary constraints are forbidden in contrast to dual-normal programs.

Let P be a program and t a fresh atom. We define

$$P[t] \;=\; \{H(r) \leftarrow t, \neg B^-(r) \mid r \in P, B^+(r) = \emptyset\} \cup \{r \mid r \in P, B^+(r) \neq \emptyset\}$$

This transformation ensures non-empty positive bodies in rules and turns out to be useful in analyzing the semantics of dual-normal programs.

An *interpretation* is a set I of atoms. An interpretation I is a *model* of a program P, written $I \models P$, if I *satisfies* each rule $r \in P$, written $I \models r$, that is, if $(H(r) \cup B^-(r)) \cap I \neq \emptyset$ or $B^+(r) \setminus I \neq \emptyset$.

In the following when we say that a set M is maximal (minimal) we refer to inclusion-maximality (inclusion-minimality). A Horn program either has no models or has a unique least model. Dual-Horn programs have a dual property.

Proposition 1
Let P be dual-Horn. Then P has no models or has a unique maximal model.

We will now describe a construction that implies this result and is also of use in arguments later in the paper. Let us define $E_0 = \emptyset$ and, for $i \geqslant 1$,

$$E_i = \{b \mid H \leftarrow b \in P[t], \; H \subseteq E_{i-1}\}.$$

Intuitively, the sets E_i consist of atoms that *must not* be in any model of $P[t]$ (must be eliminated). The construction is dual to that for Horn programs. More precisely, the sets E_i can be alternatively defined as the results of recursively applying to $E_0 = \emptyset$ the one-step provability operator for the *definite Horn* program $P'[t] = \{b \leftarrow H \mid H \leftarrow b \in P[t]\}$. The following result summarizes properties of the program $P[t]$ and sets E_i.

Proposition 2
Let P be dual-Horn. Then, (1) $E_0 \subseteq E_1 \subseteq \ldots \subseteq \mathrm{at}(P) \cup \{t\}$; (2) $(\mathrm{at}(P) \cup \{t\}) \setminus \bigcup_{i=0}^{\infty} E_i$ is a maximal model (over $\mathrm{at}(P) \cup \{t\}$) of $P[t]$; (3) for every set M of atoms, M is a model of P if and only if $M \cup \{t\}$ is a model of $P[t]$; and (4) P has a model if and only if t belongs to the maximal model (over $\mathrm{at}(P) \cup \{t\}$) of $P[t]$ (or, equivalently, $t \notin \bigcup_{i=0}^{\infty} E_i$).

---

[1] Singular programs were also considered by Janhunen (2006), however under a different name.

Properties (3) and (4) imply Proposition 1. The construction can be implemented to run in linear time by means of the algorithm by Dowling and Gallier (1984) for computing the least model of a Horn program.

The *Gelfond-Lifschitz reduct* P^I of a program P *relative to* an interpretation I is defined as $P^I = \{H(r) \leftarrow B^+(r) \mid r \in P, I \cap B^-(r) = \emptyset\}$. Observe that for a dual-normal program P any reduct of P_r is dual-Horn. An interpretation I is an *answer set* of a program P if I is a minimal model of P^I (Gelfond and Lifschitz 1991; Przymusinski 1991). The set of all answer sets of a program P is denoted by $AS(P)$.

The following well-known characterization of answer sets is often invoked when considering the complexity of deciding the existence of answer sets.

Proposition 3
The following statements are equivalent for any program P and any set M of atoms: (1) $M \in AS(P)$, (2) M is a model of P and a minimal model of P_r^M, and (3) M is a model of P_c and $M \in AS(P_r)$.

This result identifies testing whether an interpretation M is a minimal model of P_r^M as the key task in deciding whether M is an answer set of P. For normal programs checking that M is a minimal model of P_r^M is easy. One just needs to compute the least model of the Horn program P_r^M and check whether it matches M. The general case requires more work. A possible approach is to reduce the task to that of deciding whether certain programs derived from P_r^M have models. Specifically, define for a program P and an atom $m \in \mathrm{at}(P)$

$$P|_m^M = P_r^M \cup \{\bot \leftarrow b \mid b \in \mathrm{at}(P) \setminus M\} \cup \{\bot \leftarrow m\}.$$

With this notation, we can restate Condition (2) in Proposition 3.

Proposition 4
An interpretation M is an answer set of a program P if and only if M is a model of P and for each $m \in M$, the program $P|_m^M$ has no models.

Clearly, if a program P is dual-normal, the programs $P|_m^M$ all are dual-Horn. Combining Propositions 2 and 4 yields the following corollary, as well as an efficient algorithm for checking whether M is an answer set of P.

Corollary 1
Let P be a dual-normal program. An interpretation M is an answer set of P if and only if M is a model of P and for every $m \in M$, $t_m \in \bigcup_{i=0}^{\infty} E_i$, where E_i are the sets computed based on $P|_m^M[t_m]$.

3 Translation into SAT

In this section, we encode dual-normal programs as propositional formulas so that the models of the resulting formulas encode the answer sets of the original programs. The main idea is to non-deterministically check for every interpretation whether it is an answer set of P. In other words, we encode into our formula a guess of an

interpretation and the efficient algorithm described above to check whether it has models (cf. Corollary 1). Note that the latter part is dual to the Horn encoding by Fichte and Szeider (2013).

Let P be a program and $p = |\text{at}(P)|$. The propositional variables in our encodings are given by all atoms $a \in \text{at}(P)$, a fresh variable t, and fresh variables a_m^i, for $a \in \text{at}(P) \cup \{t\}$, $m \in \text{at}(P)$, and $0 \leqslant i \leqslant p$. We use the variables a_m^i and t_m^i to simulate the computation of $\bigcup_{i=0}^{\infty} E_i$ based on the program $P|_m^M[t_m]$, when testing minimality of an interpretation M by trying to exclude m (cf. Corollary 1). The superscript i generates copies of atoms that represent the set E_i. Moreover, we write $P \sqcap B$ as a shorthand for $\{r \in P \mid B^+(r) = B\}$ and we write $E_i|_m^M$ to indicate that a set E_i is considered with respect to $P|_m^M[t_m]$ instead of $P[t_m]$.

The following auxiliary formulas simulate, according to Corollary 1, an inductive top-down computation of the maximal models of $P|_m^M[t_m]$, where M is an interpretation and $m \in M$. Since $P|_m^M[t_m]$ is dual-Horn the main part of our first auxiliary formulas is the encoding of the set $(\text{at}(P) \cup \{t_m\}) \setminus \bigcup_{i=0}^{\infty} E_i|_m^M$ where $m \in M$ and $0 \leqslant i \leqslant p$ (cf. Proposition 2 Properties (1) and (2)).

For the initial level 0, the following formula F_m^0 encodes $E_0|_m^M$. That is, it ensures that m does not belong to a model of F_m^0 and all other variables belong to a model of F_m^0 if and only if they do for the current interpretation over $\text{at}(P)$:

$$F_m^0 = \neg m_m^0 \wedge t_m^0 \wedge \bigwedge_{a \in \text{at}(P) \setminus \{m\}} (a_m^0 \leftrightarrow a).$$

The next formula encodes the set $(\text{at}(P) \cup \{t_m\}) \setminus E_i|_m^M$. In other words, we ensure that an atom a does not belong to the model if and only if there is a rule $r \in P|_m^M[t_m]$ where already all atoms in the head do not belong to the model (according to the previous step), and analogously for t_m^i:

$$F_m^i = \bigwedge_{a \in \text{at}(P) \setminus \{m\}} \left(a_m^i \leftrightarrow (a_m^{i-1} \wedge C_m^i(P_r \sqcap \{a\})) \right) \wedge \left(t_m^i \leftrightarrow (t_m^{i-1} \wedge C_m^i(P_r \sqcap \emptyset)) \right)$$

$$(\text{for } 1 \leqslant i \leqslant p) \quad \text{where } C_m^i(R) = \bigwedge_{r \in R} \left(\bigvee_{a \in H(r)} a_m^{i-1} \vee \bigvee_{a \in B^-(r)} a \right).$$

Note that in $C_m^i(R)$ the heads are evaluated with respect to the previous level while the negative bodies are evaluated with respect to the current model candidate, thus simulating the concept of reduct inherent in $P|_m^M[t_m]$.

Finally, the following auxiliary formula encodes the condition that an interpretation satisfies each rule $r \in P$:

$$F^{Mod} = \bigwedge_{r \in P} \left(\bigvee_{a \in H(r) \cup B^-(r)} a \vee \bigvee_{a \in B^+(r)} \neg a \right).$$

We now put these formulas together to obtain a formula $F(P)$ expressing that some interpretation $M \subseteq \text{at}(P)$ is a model of P and for every atom $a \in M$, atom t_a does not belong to the maximal model of $P|_a^M[t_a]$:

$$F(P) = F^{Mod} \wedge \bigwedge_{a \in \text{at}(P)} \left[a \rightarrow \left(\bigwedge_{i=0}^{p} F_a^i \wedge \neg t_a^p \right) \right].$$

It is easy to see that the formula $F(P)$ is of size $O(\|P\| \cdot |\text{at}(P)|^3)$, where $\|P\|$ stands for the size of P, and obviously we can construct it in polynomial time from P. The correctness of the translation is formally stated in the following result.

Theorem 1
Let P be a dual-normal program. Then, $AS(P) = \{M \cap \text{at}(P) \mid M \in Mod(F(P))\}$, where $Mod(F)$ denotes the set of all models of F.

Our encoding can be improved by means of an explicit encoding of the induction levels using counters (see e.g., (Janhunen 2006)). This allows to reduce the size of the encoding to $O(|\text{at}(P)| \cdot \|P\| \cdot \log |\text{at}(P)|)$.

4 Translation into normal programs

We now provide a polynomial-time translation from programs to programs that allows us to swap heads with positive bodies. It serves several purposes. (1) The translation delivers a normal program when the input program is dual-normal, and it delivers a dual-normal program when the input is normal. Given the complexity results by Truszczyński (2011), the existence of such translations is not surprising. However, the fact that there exists a *single* bidirectional translation, not tailored to any specific program class, is interesting. (2) When applied to head-cycle free programs (Ben-Eliyahu and Dechter 1994), the translation results in programs that we call *body-cycle* free. Body-cycle free programs are in many respects dual to head-cycle free ones.

To proceed, we need one more technical result which provides yet another characterization of answer sets of programs. It is closely related to the one given by Corollary 1 but more convenient to use when analyzing the translation we give below. Let P be a program and t a fresh atom. For every pair of atoms x, y, where $x \in \text{at}(P)$ and $y \in \text{at}(P) \cup \{t\}$ we introduce a fresh atom y_x, as an auxiliary atom representing a copy of y in P with respect to x; we clarify the role of these atoms below after the proof of Proposition 5.

Moreover, for every set $Y \subseteq \text{at}(P) \cup \{t\}$, let $Y_x = \{y_x \mid y \in Y\}$. With this notation in hand, we define

$$P_x = \{B_x^+ \leftarrow H_x, \neg B^- \mid H \leftarrow B^+, \neg B^- \in P_r[t]\},$$

and we write P_x^M for $(P^M)_x$ and P_r^M for $(P^M)_r = (P_r)^M$.

Proposition 5
Let P be a program. An interpretation $M \subseteq \text{at}(P)$ is an answer set of P if and only if M is a model of P, and for every $x \in M$, t_x belongs to every minimal model of $P_x^M \cup \{x_x\} \cup (\text{at}(P) \setminus M)_x$.

Proof
($\Leftarrow$) Since M is a model of P, M is a model of P^M. Thus, M is a model of P_r^M. By Proposition 3, it suffices to show that M is a minimal model of P_r^M.

Let us assume that for some $N \subset M$, $N \models P_r^M$. Let $x \in M \setminus N$. Finally, let us set $N' = \text{at}(P) \setminus N$. We will show that N'_x is a model of P_x^M. To this end, let us consider

a rule $U_x \leftarrow V_x$ in P_x^M such that $U_x \neq \{t_x\}$, and assume that $V_x \subseteq N_x'$. It follows that $V \subseteq N'$. Since the rule $V \leftarrow U$ belongs to P_r^M, $N \models P^M$, and $V \cap N = \emptyset$, we have $U \not\subseteq N$. Thus, $U \cap N' \neq \emptyset$ and so, $U_x \cap N_x' \neq \emptyset$. Hence, $N_x' \models U_x \leftarrow V_x$. Next, let us consider a rule $t_x \leftarrow V_x$ in P_x^M. Since $V \leftarrow$ is a rule in P_r^M and $N \models P_r^M$, we have $V \cap N \neq \emptyset$. Thus, $V \not\subseteq N'$ and so, $V_x \not\subseteq N_x'$. Consequently, $N_x' \models t_x \leftarrow V_x$.

Since $\{x\} \cup (\mathrm{at}(P) \setminus M) \subseteq N'$, it follows that $N_x' \models P_x^M \cup \{x_x\} \cup (\mathrm{at}(P) \setminus M)_x$. Since $t \notin N'$, $t_x \notin N_x'$. Thus, there is a minimal model of $P_x^M \cup \{x_x\} \cup (\mathrm{at}(P) \setminus M)_x$ that does not contain t_x, a contradiction (each minimal model of $P_x^M \cup \{x_x\} \cup (\mathrm{at}(P) \setminus M)_x$ contained in N_x' has this property).

($\Rightarrow$) Since $M \in AS(P)$, M is a model of P. Let us assume that for some $x \in M$ and for some minimal model N_x' of $P_x^M \cup \{x_x\} \cup (\mathrm{at}(P) \setminus M)_x$, $t_x \notin N_x'$. Let us define $N = \mathrm{at}(P) \setminus N_x'$. Since $\{x\} \cup (\mathrm{at}(P) \setminus M) \subseteq N'$, N is a subset of $M \setminus \{x\}$. Reasoning similarly as before, we can show that N is a model of P_r^M. This is a contradiction, as M is minimal model of P_r^M. Thus, the assertion follows by Proposition 3. □

By Proposition 5 checking whether M is an answer set of P requires to verify a certain condition for every $x \in M$. That condition could be formulated in terms of atoms in $\mathrm{at}(P) \cup \{t\}$ (by dropping the subscripts x in the atoms of the program P_x and in the condition). However, if a single normal program is to represent the condition for all $x \in M$ together, we have to combine the programs P_x. To avoid unwanted interactions, we first have to standardize the programs apart. This is the reason why we introduce atoms y_x and use them to define copies of P_x customized to individual x's.

Given a program P and the customized programs P_x, we now describe the promised translation. To this end, for every atom $x \in \mathrm{at}(P)$, we introduce a fresh atom $\bar{x}$. We set:

$$P_{\mathrm{xor}} = \{x \leftarrow \neg\bar{x}; \; \bar{x} \leftarrow \neg x \mid x \in \mathrm{at}(P)\}$$
$$P_{\mathrm{aux}} = \{x_x \leftarrow \neg\bar{x}; \; y_x \leftarrow \neg\bar{x}, \neg y \mid x, y \in \mathrm{at}(P)\}$$
$$P_{\mathrm{diag}} = P_{\mathrm{xor}} \cup P_{\mathrm{aux}} \cup \bigcup_{x \in \mathrm{at}(P)} P_x$$
$$P_{\mathrm{mod}} = \{\bot \leftarrow \neg H, B^+, \neg B^- \mid H \leftarrow B^+, \neg B^- \in P\}$$
$$P_{\mathrm{true}} = \{\bot \leftarrow x, \neg t_x \mid x \in \mathrm{at}(P)\}$$
$$P_{\mathrm{trans}} = P_{\mathrm{diag}} \cup P_{\mathrm{mod}} \cup P_{\mathrm{true}}$$

The following observations are immediate and central:

1. For a dual-normal program P, P_{trans} is normal.
2. For a normal program P, P_{trans} is dual-normal.

Hence, the following result not only establishes the connection between the answer sets of P and P_{trans} but also proves that the transformation encodes dual-normal as normal programs, as desired, and at the same time, encodes normal programs

as dual-normal ones. Moreover, the transformation can be implemented to run in polynomial time and so, produces polynomial-size programs.

Theorem 2
Let P be a program, $M \subseteq \text{at}(P)$, $P' = \bigcup_{x \in M}(P_x^M \cup \{x_x\} \cup (\text{at}(P) \setminus M)_x)$ and $M_P = M \cup \{\overline{x} \mid x \in \text{at}(P) \setminus M\}$. Then $M \in AS(P)$ if and only if for every minimal model N of P', $M_P \cup N \in AS(P_{\text{trans}})$. Moreover, every answer set of P_{trans} is of the form $M_P \cup N$ for $M \subseteq \text{at}(P)$ and a minimal model N of P'.

Proof
($\Rightarrow$) Let M be an answer set of P and let N be any minimal model of P'. Since M is a model of P by Proposition 3, $M_P \cup N$ satisfies all constraints in P_{mod}. Proposition 5 implies that for every $x \in M$, $t_x \in N$. Thus, $M_P \cup N$ also satisfies all constraints in P_{true}. To prove that $M_P \cup N \in AS(P_{\text{trans}})$ it remains to show that $M_P \cup N \in AS(P_{\text{diag}})$ (cf. Proposition 3). To this end, we observe that, for each $x \in \text{at}(P)$, $P_x^{M_P \cup N} = P_x^M$ and thus $P_{\text{diag}}^{M_P \cup N} = \bigcup_{x \in \text{at}(P)} P_x^M \cup M_P \cup \bigcup_{x \in M} (\{x_x\} \cup (\text{at}(P) \setminus M)_x)$. Since all rules in $\bigcup_{x \in \text{at}(P) \setminus M} P_x^M$ have a nonempty body that is disjoint with $M_P \cup N$, and since N is a model of $P' = \bigcup_{x \in M}(P_x^M \cup \{x_x\} \cup (\text{at}(P) \setminus M)_x)$, $M_P \cup N$ is a model of $P_{\text{diag}}^{M_P \cup N}$. Since N is a minimal model of P', $M_P \cup N$ is a minimal model of $P_{\text{diag}}^{M_P \cup N}$.

($\Leftarrow$) Let N be a minimal model of P' and $M_P \cup N$ an answer set of P_{trans}. Clearly, $M_P \cup N$ satisfies the constraints in P_{mod} and so, M is a model of P. Let $x \in M$. Since $M_P \cup N$ satisfies all constraints in P_{true}, $t_x \in M_P \cup N$. Thus, $t_x \in N$. By Proposition 5, M is an answer set of P.

To prove the second part of the assertion, let us consider an answer set A of P_{trans}. Let us define $M = A \cap \text{at}(P)$. Because of the rules in P_{xor}^x, $A = M_P \cup N$ for some set $N \subseteq \bigcup_{x \in \text{at}(P)}(\text{at}(P) \cup t)_x$. By Proposition 3, A is an answer set of P_{diag} that is, A is a minimal model of P_{diag}^A. As above, we have $P_{\text{diag}}^A = \left(\bigcup_{x \in \text{at}(P)} P_x\right)^M \cup M_P \cup \bigcup_{x \in M} (\{x_x\} \cup (\text{at}(P) \setminus M)_x)$ and conclude that N is a minimal model of P'. $\quad\square$

Our translation allows us to extend the class of dual-normal programs so that the problem to decide the existence of answer sets remains within the first level of the polynomial hierarchy. We recall that a program P is *head-cycle free* (HCF) (Ben-Eliyahu and Dechter 1994) if the positive dependency digraph of P has no directed cycle that contains two atoms belonging to the head of a rule in P. The *positive dependency digraph* of P has as vertices the atoms $\text{at}(P)$ and a directed edge (x, y) between any two atoms $x, y \in \text{at}(P)$ for which there is a rule $r \in P$ with $x \in H(r)$ and $y \in B^+(r)$. It is well known that it is NP-complete to decide whether a head-cycle free program has an answer set. The class of HCF programs arguably is the most natural class of programs that contains all normal programs and for which deciding the existence of answer sets is NP-complete.

We now define a program P to be *body-cycle free* (BCF) if the positive dependency graph of P, has no directed cycle that contains two atoms belonging to the *positive body* of a rule in P. In analogy to HCF programs, BCF programs trivially contain the class of dual-normal programs. Inspecting our translation, yields the following observations:

1. For a HCF program P, P_{trans} is BCF.
2. For a BCF program P, P_{trans} is HCF.

Since P_{trans} is efficiently obtained from P, the following result is a direct consequence of Theorem 2 and the fact that the consistency problem for HCF programs is NP-complete.

Theorem 3
The problem to decide whether a BCF program P has an answer set is NP-complete.

The translation P_{trans} preserves the cycle-freeness of the positive dependency graph (the positive dependency graph of P is cycle-free if and only if the positive dependency graph of P_{trans} is cycle-free). That is essential for our derivation of Theorem 3. However, in general, there is no one-to-one correspondence between answer sets of P and answer sets of P_{trans}. Thus, as a final result in this section, we provide a slight adaption of the translation P_{trans} in which the answer sets of programs P and P_{trans} are in a *one-to-one correspondence*. To this end define, $P^* = P_{\text{trans}} \cup \{y_x \leftarrow t_x \mid x, y \in \text{at}(P)\}$. Note that P^* still turns dual-normal programs to normal programs and vice versa, but we lose the property that cycle-freeness is preserved (the new rules may introduce additional cycles in the positive dependency graph). Thus, both Theorem 2 and Theorem 4 are of interest.

Theorem 4
Let P be a program, $M \subseteq \text{at}(P)$ and M_P as in Theorem 2. Then, $M \in AS(P)$ if and only if $M' = M_P \cup \bigcup_{x \in M}(\text{at}(P) \cup \{t\})_x \in AS(P^*)$. Moreover, every answer set of P^* is of the form M' for some $M \subseteq \text{at}(P)$.

5 Expressibility of dual-normal programs

SE-models, originating from the work by Turner (2001), and UE-models, proposed by Eiter and Fink (2003), characterize strong and uniform equivalence of programs, respectively. More recently, they turned out to be useful also for comparing program classes with respect to their expressivity (see e.g., work by Eiter *et al.* (2013)). In what follows, we first recall the main results from the literature, focusing on disjunctive and normal programs. Then, we complement these results by characterizations of collections of SE- and UE-models of dual-normal programs. Finally, we strengthen existing complexity results.

SE-models and UE-models. An *SE-interpretation* is a pair (X, Y) of sets of atoms such that $X \subseteq Y$. We denote by $\mathscr{S}_Z$ the class $\{(X, Y) \mid Y \subseteq Z\}$ of all SE-interpretations over Z. An SE-interpretation (X, Y) is an *SE-model* of a program P, written $(X, Y) \models_{SE} P$, if $Y \models P$ and $X \models P^Y$. SE-models of a program P contain, in particular, all information needed to identify the answer sets of P. Specifically, Y is an answer set of P if and only if $\langle Y, Y \rangle$ is an SE-model of P and for every $X \subset Y$, $\langle X, Y \rangle$ is not.

An SE-model (X, Y) of P is a *UE-model* of P if for every SE-model (X', Y) of P such that $X \subset X'$, $X' = Y$ holds. We write $SE(P)$ ($UE(P)$) for all SE-interpretations that are SE-models (UE-models) of a program P.

Programs P and Q are *equivalent*, denoted by $P \equiv Q$, if P and Q have the same answer sets. They are *strongly equivalent*, denoted by $P \equiv_s Q$, if for every program R,

$P \cup R \equiv Q \cup R$; and *uniformly equivalent*, denoted $P \equiv_u Q$, if for every set F of normal facts, $P \cup F \equiv Q \cup F$. The main results concerning these notions are (1) $P \equiv_s Q$ if and only if $SE(P) = SE(Q)$ (Lifschitz *et al.* 2001) and (2) $P \equiv_u Q$ if and only if $UE(P) = UE(Q)$ (Eiter and Fink 2003).

We now recall definitions of useful properties of sets of SE-interpretations (Eiter *et al.* 2013). A set $\mathscr{S}$ of SE-interpretations is *complete* if (1) $(X, Y) \in \mathscr{S}$ implies $(Y, Y) \in \mathscr{S}$; and (2) $(X, Y), (Z, Z) \in \mathscr{S}$ and $Y \subseteq Z$ imply $(X, Z) \in \mathscr{S}$. Next, $\mathscr{S}$ is *closed under here-intersection* if for all $(X, Y), (X', Y) \in \mathscr{S}$ we have $(X \cap X', Y) \in \mathscr{S}$. Finally, $\mathscr{S}$ is *UE-complete* if (1) $(X, Y) \in \mathscr{S}$ implies $(Y, Y) \in \mathscr{S}$; (2) $(X, Y), (Z, Z) \in \mathscr{S}$ and $Y \subset Z$ imply that there is Y' such that $Y \subseteq Y' \subset Z$ and $(Y', Z) \in \mathscr{S}$; and (3) $(X, Y), (X', Y) \in \mathscr{S}$ and $X \subset X'$ imply $X' = Y$.

The following results are due to Eiter *et al.* (2013). For each program P, $SE(P)$ is complete. Conversely, for every complete set $\mathscr{S} \subseteq \mathscr{S}_A$ there is a program P with $\mathrm{at}(P) \subseteq A$ and $SE(P) = \mathscr{S}$. For each normal program P, $SE(P)$ is complete and closed under here-intersection. Conversely, for every set $\mathscr{S}$ of SE-interpretations over A that is complete and closed under here-intersection there is a normal program P with $\mathrm{at}(P) \subseteq A$ and $SE(P) = \mathscr{S}$. Next, for every program P, $UE(P)$ is UE-complete. Conversely, for every UE-complete set $\mathscr{U} \subseteq \mathscr{S}_A$ of SE-interpretations over A there is a normal program P such that $\mathrm{at}(P) = A$ and $\mathscr{U} = UE(P)$. Hence, for every disjunctive program P there exists a normal program P' with $UE(P) = UE(P')$ (however, such P' can be exponentially larger than P (Eiter *et al.* 2004)). Finally, we make use of the following technical result.

Lemma 1

For every SE-interpretation (X, Y), $(X, Y) \models_{SE} A \leftarrow B, \neg C$ if and only if at least one of the following conditions holds: (1) $Y \cap C \neq \emptyset$; (2) $B \setminus Y \neq \emptyset$; (3) $X \cap A \neq \emptyset$; (4) $Y \cap A \neq \emptyset$ and $B \setminus X \neq \emptyset$.

Properties of Dual-Normal Programs. Our results rely on some new classes of sets of SE-interpretations. First, we introduce sets of SE-interpretations that are closed under here-union. This is the dual concept to sets closed under here-intersection. We will use it to characterize the SE-models of dual-normal programs. To characterize the UE models of dual-normal programs we need an additional, quite involved, concept of a splittable set.

Definition 2

A set $\mathscr{S}$ of SE-interpretations is called (1) *closed under here-union* if for any $(X, Y) \in \mathscr{S}$ and $(X', Y) \in \mathscr{S}$, also $(X \cup X', Y) \in \mathscr{S}$; (2) *splittable* if for every Z such that $(Z, Z) \in \mathscr{S}$ and every $(X_1, Y_1), \ldots, (X_k, Y_k) \in \mathscr{S}$ such that $Y_i \subseteq Z$ $(i = 1, \ldots, k)$, $(X_1 \cup \ldots \cup X_k, Z) \in \mathscr{S}$ or $(Z', Z) \in \mathscr{S}$ for some Z', such that $X_1 \cup \ldots \cup X_k \subseteq Z' \subset Z$.

Neither property implies the other in general. However, for UE-complete sets of SE-interpretations, splittability implies closure under here-union.

Proposition 6
If a UE-complete collection $\mathscr{S}$ of SE-interpretations is splittable, it is closed under here-union.

Proof
Let $(X_1, Z), (X_2, Z) \in \mathscr{S}$. By UE-completeness, $(Z, Z) \in \mathscr{S}$. Thus, if $X_1 \cup X_2 = Z$ then $(X_1 \cup X_2, Z) \in \mathscr{S}$. Otherwise, by splittability, $X_1 \cup X_2 \subseteq Z'$ for some Z' such that $Z' \subset Z$ and $(Z', Z) \in \mathscr{S}$. Since $X_1 \subseteq Z' \subset Z$ and $(X_1, Z), (Z', Z) \in \mathscr{S}$, $Z' = X_1$ (by Condition (3) of UE-completeness). Consequently, $X_1 \cup X_2 = X_1$ and so, $(X_1 \cup X_2, Z) \in \mathscr{S}$ in this case, too.   □

The converse does not hold, that is, for UE-complete sets, splittability is a strictly stronger concept than closure under here-union. As an example consider the set $\mathscr{S} = \{(b, b), (c, c), (ab, abcd), (cd, abcd), (abcd, abcd)\}$ that is UE-complete and closed under here-union. This set is *not* splittable. Indeed, $(abcd, abcd), (b, b), (c, c) \in \mathscr{S}$, yet there is no Z' such that $\{bc\} \subseteq Z' \subset \{abcd\}$ and $(Z', abcd) \in \mathscr{S}$.

As announced above, closure under here-union is an essential property of sets of SE-models of dual-normal programs.

Theorem 5
For every dual-normal program P, $SE(P)$ is complete and closed under here-union.

Proof
$SE(P)$ is complete for every program P. Let $(X, Y), (X', Y) \in SE(P)$. We need to show that for every rule $r = A \leftarrow B, \neg C$ in P, $(X \cup X', Y) \models_{SE} r$. To this end, let us assume that none of Conditions (1), (2), and (3) of Lemma 1 holds for $(X \cup X', Y)$ and r. Since $X \subseteq X \cup X'$ and $X' \subseteq X \cup X'$, none of Conditions (1), (2), and (3) holds for (X, Y) and r either. Since $(X, Y) \models_{SE} r$, Condition (4) must hold, that is, we have $Y \cap A \neq \emptyset$ and $B \setminus X \neq \emptyset$. The same argument applied to (X', Y) implies that also $B \setminus X' \neq \emptyset$. Since P is dual-normal, $B = \{b\}$ and $b \notin X \cup X'$. Thus, $B \setminus (X \cup X') \neq \emptyset$ and so, Condition (4) of Lemma 1 holds for $(X \cup X', Y)$ and r. Consequently, $(X \cup X', Y) \models_{SE} r$.   □

The conditions of Theorem 5 are not only necessary but also sufficient.

Theorem 6
For every set $\mathscr{S} \subseteq \mathscr{S}_A$ of SE-interpretations that is complete and closed under here-union, there exists a dual-normal program P with $\mathrm{at}(P) \subseteq A$ and $SE(P) = \mathscr{S}$.

Proof
We refer to the online appendix (Appendix A).   □

Thus the two theorems together provide a complete characterization of collections of SE-interpretations that can arise as collections of SE-models of dual-normal programs.

We now turn to the corresponding results for sets of UE-models of dual-normal programs. The key role here is played by the notion of splittability.

Theorem 7
For every dual-normal program P, $UE(P)$ is UE-complete and splittable.

Proof
The set $UE(P)$ is UE-complete for every program P. Thus, we only need to show splittability. Toward this end, let $(X_1, Y_1), \ldots, (X_k, Y_k), (Z, Z) \in UE(P)$, where $Y_i \subseteq Z$, for every $i = 1, \ldots, k$. Since, $(X_1, Y_1), \ldots, (X_k, Y_k), (Z, Z) \in SE(P)$, it follows that $(X_1, Z), \ldots, (X_k, Z) \in SE(P)$ (by the second condition of completeness). Since $SE(P)$ is closed under here-union, $(X_1 \cup \ldots \cup X_k, Z) \in SE(P)$. If $(X_1 \cup \ldots \cup X_k, Z) \in UE(P)$ we are done. Otherwise, $X_1 \cup \ldots \cup X_k \subset Z$ (since $(Z, Z) \in UE(P)$) and, by the definition of UE-models and finiteness of P, there is Z' such that $X_1 \cup \ldots \cup X_k \subset Z' \subset Z$ such that $(Z', Z) \in UE(P)$. □

As before, the conditions are also sufficient.

Theorem 8
For every set $\mathscr{U} \subseteq \mathscr{S}_A$ of SE-interpretations that is UE-complete and splittable, there is a dual-normal program P with $\mathrm{at}(P) \subseteq A$ such that $UE(P) = \mathscr{U}$.

Proof
We refer to the online appendix (Appendix A). □

We briefly discuss some implications of our results. Let $P = \{a \vee b; \perp \leftarrow \neg c; \ c \leftarrow a, b; \ a \leftarrow c; \ b \leftarrow c\}$. Then $SE(P) = \{(abc, abc), (a, abc), (b, abc)\}$ and it is neither closed under here-union nor under here-intersection. Thus, for P there are no strongly equivalent programs in the classes of normal and dual-normal programs. Moreover, $UE(P)$ is not closed under here-union and so, not splittable (Proposition 6). Therefore there is no dual-normal program P' such that $P \equiv_u P'$ (such a normal P' exists, however). Now let us consider the normal program $Q = P \setminus \{a \vee b\}$. We have $SE(Q) = SE(P) \cup \{(\emptyset, abc)\}$. Since $SE(Q)$ is not closed under here-union, there is no dual-normal program strongly equivalent to Q. Finally, consider the dual-normal program $R = P \setminus \{c \leftarrow a, b\}$. We have $SE(R) = SE(P) \cup \{(ab, abc)\}$. Since $SE(R)$ is not closed under here-intersection, there is no normal program strongly equivalent to R.

Complexity. We complement the following known results (Eiter *et al.* 2007): Checking strong equivalence between programs is coNP-complete; tractability is only known for the case when both programs are Horn. Checking uniform equivalence between programs is Π_2^P-complete. If one of the programs is normal, then the problem is coNP-complete.

Theorem 9
Checking strong equivalence between singular programs remains coNP-hard.

Proof
Take the standard reduction from UNSAT (as e.g. used by Pearce *et al.* (2009)) and let $F = \bigwedge_{i=1}^{n} (l_{i1} \vee l_{i2} \vee l_{i3})$. Define the singular program $P[F] = \{v \leftarrow \neg \bar{v}; \ \bar{v} \leftarrow \neg v; \ \leftarrow v, \neg v \mid v \in \mathrm{at}(F)\} \cup \{\leftarrow \neg l_{i1}^*, \neg l_{i2}^*, \neg l_{i3}^* \mid 1 \leqslant i \leqslant n\}$ where $l^* = l$ for positive literals and $l^* = \bar{v}$ for negative ones. One can show that F is a positive instance of UNSAT

if and only if $P[F] \equiv_s \{a \leftarrow; \leftarrow a\}$. Since the reduction works in polynomial time, coNP-hardness follows. ☐

Theorem 10

Checking uniform equivalence between dual-normal programs is coNP-complete. Hardness holds even in the case the programs are singular.

Proof

For membership, consider the following algorithm for the complementary problem. We guess (X, Y) and check whether $(X, Y) \in UE(P) \setminus UE(Q)$ or $(X, Y) \in UE(Q) \setminus UE(P)$. Checking whether $(X, Y) \in UE(P)$ can be done efficiently: First check $(Y, Y) \in UE(P)$ which reduces to classical model checking. If the test fails or $X = Y$ we are done. Otherwise, we compute for each $y \in Y \setminus X$ the maximal models of the dual-Horn theories $P^Y \cup X \cup \{\leftarrow z \mid z \in At \setminus Y\} \cup \{\leftarrow y\}$. This can be done in polynomial time, too. If all maximal models are equal to X, we return true; otherwise false. For hardness, one can employ the reduction used in the proof of Theorem 6.6 in (Eiter *et al.* 2007). ☐

6 Conclusions

We studied properties of dual-normal programs, the "forgotten" class of disjunctive programs, for which deciding the existence of answer sets remains NP-complete. We provided translations of dual-normal programs to propositional theories and to normal programs, and characterizations of sets of SE-interpretations that arise as sets of SE- and UE-models of dual-normal programs. We also established the coNP-completeness of deciding strong and uniform equivalence between dual-normal programs, showing hardness even under additional syntactic restrictions.

Our paper raises several interesting issues for future work. First, the BCF programs that we introduced as a generalization of dual-normal programs deserve further study because of their duality to HCF programs, and good computational properties (NP-completeness of deciding existence of answer sets). We believe that BCF programs provide a promising class to encode certain problems, since they also allow certain conjunctions in the positive body. Recall that the operation of *shifting* transforms HCF programs into normal ones while preserving the answer sets (Ben-Eliyahu and Dechter 1994). An analog of shifting for BCF programs would introduce negations in the heads of the rules. Thus, we plan to explore shifting within the broader setting of Lifschitz-Woo programs (Lifschitz and Woo 1992). On the other hand, singular programs, another class of programs we introduced, deserve attention due to their simplicity — they are both normal and dual-normal. As concerns dual-normal programs themselves, the key question is to establish whether more concise translations to SAT and normal programs are possible, as such translations may lead to effective ways of computing answer sets.

Acknowledgements

References

BALDUCCINI, M., GELFOND, M. AND NOGUEIRA, M. 2006. Answer set based design of knowledge systems. *Ann. Math. Artif. Intell. 47,* 1-2, 183–219.

BEN-ELIYAHU, R. AND DECHTER, R. 1994. Propositional semantics for disjunctive logic programs. *Ann. Math. Artif. Intell. 12,* 1-2, 53–87.

BREWKA, G., EITER, T. AND TRUSZCZYŃSKI, M. 2011. Answer set programming at a glance. *Communications of the ACM 54,* 12, 92–103.

DOWLING, W. F. AND GALLIER, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Logic Programming 1,* 3, 267–284.

EITER, T. AND FINK, M. 2003. Uniform equivalence of logic programs under the stable model semantics. In *Proceedings 19th International Conference on Logic Programming (ICLP 2003)*. LNCS, vol. 2916. Springer, 224–238.

EITER, T., FINK, M., PÜHRER, J., TOMPITS, H. AND WOLTRAN, S. 2013. Model-based recasting in answer-set programming. *J. Applied Non-Classical Logics 23,* 1-2, 75–104.

EITER, T., FINK, M., TOMPITS, H. AND WOLTRAN, S. 2004. On eliminating disjunctions in stable logic programming. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*. The AAAI Press, 447–458.

EITER, T., FINK, M. AND WOLTRAN, S. 2007. Semantical Characterizations and Complexity of Equivalences in Answer Set Programming. *ACM Trans. on Computational Logic 8,* 3.

EITER, T. AND GOTTLOB, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell. 15,* 3/4, 289–323.

FICHTE, J. K. AND SZEIDER, S. 2013. Backdoors to normality for disjunctive logic programs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI 2013)*. The AAAI Press, 320–327.

GEBSER, M., KAMINSKI, R., KAUFMANN, B. AND SCHAUB, T. 2012. *Answer Set Solving in Practice.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Comput. 9,* 3/4, 365–385.

GUZIOLOWSKI, C., VIDELA, S., EDUATI, F., THIELE, S., COKELAER, T., SIEGEL, A. AND SAEZ-RODRIGUEZ, J. 2013. Exhaustively characterizing feasible logic models of a signaling network using answer set programming. *Bioinformatics 29,* 18, 2320–2326. Erratum see Bioinformatics 30, 13, 1942.

JANHUNEN, T. 2006. Some (in)translatability results for normal logic programs and propositional theories. *J. Applied Non-Classical Logics 16,* 1-2, 35–86.

JANHUNEN, T., NIEMELÄ, I., SEIPEL, D., SIMONS, P. AND YOU, J.-H. 2006. Unfolding partiality and disjunctions in stable model semantics. *ACM Trans. Comput. Log. 7,* 1, 1–37.

LIFSCHITZ, V., PEARCE, D. AND VALVERDE, A. 2001. Strongly equivalent logic programs. *ACM Trans. on Computational Logic 2,* 4, 526–541.

LIFSCHITZ, V. AND WOO, T. Y. 1992. Answer sets in general nonmonotonic reasoning. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR 1992)*. Morgan Kaufmann, 603–614.

NOGUEIRA, M., BALDUCCINI, M., GELFOND, M., WATSON, R. AND BARRY, M. 2001. An A-Prolog decision support system for the Space Shuttle. In *Proceedings of the 3rd International*

Symposium on Practical Aspects of Declarative Language (PADL 2001). LNCS, vol. 1990. Springer, 169–183.

PEARCE, D., TOMPITS, H. AND WOLTRAN, S. 2009. Characterising equilibrium logic and nested logic programs: Reductions and complexity. *Theory Pract. Log. Program. 9,* 5, 565–616.

PRZYMUSINSKI, T. 1991. Stable semantics for disjunctive programs. *New Generation Comput. 9,* 401–424.

RICCA, F., GRASSO, G., ALVIANO, M., MANNA, M., LIO, V., IIRITANO, S. AND LEONE, N. 2012. Team-building with answer set programming in the Gioia-Tauro seaport. *Theory Pract. Log. Program. 12,* 361–381.

SOININEN, T. AND NIEMELÄ, I. 1998. Developing a declarative rule language for applications in product configuration. In *Proceedings of the 1st International Workshop on Practical Aspects of Declarative Languages (PADL 1999)*. LNCS, vol. 1551. Springer, 305–319.

TRUSZCZYŃSKI, M. 2011. Trichotomy and dichotomy results on the complexity of reasoning with disjunctive logic programs. *Theory Pract. Log. Program. 11,* 6, 881–904.

TURNER, H. 2001. Strong equivalence for logic programs and default theories (made easy). In *Proceedings of the 6th International Conference on Logic Programming and Nonmotonic Reasoning (LPNMR 2001)*, T. Eiter, W. Faber, and M. Truszczyński, Eds. LNCS, vol. 2173. Springer, Vienna, Austria, 81–92.