Relational Theories with Null Values and Non-Herbrand Stable Models

Vladimir Lifschitz, Karl Pichotta, and Fangkai Yang
Department of Computer Science
University of Texas at Austin
{vl,pichotta,fkyang}@cs.utexas.edu

Abstract

Generalized relational theories with null values in the sense of Reiter are first-order theories that provide a semantics for relational databases with incomplete information. In this paper we show that any such theory can be turned into an equivalent logic program, so that models of the theory can be generated using computational methods of answer set programming. As a step towards this goal, we develop a general method for calculating stable models under the domain closure assumption but without the unique name assumption.

Introduction

We re-examine here some of the problems discussed in two important papers on the semantics of null values that were published many years ago. The first of them is Ray Reiter's paper "Towards a logical reconstruction of relational database theory" (Reiter 1984). Generalized relational theories with null values in the sense of Reiter are first-order theories that provide a semantics for relational databases with incomplete information. The incompleteness can be of two kinds. One is represented by inclusive disjunction; for instance, the formula

$$SUPPLIES(Foo, p_1) \vee SUPPLIES(Foo, p_3) \qquad (1)$$

says: *Foo* supplies p_1 or p_3, maybe both. The other is represented by null values; by writing

$$SUPPLIER(\omega),\ SUPPLIES(\omega, p_3), \qquad (2)$$

where ω is a null value, we express that some supplier, which may or may not already be in the database, supplies p_3.

The second paper, by Bonnie Traylor and Michael Gelfond, is entitled "Representing null values in logic programming" (Traylor & Gelfond 1994). The authors define, among other things, the "logic programming counterpart" of a generalized relational theory with null values—a logic program whose meaning under the answer set semantics is similar to the meaning of the theory under the standard semantics of first-order logic.

We propose here an alternative approach to turning Reiter's theories into logic programs, which represents the meaning of the theory more closely than the one due to Traylor & Gelfond. We show also how these logic programs can be executed using computational methods of answer set programming (ASP) (Marek & Truszczyński 1999, Niemelä 1999, Lifschitz 2008)—for instance, by running the answer set solver CLINGO.[1]

The difference between null values and other object constants emphasized in Reiter's semantics is that null values are exempt from the unique name assumption: a null value may represent an object that has a name in the database, and two different null values may represent the same object. This fact leads us to the general problem of using answer set solvers for calculating the stable models that satisfy the domain closure assumption but may not satisfy the unique name assumption. Such models are allowed in some versions of the stable model semantics (Ferraris, Lee, & Lifschitz 2007, Ferraris, Lee, & Lifschitz 2011), just as they are allowed in the definition of circumscription (McCarthy 1980, McCarthy 1986). But existing answer set solvers do not deal with stable models of this kind directly. To take a simple example, the formula

$$P(a) \vee P(b) \qquad (3)$$

has minimal models of three kinds: in some of them, $P(a)$ is true, and $P(b)$ is false; in others, $P(a)$ is false, and $P(b)$ is true; finally, there are minimal models in which both $P(a)$ and $P(b)$ are true, along with the formula $a = b$. We will see how syntactic expressions describing these three possibilities can be generated by an answer set solver. Our method is applicable, in particular, to logic programs representing relational theories with null values.

The word "generalized" in Reiter's terminology indicates the possibility of including disjunctive information, and in the rest of the paper it will be omitted.

Relational Theories without Null Values

Review of Reiter's Semantics of Relational Theories

We begin with a signature that consists of finitely many object and predicate constants. A *positive ground clause* is a formula of the form $A_1 \vee \cdots \vee A_r$ ($r \geq 1$), where each A_i is a ground atomic formula whose predicate is distinct from the equality symbol. For instance, (1) is a positive ground clause. For any finite set Δ of positive ground clauses, the

[1] `http://potassco.sourceforge.net`

corresponding *relational theory* T_Δ is the set consisting of the following sentences:

- the *domain closure axiom DCA*:

$$\forall x \bigvee_a x = a$$

  where the disjunction extends over all object constants a;

- the *unique name axioms* $a \neq b$ for all pairs of distinct object constants a, b;

- the clauses Δ;

- for each predicate constant P, the *completion axiom*

$$\forall \mathbf{x} \left[P(\mathbf{x}) \rightarrow \bigvee_{\mathbf{a} \in W_P} \mathbf{x} = \mathbf{a} \right] \tag{4}$$

  where $\mathbf{x}$ is a tuple of distinct object variables, and W_P is the set of all tuples $\mathbf{a}$ of object constants such that $P(\mathbf{a})$ belongs to a clause from Δ.[2]

In view of the domain closure axiom *DCA* and the unique name axioms, any model of T_Δ is isomorphic to a Herbrand model.[3] Consequently, in the discussion of models of T_Δ we can restrict attention to Herbrand models.

Consider, for instance, Example 4.1 from (Reiter 1984). Its signature includes the object constants

$$p_1,\ p_2,\ p_3,\ \mathit{Acme},\ \mathit{Foo},$$

the unary predicate constants

$$\mathit{PART},\ \mathit{SUPPLIER},$$

and the binary predicate constants

$$\mathit{SUPPLIES},\ \mathit{SUBPART}.$$

The set Δ describes the following supplier and parts world:

PART	*SUPPLIER*	*SUPPLIES*	*SUBPART*
p_1	*Acme*	$\langle \mathit{Acme}\ \ p_1 \rangle$	$\langle p_1\ \ p_2 \rangle$
p_2	*Foo*	$\langle \mathit{Foo}\ \ p_2 \rangle$	
p_3			

In other words, it includes the corresponding atomic formulas:

$$\mathit{PART}(p_1),\ \mathit{PART}(p_2),\ \ldots,\ \mathit{SUBPART}(p_1, p_2). \tag{5}$$

In addition, Δ includes clause (1).

[2]The equality between two tuples of terms of the same length, such as $\mathbf{x} = \mathbf{a}$, stands for the conjunction of the equalities between the corresponding members of the tuples. We do not include equality axioms from (Reiter 1984) because we assume here the version of the semantics of first-order formulas that treats equality as identity (see, for instance, (Lifschitz, Morgenstern, & Plaisted 2008, Section 1.2.2)).

[3]Recall that in the absence of function constants of arity > 0 a *Herbrand interpretation* is an interpretation such that (i) its universe is the set of all object constants, and (ii) each object constant is interpreted as itself. A Herbrand interpretation can be identified with the set of all ground atomic formulas that are true in it and whose predicate is distinct from the equality symbol.

The completion axioms in this example are

$$\begin{array}{l}
\forall x(\mathit{PART}(x) \rightarrow x = p_1 \vee x = p_2 \vee x = p_3), \\
\forall x(\mathit{SUPPLIER}(x) \rightarrow x = \mathit{Acme} \vee x = \mathit{Foo}), \\
\forall xy(\mathit{SUPPLIES}(x, y) \rightarrow (x = \mathit{Acme} \wedge y = p_1) \\
\qquad\qquad\qquad\qquad \vee (x = \mathit{Foo} \wedge y = p_2) \\
\qquad\qquad\qquad\qquad \vee (x = \mathit{Foo} \wedge y = p_1) \\
\qquad\qquad\qquad\qquad \vee (x = \mathit{Foo} \wedge y = p_3)), \\
\forall xy(\mathit{SUBPART}(x, y) \rightarrow (x = p_1 \wedge y = p_2)).
\end{array}$$

Theory T_Δ has 3 Herbrand models:

$$\begin{array}{l}
I_1 = I \cup \{\mathit{SUPPLIES}(\mathit{Foo}, p_1)\}, \\
I_2 = I \cup \{\mathit{SUPPLIES}(\mathit{Foo}, p_3)\}, \\
I_3 = I \cup \{\mathit{SUPPLIES}(\mathit{Foo}, p_1), \mathit{SUPPLIES}(\mathit{Foo}, p_3)\},
\end{array}$$

where I is the set of atomic formulas (5).

Representing Relational Theories by Logic Programs

For any set Δ of positive ground clauses, by Π_Δ we denote the set of rules

$$1\{A_1, \ldots, A_r\} \tag{6}$$

for all clauses $A_1 \vee \cdots \vee A_r$ from Δ. Recall that this is an expression in the input language of CLINGO[4] that allows us to decide arbitrarily whether or not to include the atomic formulas $A_1, \ldots, A_r$ in the answer set as long as at least one of them is included.

The translation $1\{A\}$ of a unit clause A is strongly equivalent (Lifschitz, Pearce, & Valverde 2001, Lifschitz, Pearce, & Valverde 2007) to the fact A. Using this simplification we can say, for instance, that the logic program representing the example above consists of the facts (5) and the rule

$$1\{\mathit{SUPPLIES}(\mathit{Foo}, p_1), \mathit{SUPPLIES}(\mathit{Foo}, p_3)\}.$$

Furthermore, this program can be made more compact using the CLINGO conventions that allow us to use semicolons to merge a group of facts into one expression:

```
part(p1;p2;p3).
supplier(acme;foo).
supplies(acme,p1;;foo,p2).
subpart(p1,p2).
1{supplies(foo,p1),supplies(foo,p3)}.
```

Given this input, CLINGO returns 3 answer sets:

```
Answer: 1
part(p1) part(p2) part(p3) supplier(acme)
supplier(foo) supplies(foo,p2) supplies(acme,p1)
subpart(p1,p2) supplies(foo,p3)
Answer: 2
part(p1) part(p2) part(p3) supplier(acme)
supplier(foo)supplies(foo,p2) supplies(acme,p1)
subpart(p1,p2) supplies(foo,p1)
Answer: 3
part(p1) part(p2) part(p3) supplier(acme)
supplier(foo)supplies(foo,p2) supplies(acme,p1)
subpart(p1,p2) supplies(foo,p1) supplies(foo,p3)
```

[4]Such expressions, "cardinality constraints," appeared originally as part of the input language of the grounder LPARSE (`http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz`).

These answer sets are identical to the Herbrand models of the corresponding relational theory. This is an instance of a general theorem that expresses the soundness of our translation:

Theorem 1. *For any set Δ of positive ground clauses, a Herbrand interpretation I is a model of T_Δ iff I is an answer set of Π_Δ.*

Proofs of theorems, including a combined proof of Theorems 1 and 2, can be found at the end of the paper.

Null Values

Review of Reiter's Semantics of Null Values

We turn now to a more general framework. As before, the underlying signature is assumed to consist of finitely many object and predicate constants. We assume that the object constants are classified into two groups, *the database constants* and *the null values*. About a unique name axiom $a \neq b$ we say that it is *required* if both a and b are database constants, and that it is *optional* otherwise. As before, Δ stands for a finite set of positive ground clauses. Let Σ be a set of optional unique name axioms. The *relational theory with null values* $T_{\Delta,\Sigma}$ is the set of sentences obtained from T_Δ by removing all optional unique name axioms that do not belong to Σ. In other words, $T_{\Delta,\Sigma}$ consists of

- the domain closure axiom DCA,
- all required unique name axioms,
- the optional unique name axioms from Σ,
- the clauses Δ;
- the completion axioms (4).

Consider, for instance, the modification of our example in which

- the object constant ω is added to signature as the only null value,
- clause (1) is replaced in Δ with clauses (2), and
- $\Sigma = \{\omega \neq p_1, \omega \neq p_2, \omega \neq p_3\}$.

Thus ω is assumed to be a supplier that supplies part p_3; it may be identical to one of the suppliers *Acme*, *Foo* or may be different from both of them, and it is certainly different from p_1, p_2, p_3. The completion axioms in this example are

$$\begin{array}{l}\forall x(PART(x) \to x = p_1 \vee x = p_2 \vee x = p_3),\\ \forall x(SUPPLIER(x) \to x = Acme \vee x = Foo \vee x = \omega),\\ \forall xy(SUPPLIES(x,y) \to (x = Acme \wedge y = p_1)\\ \qquad\qquad\qquad\qquad \vee (x = Foo \wedge y = p_2)\\ \qquad\qquad\qquad\qquad \vee (x = \omega \wedge y = p_3)),\\ \forall xy(SUBPART(x,y) \to (x = p_1 \wedge y = p_2)).\end{array}$$

The set of unique name axioms of $T_{\Delta,\Sigma}$ includes neither $\omega \neq Acme$ nor $\omega \neq Foo$. Accordingly, this theory has models of different kinds: some of them satisfy $\omega = Acme$; some satisfy $\omega = Foo$; in some models, both equalities are false. We will later return to this example to give a complete description of its models.

Representing Theories with Null Values by Logic Programs

Since the axiom set $T_{\Delta,\Sigma}$ may not include some of the optional unique name axioms, it may have models that are not isomorphic to any Herbrand model. For this reason, the problem of relating $T_{\Delta,\Sigma}$ to logic programs becomes easier if we start with a semantics of logic programs that is not restricted to Herbrand models.

A version of the stable model semantics that covers non-Herbrand models is described in (Ferraris, Lee, & Lifschitz 2011, Section 2).[5] That paper deals with models of a first-order sentence and defines under what conditions such a model is considered stable relative to a subset $\mathbf{p}$ of the predicate constants of the underlying signature. The predicates from $\mathbf{p}$ are called "intensional." Unless stated otherwise, we will assume that $\mathbf{p}$ consists of all predicate constants of the underlying signature, so that every predicate constant (other than equality) is considered intensional. When this definition of a stable model is applied to a logic program, each rule of the program is viewed as shorthand for a first-order sentence, and the program is identified with the conjunction of these sentences. For instance, rule (6) can be viewed as shorthand for the formula

$$\bigwedge_{i=1}^{r}(A_i \vee \neg A_i) \wedge \neg \bigwedge_{i=1}^{r} \neg A_i.$$

(The first conjunctive term says, "choose the truth value of each A_i arbitrarily"; the second term prohibits making all atoms A_i false.)

The paper referenced above defines a syntactic transformation $\mathrm{SM}_{\mathbf{p}}$ that turns a first-order sentence F into a conjunction

$$F \wedge \cdots$$

where the dots stand for a second-order sentence (the "stability condition"). The stable models of F are defined as arbitrary models (in the sense of second-order logic) of $\mathrm{SM}_{\mathbf{p}}[F]$.

From this perspective, Theorem 1 asserts that a Herbrand interpretation is a model of T_Δ iff it is a model of $\mathrm{SM}_{\mathbf{p}}[\Pi_\Delta]$, where $\mathbf{p}$ is the set of all predicate constants of the underlying signature.

By $\Pi_{\Delta,\Sigma}$ we denote the conjunction of Π_Δ with DCA and with all unique name axioms from $T_{\Delta,\Sigma}$ (that is to say, with all unique name axioms except for the optional axioms that do not belong to Σ). The following theorem expresses the soundness of this translation:

Theorem 2. *For any set Δ of positive ground clauses and any set Σ of optional unique name axioms, $T_{\Delta,\Sigma}$ is equivalent to $\mathrm{SM}_{\mathbf{p}}[\Pi_{\Delta,\Sigma}]$, where $\mathbf{p}$ is the set of all predicate constants.*

In other words, an interpretation I is a model of $T_{\Delta,\Sigma}$ iff I is a stable model of $\Pi_{\Delta,\Sigma}$.

One useful property of the operator $\mathrm{SM}_{\mathbf{p}}$ is that

$$\mathrm{SM}_{\mathbf{p}}[F \wedge G] \text{ is equivalent to } \mathrm{SM}_{\mathbf{p}}[F] \wedge G$$

[5] Other possible approaches to the semantics of logic programs that are not limited to Herbrand models use program completion (Clark 1978) without Clark's equality axioms and the logic of nonmonotone inductive definitions (Denecker & Ternovska 2008).

whenever G does not contain intensional predicates (that is, predicate constants from $\mathbf{p}$).[6] For instance, let $\Pi^-_{\Delta,\Sigma}$ be the conjunction of Π_Δ with the unique name axioms from $T_{\Delta,\Sigma}$; then $\Pi_{\Delta,\Sigma}$ is $\Pi^-_{\Delta,\Sigma} \wedge DCA$. Since DCA does not contain intensional predicates (recall that all atomic parts of DCA are equalities), $\mathrm{SM}_{\mathbf{p}}[\Pi_{\Delta,\Sigma}]$ is equivalent to $\mathrm{SM}_{\mathbf{p}}[\Pi^-_{\Delta,\Sigma}] \wedge DCA$. The assertion of Theorem 2 can be reformulated as follows: an interpretation I is a model of $T_{\Delta,\Sigma}$ iff I is a stable model of $\Pi^-_{\Delta,\Sigma}$ that satisfies DCA.

As we have seen, the translation Π_Δ makes it possible to generate models of T_Δ using an answer set solver. Unfortunately, the translation $\Pi_{\Delta,\Sigma}$ does not do the same for relational theories with null values. In the presence of null values we are interested in non-Herbrand models (for instance, in the models of the theory from the example above that satisfy $\omega = Acme$), but answer set solvers are designed to generate Herbrand stable models only. There is also a more basic question: a Herbrand interpretation can be viewed as a set of ground atomic formulas, but how will we describe non-Herbrand models by syntactic expressions? These questions are addressed in the next section.

Stable Models without the Uniqueness of Names

Diagrams

Consider a signature σ consisting of finitely many object and predicate constants. By HB_σ we denote the Herbrand base of σ, that is, the set of its ground atomic formulas whose predicate is distinct from the equality symbol. By EHB_σ ("extended" Herbrand base) we denote the set of all ground atomic formulas, including equalities between object constants. For any interpretation I of σ satisfying the domain closure axiom (*DCA-interpretation*, for short), by $D(I)$ we will denote the set of the formulas from EHB_σ that are true in I. This set will be called the *diagram* of I.[7]

If a subset X of EHB_σ is the diagram of a DCA-interpretation then it is clear that

- the set of equalities in X is closed under reflexivity (it includes $a = a$ for every object constant a), symmetry (includes $b = a$ whenever it includes $a = b$), and transitivity (includes $a = c$ whenever it includes $a = b$ and $b = c$), and
- X is closed under substitution: it includes $P(b_1, \dots, b_n)$ whenever it includes $P(a_1, \dots, a_n), a_1 = b_1, \dots, a_n = b_n$.

The converse holds also:

Theorem 3. *If a subset X of EHB_σ is closed under substitution, and the set of equalities in X is closed under reflexivity, symmetry, and transitivity, then there exists a DCA-interpretation I such that $D(I) = X$. Furthermore, this interpretation is unique up to isomorphism.*

Since relational theories with null values include the domain closure assumption, Theorem 3 shows that their models can be completely characterized by diagrams. In the example above, the theory has 3 non-isomorphic models J_1, J_2, J_3. The diagram of J_1 consists of the formulas (5), (2), and $a = a$ for all object constants a. The diagrams of the other two are given by the formulas

$$J_2 = J_1 \cup \{SUPPLIES(Acme, p_3), SUPPLIES(\omega, p_1), \omega = Acme, Acme = \omega\},$$
$$J_3 = J_1 \cup \{SUPPLIES(Foo, p_3), SUPPLIES(\omega, p_2), \omega = Foo, Foo = \omega\}.$$

Calculating General Stable Models

The problem that we are interested in can be now stated as follows: Given a first-order sentence F, we would like to construct a first-order sentence F' such that the diagrams of all DCA-interpretations satisfying $\mathrm{SM}_{\mathbf{p}}[F]$ can be easily extracted from the Herbrand interpretations satisfying $\mathrm{SM}_{\mathbf{p}}[F']$. We say "can be easily extracted from" rather than "are identical to" because diagrams include equalities between object constants, and Herbrand models do not; occurrences of equality in F will have to be replaced in F' by another symbol. Our goal, in other words, is to define F' in such a way that diagrams of the stable DCA-models of F will be nearly identical to Herbrand stable models of F'.

The examples of F that we are specifically interested in are the formulas $\Pi^-_{\Delta,\Sigma}$, because stable DCA-models of that sentence are identical to models of $T_{\Delta,\Sigma}$. As a simpler example, consider formula (3). It has 3 minimal DCA-models, with the diagrams

$$\begin{aligned} &K_1 = \{P(a), a = a, b = b\}, \\ &K_2 = \{P(b), a = a, b = b\}, \\ &K_3 = \{P(a), P(b), a = a, b = b, a = b, b = a\}. \end{aligned} \tag{7}$$

Our translation $F \mapsto F'$ will allow us to construct these diagrams using ASP.

The solution described below uses the binary predicate constant Eq, which is assumed not to belong to σ. For any first-order formula F of the signature σ, $F^=_{Eq}$ stands for the formula of the signature $\sigma \cup \{Eq\}$ obtained from F by replacing each subformula of the form $t_1 = t_2$ with $Eq(t_1, t_2)$. (Here t_1, t_2 are terms, that is, object constants or object variables.) The notation $X^=_{Eq}$, where X is a set of formulas of the signature σ, is understood in a similar way. By E_σ we denote the conjunction of the logically valid sentences

$$\forall x(x = x).$$
$$\forall xy(x = y \to y = x),$$
$$\forall xyz(x = y \wedge y = z \to x = z),$$

and

$$\forall \mathbf{xy}(P(\mathbf{x}) \wedge \mathbf{x} = \mathbf{y} \to P(\mathbf{y}))$$

for all predicate constants P from σ, where $\mathbf{x}$, $\mathbf{y}$ are disjoint tuples of distinct variables.

In the statement of the theorem below, F is an arbitrary sentence of the signature σ, and $\mathbf{p}$ stands for the set of all predicate constants of σ.

Theorem 4. *For any DCA-interpretation I of the signature σ that satisfies $\mathrm{SM}_{\mathbf{p}}[F]$, the Herbrand interpretation $D(I)^=_{Eq}$ of the signature $\sigma \cup \{Eq\}$ satisfies*

$$\mathrm{SM}_{\mathbf{p}}[(F \wedge E_\sigma)^=_{Eq}]. \tag{8}$$

Conversely, any Herbrand model of this formula is $D(I)^=_{Eq}$ for some DCA-interpretation I of σ satisfying $\mathrm{SM}_{\mathbf{p}}[F]$.

[6] See (Ferraris, Lee, & Lifschitz 2011, Section 5.1).

[7] This is essentially the "positive diagram" of I, as this term is used in model theory (Robinson 1963, Section 2.1), for the special case of DCA-interpretations.

In other words, the transformation $I \mapsto D(I)^{=}_{Eq}$ maps the class of stable *DCA*-models of F onto the set of Herbrand stable models of $(F \wedge E_\sigma)^{=}_{Eq}$. The second part of Theorem 3 shows that this transformation is one-to-one up to isomorphism.

By Theorem 2 from (Ferraris, Lee, & Lifschitz 2011), formula (8) is equivalent to

$$\mathrm{SM}_{\mathbf{p},Eq}[(F \wedge E_\sigma)^{=}_{Eq} \wedge \forall xy(Eq(x,y) \vee \neg Eq(x,y))]. \quad (9)$$

The advantage of this reformulation is that it treats all predicate constants of the signature $\sigma \cup \{Eq\}$ as intensional. This is useful, because existing answer set solvers calculate Herbrand stable models under the assumption that all predicate constants occurring in the program (except for "predefined predicates") are intensional.

For example, the diagrams (7) of the minimal *DCA*-models of (3) are identical, modulo replacing $=$ with Eq, to the Herbrand stable models of the conjunction of the formulas (3),

$$\begin{array}{c} \forall x Eq(x,x), \\ \forall xy(Eq(x,y) \to Eq(y,x)), \\ \forall xyz(Eq(x,y) \wedge Eq(y,z) \to Eq(x,z)), \end{array} \quad (10)$$

$$\forall xy(P(x) \wedge Eq(x,y) \to P(y)),$$

and

$$\forall xy(Eq(x,y) \vee \neg Eq(x,y)). \quad (11)$$

In logic programming syntax, this conjunction can be written as

```
p(a)|p(b).
eq(X,X).
eq(X,Y) :- eq(Y,X).
eq(X,Z) :- eq(X,Y), eq(Y,Z).
p(Y) :- p(X), eq(X,Y).
{eq(X,Y)}.
```

To make this program safe[8] we need to specify that the only possible values of the variables `X` and `Y` are `a` and `b`. This can be accomplished by including the lines

```
u(a;b).
#domain u(X). #domain u(Y).
#hide u/1.
```

(The auxiliary predicate symbol `u` describes the "universe" of the program.) Now the program can be grounded by GRINGO, and its Herbrand stable models can be generated by CLASPD.[9] The output

```
Answer: 1
eq(b,b) eq(a,a) p(b)
Answer: 2
eq(b,b) eq(a,a) p(a)
Answer: 3
eq(b,b) eq(a,a) eq(b,a) eq(a,b) p(a) p(b)
```

is essentially identical to the list (7) of minimal models, as could be expected on the basis of Theorem 4.

The Python script NONH.PY (for "non-Herbrand") is a preprocessor that turns a program F of a signature σ without function symbols of arity > 0, written in the input language of GRINGO, into the program

$$(F \wedge E_\sigma)^{=}_{Eq} \wedge \forall xy(Eq(x,y) \vee \neg Eq(x,y)),$$

written in the language of GRINGO also. Thus the Herbrand stable models of the output of NONH.PY are the diagrams of the stable *DCA*-models of the input (with equality replaced by Eq). As in the example above, a "universe" predicate is used to ensure that whenever the input of NONH.PY is safe, the output is safe also. The diagrams of the minimal *DCA*-models of formula (3) can be generated by saving that formula, in the form

```
p(a)|p(b).
```

in a file, say `disjunction.lp`, and then executing the command

```
% nonH.py disjunction.lp | gringo | claspD 0
```

(the CLASPD option 0 instructs it to generate all answer sets, not one). The script can be downloaded from `http://www.cs.utexas.edu/users/fkyang/nonH/`.

Calculating Models of a Relational Theory with Null Values

The method applied above to the disjunction $P(a) \vee P(b)$ can be applied also to the formula $\Pi^{-}_{\Delta,\Sigma}$. Stable *DCA*-models of this formula can be generated using CLINGO with the preprocessor NONH.PY. The preprocessor has two options that can be useful here. The command line

```
% nonH.py <filename> -una <list of constants>
```

instructs the preprocessor to conjoin its output with the unique name axioms $a \neq b$ for all pairs a, b of distinct object constants from the given list. The command line

```
% nonH.py <filename> -no-una <list of constants>
```

adds the unique name axioms $a \neq b$ for all pairs a, b of distinct object constants such that at least one of them does not occur in the given list. The diagrams of models of our relational theory with null values can be generated by saving the rules

```
part(p1;p2;p3).
supplier(acme;foo;omega).
supplies(acme,p1;;foo,p2;;omega,p3).
subpart(p1,p2).
:- omega==p1.
:- omega==p2.
:- omega==p3.
```

in a file, say `db.lp`, and then executing the command

```
% nonH.py db.lp -no-una omega | clingo 0
```

[8] Safety is a syntactic condition required for "intelligent instantiation"—part of the process of generating answer sets. In the program above, the rules `eq(X,X)` and `{eq(X,Y)}` are unsafe.

[9] GRINGO and CLASPD are "relatives" of CLINGO; see Footnote ([1]) for a reference. CLINGO itself cannot be used in this case because the program is disjunctive. CMODELS (`http://www.cs.utexas.edu/users/tag/cmodels.html`) would do as well. Using the solver DLV (`http://www.dlvsystem.com`) will become an option too after eliminating choice rules in favor of disjunctive rules with auxiliary predicates. We are grateful to Yuliya Lierler for helping us identify the software capable of executing this program.

The following output will be produced:

```
Answer: 1
part(p1) part(p3) part(p2) supplier(acme)
supplier(omega) supplier(foo) supplies(omega,p3)
supplies(foo,p2) supplies(acme,p1) subpart(p1,p2)
eq(omega,omega) eq(foo,foo) eq(acme,acme)
eq(p3,p3) eq(p2,p2) eq(p1,p1) eq(omega,foo)
eq(foo,omega) supplies(omega,p2) supplies(foo,p3)
Answer: 2
part(p1) part(p3) part(p2) supplier(acme)
supplier(omega) supplier(foo) supplies(omega,p3)
supplies(foo,p2) supplies(acme,p1) subpart(p1,p2)
eq(omega,omega) eq(foo,foo) eq(acme,acme)
eq(p3,p3) eq(p2,p2) eq(p1,p1)
Answer: 3
part(p1) part(p3) part(p2) supplier(acme)
supplier(omega) supplier(foo) supplies(omega,p3)
supplies(foo,p2) supplies(acme,p1) subpart(p1,p2)
eq(omega,omega) eq(foo,foo) eq(acme,acme)
eq(p3,p3) eq(p2,p2) eq(p1,p1) eq(omega,acme)
eq(acme,omega) supplies(acme,p3)
supplies(omega,p1)
```

It is essentially identical to the set of diagrams J_1, J_2, J_3.

Comparison with the Traylor—Gelfond Translation

The approach to encoding relational theories with null values by logic programs proposed in (Traylor & Gelfond 1994) does not have the property established for $\Pi_{\Delta,\Sigma}$ in Theorem 2: generally, there is no 1–1 correspondence between the models of $T_{\Delta,\Sigma}$ and the answer sets of the Traylor—Gelfond translation. For instance, the logic programming counterpart of our main example in the sense of (Traylor & Gelfond 1994) has 2 answer sets, not 3. It uses strong (classical) negation (Gelfond & Lifschitz 1991), and its answer sets are incomplete sets of literals. One of them, for instance, includes $SUPPLIES(Foo, p_1)$ but does not include either of the two complementary literals $SUPPLIES(Foo, p_3)$, $\neg SUPPLIES(Foo, p_3)$. This is how the program expresses the possibility of p_3 being supplied by *Foo*, along with p_1. The result of (Traylor & Gelfond 1994) describes the relation of $T_{\Delta,\Sigma}$ to the intersection of the answer sets of its logic programming counterpart, not to the individual answer sets.

Logic programming counterparts in the sense of (Traylor & Gelfond 1994), like our programs $\Pi_{\Delta,\Sigma}$, can be turned into executable ASP code. The reason why that was not done in that paper is simply that the paper was written too early—the first answer set solver appeared on the scene two years after its publication (Niemelä & Simons 1996).

Proofs of Theorems

Proofs of Theorems 1 and 2

Lemma 1. *For any finite set Δ of positive ground clauses, formula $\mathrm{SM}_{\mathbf{p}}[\Pi_\Delta]$ is equivalent to the conjunction of the clauses Δ and the completion axioms (4).*

Proof. Let C be the conjunction of the formulas

$$P(\mathbf{a}) \vee \neg P(\mathbf{a}) \tag{12}$$

for all atomic formulas $P(\mathbf{a})$ occurring in Δ. It is clear that Π_Δ is strongly equivalent[10] to the conjunction of C with the formulas

$$\neg \bigwedge_{i=1}^{r} \neg A_i \tag{13}$$

for all clauses $A_1 \vee \cdots \vee A_r$ from Δ. According to Theorem 3 from (Ferraris, Lee, & Lifschitz 2011), it follows that $\mathrm{SM}_{\mathbf{p}}[\Pi_\Delta]$ is equivalent to the conjunction of $\mathrm{SM}_{\mathbf{p}}[C]$ with formulas (13). Furthermore, (12) is strongly equivalent to $\neg\neg P(\mathbf{a}) \to P(\mathbf{a})$. Consequently C is strongly equivalent to the conjunction of the formulas

$$\forall \mathbf{x} \left[\bigvee_{\mathbf{a} \in W_P} (\neg\neg P(\mathbf{x}) \wedge \mathbf{x} = \mathbf{a}) \ \to \ P(\mathbf{x}) \right]$$

for all predicate constants P. By Theorem 11 from (Ferraris, Lee, & Lifschitz 2011), it follows that $\mathrm{SM}_{\mathbf{p}}[C]$ is equivalent to

$$\forall \mathbf{x} \left[P(\mathbf{x}) \ \leftrightarrow \ \bigvee_{\mathbf{a} \in W_P} (\neg\neg P(\mathbf{x}) \wedge \mathbf{x} = \mathbf{a}) \right]. \tag{14}$$

It remains to observe that (13) is equivalent to $A_1 \vee \cdots \vee A_r$, and that (14) is equivalent to (4).

Theorem 1. *For any set Δ of positive ground clauses, a Herbrand interpretation I is a model of T_Δ iff I is an answer set of Π_Δ.*

Proof. A Herbrand interpretation is a model of T_Δ iff it satisfies the clauses Δ and the completion axioms (4). On the other hand, a Herbrand interpretation is an answer set of Π_Δ iff it satisfies $\mathrm{SM}_{\mathbf{p}}[\Pi_\Delta]$. Consequently the assertion of the theorem follows from Lemma 1.

Theorem 2. *For any set Δ of positive ground clauses and any set Σ of optional unique name axioms, $T_{\Delta,\Sigma}$ is equivalent to $\mathrm{SM}_{\mathbf{p}}[\Pi_{\Delta,\Sigma}]$, where $\mathbf{p}$ is the set of all predicate constants.*

Proof. Recall that $\Pi_{\Delta,\Sigma}$ is $\Pi_\Delta \wedge DCA \wedge U$, where U is the conjunction of all unique name axioms from $T_{\Delta,\Sigma}$. Since neither DCA nor U contains intensional predicates, $\mathrm{SM}_{\mathbf{p}}[\Pi_{\Delta,\Sigma}]$ is equivalent to $\mathrm{SM}_{\mathbf{p}}[\Pi_\Delta] \wedge DCA \wedge U$. By Lemma 1, it follows that $\mathrm{SM}_{\mathbf{p}}[\Pi_{\Delta,\Sigma}]$ is equivalent to the conjunction of the clauses Δ, the completion axioms (4), and the formulas DCA and U; that is to say, it is equivalent to $T_{\Delta,\Sigma}$.

Proof of Theorem 3

Theorem 3. *If a subset X of EHB_σ is closed under substitution, and the set of equalities in X is closed under reflexivity, symmetry, and transitivity, then there exists a DCA-interpretation I such that $D(I) = X$. Furthermore, this interpretation is unique up to isomorphism.*

Proof. The binary relation

$$a = b \ \text{ is in } \ X \tag{15}$$

[10] See (Ferraris, Lee, & Lifschitz 2011, Section 5).

between object constants a, b is an equivalence relation on the set of object constants. For any predicate constant P, the n-ary relation

$$P(a_1, \dots, a_n) \text{ is in } X \tag{16}$$

between object constants $a_1, \dots, a_n$ can be extended to equivalence classes of (15). Consider the interpretation I such that

- the universe of I is the set of equivalence classes of relation (15),
- I interprets each object constant a as the equivalence class that contains a,
- I interprets each predicate constant P as the extension of the corresponding relation (16) to equivalence classes.

Interpretation I satisfies DCA, and $D(I) = X$.

To prove the second claim, consider any DCA-interpretation J such that $D(J) = X$. For any object constant a, let $f(a)$ be the element of the universe of J that represents a. Function f can be extended to equivalence classes of relation (15), and this extension is an isomorphism between I and J.

Proof of Theorem 4

The proof of Theorem 4 is based on the fact that a DCA-interpretation I satisfies a first-order sentence F of the signature σ iff the Herbrand interpretation $D(I)^{=}_{Eq}$ satisfies $F^{=}_{Eq}$. This is easy to verify by induction on the size of F. What we need actually is a similar proposition for second-order sentences, because the formulas obtained by applying the operator $\mathrm{SM}_{\mathbf{p}}$ contain predicate variables. The straightforward generalization to second-order sentences is invalid, however. For instance, let F be the formula

$$\exists v(v(a) \wedge \neg v(b)) \tag{17}$$

(v is a unary predicate variable). This formula is equivalent to $a \neq b$. If the universe of an interpretation I is a singleton then I does not satisfy F. On the other hand, the result of replacing $=$ with Eq in F is F itself, because this formula does not contain equality. It is satisfied by every Herbrand interpretation, including $D(I)^{=}_{Eq}$.

To overcome this difficulty, we will define the transformation $F \mapsto F^{=}_{Eq}$ for second-order sentences in such a way that it will involve, in addition to replacing $=$ with Eq, restricting the second-order quantifiers in F.

In this section, a *second-order formula* is a formula that may involve predicate variables, either free or existentially quantified, but not function variables. (An extension to universally quantified predicate variables is straightforward, but it is not needed for our purposes.) For any predicate variable v, $Sub(v)$ stands for the formula

$$\forall x_1 \cdots x_n y_1 \cdots y_n (v(x_1, \dots, x_n) \wedge Eq(x_1, y_1) \wedge \cdots \wedge Eq(x_n, y_n) \rightarrow v(y_1, \dots, y_n)),$$

where n is the arity of v. For any second-order formula F of the signature σ, $F^{=}_{Eq}$ stands for the second-order formula of the signature $\sigma \cup \{Eq\}$ obtained from F by

- replacing each subformula of the form $t_1 = t_2$ with $Eq(t_1, t_2)$, and
- restricting each second-order quantifier $\exists v$ to $Sub(v)$.

For instance, is F is (17) then $F^{=}_{Eq}$ is

$$\exists v(Sub(v) \wedge v(a) \wedge \neg v(b)).$$

In application to first-order formulas, the notation $F^{=}_{Eq}$ has the same meaning as before.

Lemma 2. *A DCA-interpretation I satisfies a second-order sentence F of the signature σ iff the Herbrand interpretation $D(I)^{=}_{Eq}$ satisfies $F^{=}_{Eq}$.*

Proof. The proof is by induction on the size of F; size is understood as follows. About second-order sentences F and G we say that F is *smaller than* G if

- F has fewer second-order quantifiers than G, or
- F has the same number of second-order quantifiers as G, and the total number of first-order quantifiers and propositional connectives in F is less than in G.

The induction hypothesis is that the assertion of the lemma holds for all sentences that are smaller than F. If F is atomic then

$$\begin{array}{lll} I \models F & \text{iff} & F \in D(I) \\ & \text{iff} & F^{=}_{Eq} \in D(I)^{=}_{Eq} \\ & \text{iff} & D(I)^{=}_{Eq} \models F^{=}_{Eq}. \end{array}$$

If F is $G \wedge H$ then $F^{=}_{Eq}$ is $G^{=}_{Eq} \wedge H^{=}_{Eq}$. Using the induction hypothesis, we calculate:

$$\begin{array}{lll} I \models F & \text{iff} & I \models G \text{ and } I \models H \\ & \text{iff} & D(I)^{=}_{Eq} \models G^{=}_{Eq} \text{ and } D(I)^{=}_{Eq} \models H^{=}_{Eq} \\ & \text{iff} & D(I)^{=}_{Eq} \models F^{=}_{Eq}. \end{array}$$

For other propositional connectives the reasoning is similar. If F is $\forall x G(x)$ then $F^{=}_{Eq}$ is $\forall x \left(G(x)^{=}_{Eq}\right)$. Using the induction hypothesis and the fact that I satisfies DCA, we calculate:

$$\begin{array}{lll} I \models F & \text{iff} & \text{for all object constants } a,\ I \models G(a) \\ & \text{iff} & \text{for all object constants } a,\ D(I)^{=}_{Eq} \models G(a)^{=}_{Eq} \\ & \text{iff} & D(I)^{=}_{Eq} \models F^{=}_{Eq}. \end{array}$$

For the first-order existential quantifier the reasoning is similar.

It remains to consider the case when F is $\exists v G(v)$, where v is a predicate variable. To simplify notation, we will assume that the arity of v is 1. For any set V of object constants, by exp_V we denote the lambda-expression[11] $\lambda x \bigvee_{a \in V}(x = a)$. Since I is a DCA-interpretation, $I \models F$ iff

$$\text{for some } V, \quad I \models G(exp_V).$$

By the induction hypothesis, this is equivalent to the condition

$$\text{for some } V, \quad D(I)^{=}_{Eq} \models H((exp_V)^{=}_{Eq}), \tag{18}$$

where $H(v)$ stands for $G(v)^{=}_{Eq}$. On the other hand, $F^{=}_{Eq}$ is $\exists v(Sub(v) \wedge H(v))$. The Herbrand interpretation $D(I)^{=}_{Eq}$ satisfies this formula iff

$$\begin{array}{ll} \text{for some } V, & D(I)^{=}_{Eq} \models Sub(exp_V) \\ & \text{and } D(I)^{=}_{Eq} \models H(exp_V). \end{array} \tag{19}$$

[11] On the use of lambda-expressions in logical formulas, see (Lifschitz 1994, Section 3.1).

We need to show that (19) is equivalent to (18).

Consider first the part

$$D(I)^{=}_{Eq} \models Sub(exp_V) \tag{20}$$

of condition (19), that is,

$$D(I)^{=}_{Eq} \models \forall xy(exp_V(x) \wedge Eq(x,y) \rightarrow exp_V(y)).$$

It is equivalent to

$$D(I)^{=}_{Eq} \models \forall y(\exists x(exp_V(x) \wedge Eq(x,y)) \rightarrow exp_V(y)).$$

Interpretation $D(I)^{=}_{Eq}$ satisfies the inverse of this implication, because it satisfies $\forall x Eq(x,x)$. Consequently condition (20) can be equivalently rewritten as

$$D(I)^{=}_{Eq} \models \forall y(\exists x(exp_V(x) \wedge Eq(x,y)) \leftrightarrow exp_V(y)).$$

The left-hand side of this equivalence can be rewritten as $\bigvee_{a\in V} Eq(a,y)$. It follows that condition (20) is equivalent to

$$D(I)^{=}_{Eq} \models \forall y \left(\bigvee_{a\in V} Eq(a,y) \leftrightarrow exp_V(y)\right).$$

Furthermore, $Eq(a,y)$ can be replaced here by $Eq(y,a)$, because $D(I)^{=}_{Eq}$ satisfies $\forall xy(Eq(x,y) \leftrightarrow Eq(y,x))$. Hence (20) is equivalent to

$$D(I)^{=}_{Eq} \models (exp_V)^{=}_{Eq} = exp_V.$$

It follows that (19) is equivalent to the condition

$$\begin{array}{ll} \text{for some } V, & D(I)^{=}_{Eq} \models (exp_V)^{=}_{Eq} = exp_V \\ & \text{and } D(I)^{=}_{Eq} \models H((exp_V)^{=}_{Eq}). \end{array} \tag{21}$$

It is clear that (21) implies (18).

It remains to check that (18) implies (21). Assume that

$$D(I)^{=}_{Eq} \models H((exp_V)^{=}_{Eq}), \tag{22}$$

and let V' be the set of object constants a such that, for some $b \in V$, $I \models a = b$. We will show that V' can be taken as V in (21). The argument uses two properties of the set V' that are immediate from its definition:

(a) $V \subseteq V'$;

(b) if $I \models a = b$ and $a \in V'$ then $b \in V'$.

Consider the first half of (21) with V' as V:

$$D(I)^{=}_{Eq} \models (exp_{V'})^{=}_{Eq} = exp_{V'}.$$

This condition can be restated as follows: for every object constant a,

$$D(I)^{=}_{Eq} \models \bigvee_{b\in V'} Eq(a,b) \quad \text{iff} \quad D(I)^{=}_{Eq} \models \bigvee_{b\in V'}(a=b),$$

or, equivalently,

$$I \models \bigvee_{b\in V'}(a=b) \quad \text{iff} \quad a \in V'.$$

The implication left-to-right follows from property (b) of V'; the implication right-to-left is obvious (take b to be a).

Consider now the second half of (21) with V' as V:

$$D(I)^{=}_{Eq} \models H((exp_{V'})^{=}_{Eq}).$$

To derive it from (22), we only need to check that

$$D(I)^{=}_{Eq} \models (exp_{V'})^{=}_{Eq} = (exp_V)^{=}_{Eq}.$$

This claim is equivalent to

$$I \models exp_{V'} = exp_V \tag{23}$$

and can be restated as follows: for every object constant a,

$$I \models \bigvee_{b\in V'}(a=b) \quad \text{iff} \quad I \models \bigvee_{b\in V}(a=b).$$

The implication left-to-right is immediate from the definition of V'; the implication righ-to-left is immediate from property (a).

In the following lemma, as in the statement of Theorem 4, F is an arbitrary sentence of the signature σ, and $\mathbf{p}$ stands for the set of all predicate constants of σ.

Lemma 3. *For any DCA-interpretation I of the signature σ,*

$$I \models \mathrm{SM}_{\mathbf{p}}[F] \quad \textit{iff} \quad D(I)^{=}_{Eq} \models \mathrm{SM}_{\mathbf{p}}[(F \wedge E_\sigma)^{=}_{Eq}].$$

Proof. Recall that $\mathrm{SM}_{\mathbf{p}}[F]$ is defined as

$$F \wedge \neg\exists \mathbf{v}((\mathbf{v} < \mathbf{p}) \wedge F^*(\mathbf{v}))$$

(Ferraris, Lee, & Lifschitz 2011, Section 2.3), so that $\mathrm{SM}_{\mathbf{p}}[(F \wedge E_\sigma)^{=}_{Eq}]$ is

$$F^{=}_{Eq} \wedge (E_\sigma)^{=}_{Eq} \wedge \neg\exists \mathbf{v}((\mathbf{v} < \mathbf{p}) \wedge F^*(\mathbf{v})^{=}_{Eq} \wedge E^*_\sigma(\mathbf{v})^{=}_{Eq}). \tag{24}$$

From the definitions of E_σ and of the transformation $F \mapsto F^*(\mathbf{v})$ (Ferraris, Lee, & Lifschitz 2011, Section 2.3) we see that $E^*_\sigma(\mathbf{v})$ is the conjunction of E_σ and the formulas

$$\forall \mathbf{xy}(v(\mathbf{x}) \wedge \mathbf{x} = \mathbf{y} \rightarrow v(\mathbf{y}))$$

for all members v of tuple $\mathbf{v}$. Consequently $E^*_\sigma(\mathbf{v})^{=}_{Eq}$ is the conjunction of $(E_\sigma)^{=}_{Eq}$ and the formulas $Sub(v)$ for all members v of tuple $\mathbf{v}$. It follows that (24) can be written as

$$\begin{array}{l} F^{=}_{Eq} \wedge (E_\sigma)^{=}_{Eq} \\ \quad \wedge \neg\exists \mathbf{v}\left((\mathbf{v} < \mathbf{p}) \wedge F^*(\mathbf{v})^{=}_{Eq} \wedge (E_\sigma)^{=}_{Eq} \wedge \bigwedge_v Sub(v)\right). \end{array}$$

This formula is equivalent to

$$F^{=}_{Eq} \wedge \neg\exists \mathbf{v}\left(\bigwedge_v Sub(v) \wedge ((\mathbf{v} < \mathbf{p}) \wedge F^*(\mathbf{v}))^{=}_{Eq}\right) \wedge (E_\sigma)^{=}_{Eq},$$

which can be written as

$$\mathrm{SM}_{\mathbf{p}}[F]^{=}_{Eq} \wedge (E_\sigma)^{=}_{Eq}.$$

The interpretation $D(I)^{=}_{Eq}$ satisfies the second conjunctive term. By Lemma 2, $D(I)^{=}_{Eq}$ satisfies the first conjunctive term iff I satisfies $\mathrm{SM}_{\mathbf{p}}[F]$.

Theorem 4. *For any DCA-interpretation I of the signature σ that satisfies $\mathrm{SM}_{\mathbf{p}}[F]$, the Herbrand interpretation $D(I)^{=}_{Eq}$ of the signature $\sigma \cup \{Eq\}$ satisfies*

$$\mathrm{SM}_{\mathbf{p}}[(F \wedge E_\sigma)^{=}_{Eq}].$$

Conversely, any Herbrand model of this formula is $D(I)^{=}_{Eq}$ for some DCA-interpretation I of σ satisfying $\mathrm{SM}_{\mathbf{p}}[F]$.

Proof. The first assertion is identical to the only-if part of Lemma 3. To prove the second assertion, consider a Herbrand model J of $\mathrm{SM}_{\mathbf{p}}[(F \wedge E_\sigma)^{=}_{Eq}]$. Since this formula entails $(E_\sigma)^{=}_{Eq}$, J is a model of $(E_\sigma)^{=}_{Eq}$ as well. It follows that the subset X of EDB_σ such that $X^{=}_{Eq} = J$ is closed under substitution, and the set of equalities in X is closed under reflexivity, symmetry, and transitivity. By Theorem 3, there exists a *DCA*-interpretation I such that $D(I) = X$, so that $D(I)^{=}_{Eq} = J$. By the if part of Lemma 3, I satisfies $\mathrm{SM}_{\mathbf{p}}[F]$.

Conclusion

This paper contributes to the direction of research on the semantics of null values started in (Reiter 1984) and (Traylor & Gelfond 1994). It demonstrates a close relationship between Reiter's semantics of disjunctive databases and cardinality constraints in answer set programming. It shows also how answer set solvers can be used for computing models of relational theories with null values.

On the other hand, this paper improves our understanding of the role of non-Herbrand stable models. Are they merely a mathematical curiosity, or can they have serious applications to knowledge representation? We have provided arguments in favor of the usefulness of this generalization of the stable model semantics by showing, first, how non-Herbrand stable models can serve for representing null values, and second, how they can be generated using existing software systems.

The generalization of the stable model semantics proposed in (Ferraris, Lee, & Lifschitz 2011) extends the original definition of a stable model in two ways: syntactically (it is applicable to arbitrary first-order formulas) and semantically (a stable model can be non-Herbrand). The preprocessor F2LP (Lee & Palla 2009) allows us to use existing answer set solvers for generating stable models of some syntactically complex formulas. On the other hand, the preprocessor NONH.PY, described in this paper, allows us to use answer set solvers for generating some non-Herbrand stable models—those that satisfy the domain closure assumption but not the unique name assumption. The two programs can be used together. For instance, the stable *DCA*-models of the formula

$$(P(a) \wedge P(b)) \vee (P(c) \wedge P(d))$$

(there are 23 of them) can be generated by running F2LP on the file

```
(p(a) & p(b)) | (p(c) & p(d)).
```

and then running consecutively NONH.PY, GRINGO, and CLASPD.

Acknowledgements

Many thanks to Marc Denecker, Michael Gelfond, and Yuliya Lierler for useful discussions related to the topic of this paper, and to the anonymous referees for valuable advice.

References

Clark, K. 1978. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases*. New York: Plenum Press. 293–322.

Denecker, M., and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Transactions on Computational Logic* 9(2).

Ferraris, P.; Lee, J.; and Lifschitz, V. 2007. A new perspective on stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 372–379.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175:236–263.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Lee, J., and Palla, R. 2009. System F2LP — computing answer sets of first-order formulas. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 515–521.

Lifschitz, V.; Morgenstern, L.; and Plaisted, D. 2008. Knowledge representation and classical logic. In van Harmelen, F.; Lifschitz, V.; and Porter, B., eds., *Handbook of Knowledge Representation*. Elsevier. 3–88.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2:526–541.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2007. A characterization of strong equivalence for logic programs with variables. In *Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 188–200.

Lifschitz, V. 1994. Circumscription. In Gabbay, D.; Hogger, C.; and Robinson, J., eds., *Handbook of Logic in AI and Logic Programming*, volume 3. Oxford University Press. 298–352.

Lifschitz, V. 2008. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1594–1597. MIT Press.

Marek, V., and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*. Springer Verlag. 375–398.

McCarthy, J. 1980. Circumscription—a form of nonmonotonic reasoning. *Artificial Intelligence* 13:27–39,171–172.

McCarthy, J. 1986. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence* 26(3):89–116.

Niemelä, I., and Simons, P. 1996. Efficient implementation of the well-founded and stable model semantics. In *Proceedings Joint Int'l Conf. and Symp. on Logic Programming*, 289–303.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25:241–273.

Reiter, R. 1984. Towards a logical reconstruction of relational database theory. In Brodie, M.; Mylopoulos, J.; and Schmidt, J., eds., *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*. Springer. 191–233.

Robinson, A. 1963. *Introduction to model theory and to the metamathematics of algebra*. North-Holland.

Traylor, B., and Gelfond, M. 1994. Representing null values in logic programming. In Nerode, A., and Matiyasevich, Y., eds., *Logical Foundations of Computer Science, Third International Symposium, LFCS'94, St. Petersburg, Russia, July 11-14, 1994, Proceedings*, volume 813 of *Lecture Notes in Computer Science*, 341–352.