Disjunctive Logic Programs with Existential Quantification in Rule Heads

Jia-Huai You[1], Heng Zhang[2], Yan Zhang[2]
[1]*University of Alberta, Edmonton T6G 2E8, Canada*
[2]*University of Western Sydney, Penrith, NSW 2751, Australia*

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

Abstract

We consider disjunctive logic programs without function symbols but with existential quantification in rule heads, under the semantics of general stable models. There are at least two interesting prospects in these programs. The first is that a program can be made more succinct by using existential variables, and the second is on the potential in representing defeasible ontological knowledge by these logic programs. This paper studies some of the properties of these programs. First, we show a simple yet intuitive definition of stable models for these programs that does not resort to second-order logic. Second, the stable models of these programs can be characterized by an extension of progression for disjunctive programs, which provides a native characterization of justification for stable models. We then study the decidability issue. While the stable model existence problem for safe disjunctive programs is decidable, with existential quantification allowed in rule heads the problem becomes undecidable. We identify an interesting decidable fragment by exploring a new notion of stratification over existential quantification.

KEYWORDS: Disjunctive Logic Programs, General Stable Models, Existential Quantification

1 Introduction

Answer set programming (ASP) has been generalized from normal logic programming (Gelfond and Lifschitz 1988) to arbitrary first-order sentences (Ferraris et al. 2011). Some classes of programs in this context have been studied and properties revealed (e.g., (Lee and Meng 2011; Lee and Palla 2012; Lee et al. 2008; Cabalar et al. 2009; Bartholomew and Lee 2010)). In this paper, we consider (first-order) disjunctive logic programs without function symbols but with existential quantification (EQ) in rule heads, which we call *E-disjunctive programs* (or just *E-programs*).

E-disjunctive programs are interesting in at least two aspects. First, they provide a flexible knowledge representation language, which may enable succinct encoding of knowledge. As an illustration, consider the k-colorability problem: given a graph and k colors, determine whether each vertex can be assigned a color in such a way that any two vertices that are connected by an arc must not have the same color. The problem can be encoded by the following program:

$$\begin{array}{l} \exists Y set(X,Y) \leftarrow vertex(X) \\ noColoring \leftarrow edge(X,Y), set(X,C), set(Y,C) \\ noColoring \leftarrow set(X,Y), \mathsf{not}\ color(Y) \end{array}$$

where free variables are universally quantified before the existential ones, e.g., the first rule sets each vertex X to some Y. It is not difficult to see that given a graph and k colors, the program has a stable model containing no $noColoring$ iff the graph is k-colorable (cf. (Eiter et al. 1997)). This encoding may be compared with a typical normal logic program encoding. While the first rule above realizes "a vertex is colored with exactly one color" by a free ride of minimization, in (Niemelä 1999) the same is encoded using two normal rules along with a new predicate symbol.

Another interest in E-disjunctive programs is their potential in representing and reasoning with defeasible ontological knowledge. For example, consider an example modified from (Bonatti et al. 2011): "Staff members are either users or non-users; users who are not known to present a security threat are given access with some access level."

$$\begin{array}{l}
user(X) \vee nonuser(X) \leftarrow staff(X) \\
\exists Y accessLevel(X,Y) \leftarrow user(X), \text{not } secThreat(X) \\
secThreat(X) \leftarrow blackListed(X) \\
\leftarrow accessLevel(X,Y), \text{not } secLevel(Y)
\end{array}$$

The constraint at the end forces variable Y to be bound to one of provided security levels by facts on the predicate $secLevel/1$ (e.g., admin, privileged, standard, basic, etc). Given appropriate facts, one can query, e.g., whether there is a black listed user with some access rights

$$Q: \ \exists X \exists Y accessLevel(X,Y) \wedge blackListed(X)$$

for which we can add a constraint

$$r_Q: \ \leftarrow accessLevel(X,Y), blackListed(X)$$

to the program Π so that Q follows from Π under stable models iff $\Pi \cup \{r_Q\}$ has no stable model.

Current literature has seen increasing interest in representing ontological knowledge by Datalog programs augmented by existential quantification in rule heads. For example, that every professor teaches at least one course can be represented by a rule, $\exists Y isInstructorOf(X,Y) \leftarrow prof(X)$, which corresponds to an axiom in description logic (DL): $prof \sqsubseteq \exists isInstructorOf$. However, the works in this direction either assume that ontological knowledge is not defeasible (Alviano et al. 2012), or only treat stratified negation (Calì et al. 2009) or well-founded negation (Gottlob et al. 2012) for non-disjunctive Datalog programs.

Despite the potential interest, E-disjunctive programs have not been closely studied. Little is known about their properties. For example, we would be interested in a simple yet intuitive definition of stable models for these programs. Such a definition arguably helps make the stable model semantics more accessible thus facilitating development of applications. Another issue is decidability. Non-trivial E-disjunctive programs fall outside of any known decidable classes of *safe logic programs* (Cabalar et al. 2009; Bartholomew and Lee 2010).

In this paper, we study the properties of E-disjunctive programs. We report three results. First, there is indeed an interesting definition of stable models for these programs, which is simple and intuitive, and can be given literally in one sentence. Second, we show that stable models of these programs can be characterized by *progression*, previously formulated for disjunctive programs (Zhou and Zhang 2011). This provides a rather direct characterization of *level mapping justification* for stable models (Fages 1994), which is useful in several aspects. For example, as shown in (Zhou and Zhang 2011), a level mapping enables a translation from disjunctive programs to Satisfiability Modulo Theories (SMT) (Nieuwenhuis et al. 2006), thus providing a basis for an alternative implementation. Another utility of progression is that it offers a direct proof that a safe disjunctive program possesses the so-called *small predicate property* (Lee et al. 2008), which can be extended to check whether an E-disjunctive program has the same property.

Next, we tackle the issue of decidability. Unlike safe disjunctive programs, with EQ added to rule heads the stable model existence problem becomes undecidable. One way to prove this is by extending an existing proof that shows query answering for conjunctive queries with non-guarded Datalog$^\exists$ programs (the class of Datalog programs that consist of positive rules with EQ in rule heads) is in general undecidable (Calì et al. 2008). Here we present a different proof based on an E-disjunctive program encoding of the Domino problem (Berger 1966) so that the latter is reduced to the stable model existence problem. This proof has an extra merit in that it allows us to derive an additional result - deciding the existence of a finite stable model for E-disjunctive programs where universal variables are safe is also undecidable. Finally, we identify a decidable fragment by exploring a new notion of stratification over existential quantification (but not over default negation), which can be seen as the first step in generalizing the notion of semi-safety for disjunctive programs (Cabalar et al. 2009; Bartholomew and Lee 2010). The paper concludes with a discussion on related work and some remarks, as well as pointers to future work. Missing proofs of the key results can be found in Appendix.

2 Preliminaries

We consider a first-order language $\mathcal{L}$ without function symbols but with the negation-as-failure operator, not, along with a countably infinite set of constants $\mathcal{C}$, a countable set of predicate constants (predicates in short) $\mathcal{P}$, and a countable set of individual variables $\mathcal{V}$, which are denoted by upper case letters.

Denote by $\tau_\mathcal{L}$ the signature of $\mathcal{L}$. An *atom* on $\tau_\mathcal{L}$ is of the form $p(t_1, ..., t_n)$, abbreviated as $p(\mathbf{t})$, where $p \in \mathcal{P}$ is an n-ary predicate and $\mathbf{t}$ an n-tuple on $\mathcal{C} \cup \mathcal{V}$. We write $p(\mathbf{X})$ if $\mathbf{X}$ if a tuple of n distinct variables. Let $\alpha = p(t_1, ..., t_n)$, we define $pred(\alpha) = p$ and $arg(\alpha, i) = t_i$ $(1 \leq i \leq n)$. Atoms and negated atoms are called *literals*. A negated atom can be either in the form $\mathsf{not}\ \alpha$ or in the form $\neg\alpha$ (under the language of general stable models $\mathsf{not}\ a$ will be identified with $\neg a$). A *ground atom* $p(t_1, ..., t_n)$ on a set D is an atom where $t_i \in D$ for all i. Let $\mathcal{M}$ be a structure of $\tau_\mathcal{L}$. $Dom(\mathcal{M})$ denotes the domain of $\mathcal{M}$, $c^\mathcal{M}$ denotes the element in $Dom(\mathcal{M})$ that is mapped in $\mathcal{M}$ from constant $c \in \mathcal{C}$, and $p^\mathcal{M}$ the n-ary relation on $Dom(\mathcal{M})$, which is assigned in $\mathcal{M}$ for the n-ary predicate $p \in \mathcal{P}$. Notationally, we also write $p(\mathbf{t}) \in \mathcal{M}$ to mean $\mathbf{t} \in p^\mathcal{M}$. In this way, $\mathcal{M}$ also denotes the set of ground atoms $p(\mathbf{t})$ such that $\mathbf{t} \in p^\mathcal{M}$, for all $p \in \mathcal{P}$. We define $\mathcal{M}^- = \{\neg\alpha \mid \alpha \text{ is a ground atom on } Dom(\mathcal{M}) \text{ and } \alpha \notin \mathcal{M}\}$.

Given a structure $\mathcal{M}$ and a ground atom α on $Dom(\mathcal{M})$, $\mathcal{M}$ *satisfies* α, denoted as $\mathcal{M} \models \alpha$, iff $\alpha \in \mathcal{M}$, and $\mathcal{M}$ *satisfies* $\mathsf{not}\ \alpha$, denoted as $\mathcal{M} \models \mathsf{not}\ \alpha$, iff $\alpha \notin \mathcal{M}$. The definition extends to conjunctions and disjunctions of ground atoms as usual. For an existentially quantified formula $\exists Y F$, where F is a first-order formula in which Y is the only free variable, $\mathcal{M} \models \exists Y F$ iff $\mathcal{M} \models F[Y/e]$ for some $e \in Dom(\mathcal{M})$, where $F[Y/e]$ is the formula obtained from F by substituting occurrences of Y in F with e.

Let $\mathcal{M}$ be a structure, X a set of ground atoms on $Dom(\mathcal{M})$, and F a conjunction of ground atoms on $Dom(\mathcal{M})$. In this paper, we will use the notation $X \cup \mathcal{M}^- \models F$ to mean that the literal set $X \cup \mathcal{M}^-$ *logically entails* F. Note that we use the same relational symbol $\models$ to express two different concepts, satisfaction and logic entailment. Above, while $X \cup \mathcal{M}^- \models F$ expresses an entailment relation, that $\mathcal{M}$ *satisfies* F amounts to the entailment relation $\mathcal{M} \cup \mathcal{M}^- \models F$.

Let $\mathcal{M}$ be a structure and $\mathcal{H}$ a set of ground atoms on $Dom(\mathcal{M})$. By $\mathcal{H} \subseteq \mathcal{M}$, we mean for all $p(\mathbf{t}) \in \mathcal{H}$, $\mathcal{M} \models p(\mathbf{t})$, and by $\mathcal{M}' = \mathcal{M} \cup \mathcal{H}$, we mean a structure $\mathcal{M}'$ with the same signature, domain and constant mapping as $\mathcal{M}$, but for any predicate p, $\mathbf{t} \in p^{\mathcal{M}'}$ iff $p(\mathbf{t}) \in \mathcal{M} \cup \mathcal{H}$. That a structure is denoted as a set also allows us to compare two structures $\mathcal{M}_1$ and $\mathcal{M}_2$ of the same signature, which are identical everywhere except for the mappings of predicates. In particular, $\mathcal{M}_1 \subseteq \mathcal{M}_2$ means for all $p(\mathbf{t})$ if $\mathcal{M}_1 \models p(\mathbf{t})$ then $\mathcal{M}_2 \models p(\mathbf{t})$; and $\mathcal{M}_1 \subset \mathcal{M}_2$ is defined as $\mathcal{M}_1 \subseteq \mathcal{M}_2$ and $\mathcal{M}_2 \not\subseteq \mathcal{M}_1$. We will use $\mathbf{X}$ (resp. $\mathbf{Y}$) to denote a vector of variables.

An *assignment* of a structure $\mathcal{M}$ is a mapping of individual variables to elements in $Dom(\mathcal{M})$, and if ϑ is an assignment of $\mathcal{M}$, $\vartheta|_{\mathbf{X}}$ denotes the sub-assignment of ϑ restricted to the variables in $\mathbf{X}$. Let F be a formula, $\mathcal{M}$ a structure of the signature of F, and ϑ an assignment of $\mathcal{M}$. $F\vartheta$ denotes the formula obtained from F by replacing all occurrences of free variables in F mentioned in ϑ, with their mapped elements.

3 Justified Stable Models

E-disjunctive programs are finite sets of rules of the form

$$\exists \mathbf{Y}\ \alpha_1; ...; \alpha_m \leftarrow \beta_1, ..., \beta_k, \mathsf{not}\ \gamma_1, ..., \mathsf{not}\ \gamma_n \tag{1}$$

where α_i, β_i and γ_i are atoms, and variables in $\mathbf{Y}$ may appear only in α_i $(1 \leq i \leq m)$.

Given an E-disjunctive program Π, we denote by $\tau(\Pi)$ the signature of Π which includes all predicates and constants occurring in Π.

Let r be a rule of the form (1). In the sequel, if not said otherwise we will use $\mathbf{X}$ to denote the free variables (also called $\forall$-variables) in r and $\mathbf{Y}$ to denote the existential variables ($\exists$-variables). We may write r as $\exists \mathbf{Y} Head(r) \leftarrow Pos(r), Neg(r)$, where $Head(r) = \{\alpha_1, ..., \alpha_m\}$ is called the *head* of the rule, $Pos(r) = \{\beta_1, ..., \beta_k\}$ the *positive body* and $Neg(r) = \{\mathsf{not}\ \gamma_1, ..., \mathsf{not}\ \gamma_n\}$ the *negative body* of the rule, respectively. Note that a rule of this form is shorthand of the rule with the full quantification, $\forall \mathbf{X} \exists \mathbf{Y} Head(r) \leftarrow Pos(r), Neg(r)$. We also define $Body(r) = Pos(r) \cup Neg(r)$. We may write a rule with empty head as a *constraint*. In addition, we assume rules in a program are *standardized apart* so they share no variables.

Let Π be an E-disjunctive program and $\mathcal{M}$ a structure of $\tau(\Pi)$. We say that $\mathcal{M}$ is a *model* of Π if $\mathcal{M}$ satisfies every rule in Π.

We now define the notion of *deductive closure*.

Definition 1
Let Π be an E-disjunctive program and $\mathcal{M}$ a structure of $\tau(\Pi)$. A *deductive closure of* Π *and* $\mathcal{M}$ is a minimal set X of ground atoms on $Dom(\mathcal{M})$ satisfying the condition: for any rule $r \in \Pi$ and any assignment η of $\mathcal{M}$, if $X \cup \mathcal{M}^- \models Body(r)\eta$, then for some assignment ϑ of $\mathcal{M}$ and $\alpha \in Head(r)$, $(\alpha\eta|_{\mathbf{X}})\vartheta \in X$.

Note that $\mathcal{M}^-$ is fixed in determining if X is minimal. Let us denote by $\Omega(\Pi, \mathcal{M})$ the set of all deductive closures of Π and $\mathcal{M}$. Note that this set is non-empty as the set of all ground atoms on $Dom(\mathcal{M})$ satisfies the condition and it contains minimal subsets that satisfy the condition.

Definition 2
Let Π be an E-disjunctive program and $\mathcal{M}$ a structure of $\tau(\Pi)$. $\mathcal{M}$ is a *justified stable model* of Π if $\mathcal{M}$ is a deductive closure of Π and $\mathcal{M}$.

Example 1
Consider the following program Π:

$$p(a); p(b) \leftarrow \qquad p(a) \leftarrow p(b) \qquad p(d) \leftarrow \mathsf{not}\ p(b)$$

$M_1 = \{p(a), p(d)\}$ is the only Herbrand stable model of Π, which is also a justified stable model of Π. Note that $M_2 = \{p(a), p(b)\}$ is not a justified stable model of Π, since it is not a minimal set satisfying the deductive closure property, as the proper subset $\{p(a)\}$ satisfies the condition.

Example 2
Let Π be the following program:

$$\begin{array}{l} h(a) \leftarrow \\ \exists Y\ p(X,Y); q(X,Y) \leftarrow \mathsf{not}\ h(X) \\ \exists Y\ p(X,X); q(Y,Y) \leftarrow h(X) \end{array}$$

Let $\mathcal{M}$ be a structure of $\tau(\Pi)$ where $Dom(\mathcal{M}) = \{1,2\}$, $a^{\mathcal{M}} = 1$, and $\mathcal{M} = \{h(1), q(2,2)\}$. One can verify that $\mathcal{M}$ is a justified stable model of Π.

For clarity, above we defined justified stable model using two definitions. In fact, we can combine the two to arrive at an equivalent but simpler definition, literally in one sentence.

Definition 3
Let Π be an E-disjunctive program and $\mathcal{M}$ a structure of $\tau(\Pi)$. $\mathcal{M}$ is a justified stable model of Π if $\mathcal{M}$ is a minimal set X satisfying the condition: for any $r \in \Pi$ and any assignment η of $\mathcal{M}$, if $X \cup \mathcal{M}^- \models Body(r)\eta$, then for some assignment ϑ of $\mathcal{M}$ and $\alpha \in Head(r)$, $(\alpha\eta|_{\mathbf{X}})\vartheta \in X$.

Before ending this section, let us show a compact encoding of a generalized problem of strategic companies (Cadoli et al. 1997; Leone et al. 2006). It is well-known that computing strategic companies is Σ_2^P-complete.

Example 3
In the strategic companies problem, there is a collection $C = \{c_1, ..., c_m\}$ $(m \geq 1)$ of companies, each producing some goods from the set G of all goods. Each company $c_i \in C$ is possibly controlled by a set of owner companies $O_i \subseteq C$. A set $C' \subseteq C$ is *a strategic set* if it is minimal among all the sets satisfying the following conditions: (1) The companies in C' produce all goods in G, and (2) The companies in C' are closed under the controlling relation, i.e. if $O_i \subseteq C'$ for some $1 \leq i \leq m$ then $c_i \in C'$. In (Cadoli et al. 1997; Leone et al. 2006), it is assumed that each product is produced by at most two companies and each company is jointly controlled by at most three other companies. We generalize the original problem by changing the first assumption to "Each product is produced by one or more companies".[1]

To encode the generalized problem, we can modify the encoding given in (Leone et al. 2006) slightly, resulting in the following program:

$$\begin{array}{l} \exists Z\ \ strat\_from(Z,X) \leftarrow prod\_by(X,Y) \\ \leftarrow strat\_from(Y,X), \mathsf{not}\ prod\_by(X,Y) \\ strat(Y) \leftarrow strat\_from(Y,X) \\ strat(W) \leftarrow contr\_by(W,X,Y,Z), strat(X), strat(Y), strat(Z) \end{array}$$

Above, $prod\_by(X,Y)$ means "Company Y produces good X"; $strat\_from(Y,X)$ says "Company Y is strategic because of good X". A problem instance also contains facts specified by predicates $prod\_by/2$ and $contr\_by/4$. Given a problem instance, a company c is strategic if $strat(c)$ is in some stable model of the program encoding the instance. It is interesting to see that with existential quantification a Σ_2^P-hard problem is encoded without disjunction.

[1] Here we omit the case where a product is produced by no company, as it is easy to check.

4 General Stable Models

We show that for E-disjunctive programs justified stable models are precisely *general stable models* (Ferraris et al. 2011) (also see (Lin and Zhou 2011)).

Let $\mathbf{p}$ and $\mathbf{u}$ be two lists of distinct predicate constants $(p_1, \ldots, p_n)$ and $(u_1, \ldots, u_n)$ with matching arities. By $\mathbf{u} \leq \mathbf{p}$, we mean $\forall \mathbf{x}(u_i(\mathbf{x}) \rightarrow p_i(\mathbf{x}))$ $(1 \leq i \leq n)$, and $\mathbf{u} < \mathbf{p}$ is defined by $(\mathbf{u} \leq \mathbf{p}) \wedge \neg(\mathbf{p} \leq \mathbf{u})$. If $\mathbf{p}$ (resp. $\mathbf{u}$) is a singleton, we may just write p (resp. u).

For any first-order sentence F, $\mathrm{SM}_{\mathbf{p}}[F]$ denotes the second-order sentence, $F \wedge \neg\exists\mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^\star(\mathbf{u}))$, where $F^\star(\mathbf{u})$ is defined recursively:

- $p_i(\mathbf{t})^\star = u_i(\mathbf{t})$ for any tuple $\mathbf{t}$ of terms;
- $F^\star = F$ for any atomic formula F that does not contain members of $\mathbf{p}$;
- $(F\ Op\ G)^\star = F^\star\ Op\ G^\star$, where Op is either $\wedge$ or $\vee$;
- $(F \rightarrow G)^\star = (F^\star \rightarrow G^\star) \wedge (F \rightarrow G)$;
- $(Qx\, F)^\star = Qx\, F^\star$, where Q is either $\forall$ or $\exists$.

For any sentence F, a $\mathbf{p}$-*stable model* of F is a structure of $\tau(F)$ that satisfies $\mathrm{SM}_{\mathbf{p}}[F]$. In this paper, we assume that $\mathbf{p}$ in $\mathrm{SM}_{\mathbf{p}}[F]$ is the set of all predicates in the signature of F.[2] We then just call these models *stable models*.

Given an E-disjunctive program Π and a rule $r \in \Pi$ of the form (1), we denote by $\pi(r)$ the following formula

$$\forall\mathbf{X}\exists\mathbf{Y}\beta_1 \wedge \ldots \wedge \beta_k \wedge \neg\gamma_1 \wedge \ldots \wedge \neg\gamma_n \rightarrow \alpha_1 \vee \ldots \vee \alpha_m \qquad (2)$$

where $\mathbf{X}$ is the list of free variables in r and for any formula Φ, $\neg\Phi$ is shorthand for $\Phi \rightarrow \bot$. By $\pi(\Pi)$, we mean the conjunction of $\pi(r)$ for all $r \in \Pi$.

An example illustrating the definition of general stable models can be found in Appendix.

Theorem 1
Let Π be an E-disjunctive program and $\mathcal{M}$ a structure of $\tau(\Pi)$. $\mathcal{M}$ is a justified stable model of Π iff $\mathcal{M}$ is a stable model of $\pi(\Pi)$.

Proof
We give a proof for the $\Rightarrow$ part, the proof for the $\Leftarrow$ part is similar. Suppose $\mathcal{M}$ is a justified stable model of Π, and we show that $\mathcal{M}$ satisfies $\pi(\Pi) \wedge \neg\exists\mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge \pi(\Pi)^\star(\mathbf{u}))$. Clearly, $\mathcal{M} \models \pi(\Pi)$. Towards a contradiction, assume $\mathcal{M} \not\models \neg\exists\mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge \pi(\Pi)^\star(\mathbf{u}))$, i.e., $\mathcal{M} \models \exists\mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge \pi(\Pi)^\star(\mathbf{u}))$, where

$$\pi(\Pi)^\star(\mathbf{u}) = \textstyle\bigwedge_{r\in\Pi} \forall\mathbf{X}\exists\mathbf{Y}(B_r^\star \rightarrow H_r^\star) \wedge \pi(r)$$

where $\forall\mathbf{X}\exists\mathbf{Y}(B_r^\star \rightarrow H_r^\star)$ is shorthand of (2). Assume, WLOG, that a rule r is in such a form that $\forall\mathbf{X}\exists\mathbf{Y}(B_r^\star \rightarrow H_r^\star)$ is of the form (3) below.

$$\forall\mathbf{X}\exists\mathbf{Y}\beta^\star \wedge (\gamma^\star \rightarrow \bot) \wedge (\gamma \rightarrow \bot) \rightarrow \alpha_1^\star \vee \ldots \vee \alpha_m^\star \qquad (3)$$

Then, there is a structure $\mathcal{M}^*$ of $\tau(\pi(\Pi)^\star(\mathbf{u}))$ such that $\mathcal{M}^* \models \pi(\Pi)^\star(\mathbf{u})$, where $p_i^{\mathcal{M}^*} = p_i^{\mathcal{M}}$ for all $p_i \in \mathbf{p}$ and the mappings for predicates in $\mathbf{u}$ satisfy $\mathbf{u} < \mathbf{p}$. Let $\mathcal{N}$ denote the interpretation of predicates in $\mathbf{u}$. That is, $\mathcal{M}^* = \mathcal{M} \cup \mathcal{N}$. Further, let $\mathcal{M}'$ be a structure of $\tau(\Pi)$, which is the same as $\mathcal{M}$ except for the mappings of predicates in $\mathbf{p}$, which is defined as $\mathcal{M}' = \{p_i(\mathbf{t}) \mid u_i(\mathbf{t}) \in \mathcal{N}\}$.

[2] Here, we assume all predicates p appearing in a program are *intensional*, e.g., by adding a rule $p(\mathbf{X}) \leftarrow p(\mathbf{X})$ to it.

As $\mathbf{u} < \mathbf{p}$, we have $\mathcal{M}' \subset \mathcal{M}$. Now, for any assignment η of $\mathcal{M}$ and any rule $r \in \Pi$, suppose $\mathcal{M}' \cup \mathcal{M}^- \models Body(r)\eta$. It follows from the def. of $\mathcal{M}^*$ and that of $\mathbf{u} < \mathbf{p}$ that $\mathcal{M}^* \models B_r^\star \eta$. As $\mathcal{M}^*$ is a model of (3), for some assignment ϑ and $\alpha^\star \in \{\alpha_1^\star, ..., \alpha_m^\star\}$, we have $(\alpha^\star \eta|_{\mathbf{X}})\vartheta \in \mathcal{M}^*$; then, α, the counterpart of $\alpha^\star$, is in $Head(r)$, and it follows from the definition of $\mathcal{M}'$ that we have $(\alpha\eta|_{\mathbf{X}})\vartheta \in \mathcal{M}'$. Thus, $\mathcal{M}'$ is a deductive closure of Π and $\mathcal{M}$. From $\mathcal{M}' \subset \mathcal{M}$, we know that $\mathcal{M}$ is not a minimal set satisfying the deductive closure condition and is therefore not a justified stable model of Π. A contradiction. $\square$

5 Progression Characterization

The idea of *progression* (Zhou and Zhang 2011) is to characterize a stable model of a disjunctive logic program by a fixpoint construction that nondeterministically chooses a minimal *hitting set* at each step. Intuitively, this means that every ground atom true in a stable model of the given program is justified by a non-circular derivation by rules in the program, a property known as *level mapping justifications* for normal logic programs (Fages 1994). In this section, we generalize this characterization to E-disjunctive programs.

Definition 4
Let S be a set and $\Phi = \{S_1, ..., S_i, ...\}$ a collection of sets such that $S_i \subseteq S$, for all i. A subset $H \subseteq S$ is said to be a *hitting set* of Φ if for all i, $H \cap S_i \neq \emptyset$. Furthermore, H is said to be a *minimal hitting set* of Φ if H is a hitting set of Φ and there is no $H' \subset H$ such that H' is also a hitting set of Φ.

In the following, let Π be an E-disjunctive program and $\mathcal{M}$ a structure of $\tau(\Pi)$. Also, let Ψ be the set of all assignments of $\mathcal{M}$, and $\mathbf{X}$ and $\mathbf{Y}$ denote the $\forall$-variables and $\exists$-variables, respectively, in Π.

Definition 5
An *evolution sequence of* Π *based on* $\mathcal{M}$, denoted as $\sigma_{\mathcal{M}}(\Pi)$, is a sequence $\sigma^0_{\mathcal{M}}(\Pi), \cdots, \sigma^t_{\mathcal{M}}(\Pi)$, $\cdots$, of structures of $\tau(\Pi)$, defined inductively as follows

1. $\sigma^0_{\mathcal{M}}(\Pi) = \mathcal{E}$, where $\mathcal{E}$ is the structure of $\tau(\Pi)$ in which all interpretations of predicates are the empty set;
2. $\sigma^{t+1}_{\mathcal{M}}(\Pi) = \sigma^t_{\mathcal{M}}(\Pi) \cup H^t$, where there exists $H^t \subseteq \mathcal{M}$ such that it is a minimal hitting set of the collection Φ^t of the following sets:
$$\bigcup_{\theta \in \Psi} (Head(r)\eta|_{\mathbf{X}})\theta|_{\mathbf{Y}} \tag{4}$$
where r is a rule in Π and η an assignment of $\mathcal{M}$ such that $(4) \cap \sigma^t_{\mathcal{M}}(\Pi) = \emptyset$, $\sigma^t_{\mathcal{M}}(\Pi) \models Pos(r)\eta$, and $\mathcal{M} \models Neg(r)\eta$; and $\sigma^{t+1}_{\mathcal{M}}(\Pi) = \sigma^t_{\mathcal{M}}(\Pi)$ if H^t does not exist.

We denote $\sigma^\infty_{\mathcal{M}}(\Pi) = \bigcup_{i=0}^\infty \sigma^i_{\mathcal{M}}(\Pi)$.

As commented in (Zhou and Zhang 2011) (also see (You et al. 2012)), the basic idea in an evolution sequence is that we start with $\sigma^0_{\mathcal{M}}(\Pi)$ and progress by nondeterministically selecting a minimal hitting set at each step, from instantiated heads of rules whose bodies are implied by $\mathcal{M}^-$ and the set of atoms already derived, under an assignment η. The condition $H^t \subseteq \mathcal{M}$ ensures that the construction is guarded, and the condition $(4) \cap \sigma^i_{\mathcal{M}}(\Pi) = \emptyset$ only allows the instances of rule-heads that are not already satisfied to be considered.

However, there is a critical difference between this definition and that of (Zhou and Zhang 2011) in the set Φ^t of sets from which a minimal hitting set is selected, in that a predicate with $\exists$-variables in a rule head may have multiple instantiations on the $\exists$-variables.

Example 4

Consider again the program Π in Example 2, where $\mathcal{M} = \{h(1), q(2,2)\}$, with $Dom(\mathcal{M}) = \{1, 2\}$ and $a^{\mathcal{M}} = 1$. $\mathcal{M}$ can be constructed by an evolution sequence as follows. We start with $\sigma^0_{\mathcal{M}}(\Pi) = \emptyset$, and as $\Phi^0 = \{\{h(1)\}, \{p(2,1), p(2,2), q(2,1), q(2,2)\}\}$ of which $\mathcal{H}^0 = \{h(1), q(2,2)\}$ is a minimal hitting set as well as guarded, we get $\sigma^1_{\mathcal{M}}(\Pi) = \{h(1), q(2,2)\}$. Now as $\sigma^2_{\mathcal{M}} = \sigma^1_{\mathcal{M}}$ and so on, $\mathcal{M}$ coincides with the fixpoint of the evolution sequence.

Lemma 1

An evolution sequence ρ of Π based on $\mathcal{M}$ always exists, and for any evolution sequence ρ of Π based on $\mathcal{M}$, $\rho^\infty_{\mathcal{M}}(\Pi) \subseteq \mathcal{M}$.

By definition, an evolution sequence ρ of Π based on $\mathcal{M}$ begins with $\rho^0_{\mathcal{M}}(\Pi) = \emptyset$. As such a sequence is increasing, it reaches a fixpoint when no more atoms can be added. The conclusion $\rho^\infty_{\mathcal{M}}(\Pi) \subseteq \mathcal{M}$ is because at each stage t, we can only add a hitting set $H^t \subseteq \mathcal{M}$ of Φ^t. The fixpoint reached by an evolution sequence can be classified to two categories: let us call them *premature fixpoint* and *normal fixpoint* respectively. The former refers to the situation where the unsatisfiability of a rule is witnessed in the progression of the sequence. For example, consider $\Pi = \{c \leftarrow a, \text{not } d;\ b \leftarrow a;\ a \leftarrow\}$ and the structure $\mathcal{M} = \{a, b\}$. We have $\rho^0_{\mathcal{M}}(\Pi) = \emptyset$, $\rho^1_{\mathcal{M}}(\Pi) = \{a\}$, and $\Phi^2 = \{\{c\}, \{b\}\}$. Since there is no hitting set H^2 of Φ^2 satisfying $H^2 \subseteq \mathcal{M}$, we have $\rho^\infty_{\mathcal{M}}(\Pi) = \rho^1_{\mathcal{M}}(\Pi) = \{a\}$, where the unsatisfiability of the first two rules is witnessed.

On the other hand, a normal fixpoint is one without witnessing the violation of any rule. This is the case when $\mathcal{M}$ is a model of Π. At each stage t, for any rule $r \in \Pi$ and any assignment η of $\mathcal{M}$, if $Body(r)\eta$ is entailed by $\rho^t_{\mathcal{M}}(\Pi) \cup \mathcal{M}^-$ then at least one instance of $Head(r)\eta|_{\mathbf{x}}$ is either in $\rho^t_{\mathcal{M}}(\Pi)$ or in the selected minimal hitting set H^{t+1}, which is guaranteed to exist. However, if $\mathcal{M}$ is a model but not a stable model of Π, any evolution sequence based on $\mathcal{M}$ will reach a structure $\mathcal{X}$ as the fixpoint such that $\mathcal{X} \subset \mathcal{M}$.

Theorem 2

Let Π be an E-disjunctive program and $\mathcal{M}$ a structure of $\tau(\Pi)$. $\mathcal{M}$ is a stable model of Π iff for all evolution sequences σ of Π based on $\mathcal{M}$, $\sigma^\infty_{\mathcal{M}}(\Pi) = \mathcal{M}$.

The proof of this theorem can be found in Appendix.

One utility of the progression characterization is that it offers a direct proof that safe disjunctive programs possess the so-called *small predicate property* (SPP) (Lee et al. 2008), which intuitively says that the extensions of predicates in any stable model of a given sentence Π consist of tuples of elements that are mapped from the constants appearing in Π. The SPP offers a condition for decidability, and a bridge in an implementation for computing stable models of these programs.

More formally, given a finite set of constants $\mathbf{c}$ and a first-order sentence F, the SPP for F is defined by the conjunction of $\forall \mathbf{x}\ p(\mathbf{x}) \rightarrow in_{\mathbf{c}}(\mathbf{x})$, for each predicate p occurring in F, where $in_{\mathbf{c}}(x_1, ..., x_m)$ stands for $\bigwedge_{1 \le j \le m} \bigvee_{c \in \mathbf{c}} x_j = c$. Let $c(F)$ be the set of constant symbols appearing in F, and we denote the SPP for F by $\mathbf{SPP}_{c(F)}$.

Definition 6
A rule of the form (1) is called $\forall$-*safe* if every $\forall$-variable appearing in its head appears in at least one positive literal of its body. An E-disjunctive program is $\forall$-*safe* if every rule in it is $\forall$-safe.

A $\forall$-safe E-disjunctive program without existential quantification is just a *semi-safe disjunctive program* (Cabalar et al. 2009). The following result is well-known. Here we offer a direct proof.

Proposition 1
Let Π be a $\forall$-safe E-disjunctive logic program, where no EQ occurs. Then for any stable model $\mathcal{A}$ of $\tau(\Pi)$, $\mathcal{A} \models \mathbf{SPP}_{c(\Pi)}$.

Proof
Let $\mathcal{A}$ be a stable model of Π. By Theorem 2, there is an evolution sequence $\sigma_{\mathcal{A}}(\Pi)$ such that $\sigma^{\infty}_{\mathcal{A}}(\Pi) = \mathcal{A}$, and this is the case for all evolution sequences of Π based on $\mathcal{A}$. Suppose $\mathcal{A} \not\models \mathbf{SPP}_{c(\Pi)}$. Then $\exists p(\mathbf{t}) \in \mathcal{A}$ s.t. for some j, $arg(p(\mathbf{t}), j) = u$ and there is no $e \in c(\Pi)$ s.t. $e^{\mathcal{M}} = u$. By induction on the construction of $\sigma^{\infty}_{\mathcal{A}}(\Pi)$, at any stage t, because Π is $\forall$-safe and no EQ occurs in Π, a selected minimal hitting set H^t is always finite and for every $q(\mathbf{s}) \in H^t$ and every position j within the arity of predicate q, $arg(q(\mathbf{s}), j) = e^{\mathcal{A}}$, for some $e \in c(\Pi)$. Thus the atom $p(\mathbf{t}) \in \mathcal{A}$ mentioned above does not exist. Therefore, we must have $\mathcal{A} \models \mathbf{SPP}_{c(\Pi)}$. $\Box$

By a similar proof, this result can be generalized to $\forall$-safe E-disjunctive programs, if each $\exists$-variable is guaranteed to be instantiated to a constant appearing in the given program. For example, the k-colorability program given in the Introduction is $\forall$-safe. By the third rule, if the predicate $set/2$ is in a stable model, it can only be in the form $set(\_, e)$ where e is mapped from a constant representing a color. It is then easy to see that the program, along with an input graph and colors, possesses the small predicate property.

6 Decidability

In this paper, we will be focusing on the problem of stable model existence, as ontology query answering under stable models can be reduced to a problem of model checking: q is true in every stable model of Π iff $\Pi \wedge \neg q$ has no stable model, where q is the existential closure of a conjunctive query (i.e., a conjunction of atoms), so $\Pi \wedge \neg q$ is equivalent to an E-disjunctive program under the stable model semantics.

Though the stable model existence problem for safe disjunctive programs is decidable, the problem becomes undecidable when EQ is allowed in rule heads. Below, we present a proof based on the idea of reducing the Domino problem (Berger 1966) to the stable model existence problem. This proof is interesting as it can be extended easily to show an additional result - deciding the existence of a finite stable model for $\forall$-safe E-disjunctive programs is also undecidable.[3]

Let $\mathbb{N}$ be the set of all non-negative integers and $\mathbb{Z}_k$ the set of non-negative integers that are less than k for all integers $k \in \mathbb{N}$. A *domino system* is defined to be a triple $D = (k, H, V)$, where $k \in \mathbb{N}$ and H, V are two subsets of $\mathbb{Z}_k \times \mathbb{Z}_k$. A *tiling of D for* $\mathbb{N} \times \mathbb{N}$ is a function τ from $\mathbb{N} \times \mathbb{N}$ into $\mathbb{Z}_k$ such that $(i, j) \in H$ if $\tau(m, n) = i$ and $\tau(m+1, n) = j$ and that $(i, j) \in V$ if $\tau(m, n) = i$ and $\tau(m, n+1) = j$. A tiling τ of D for $\mathbb{N} \times \mathbb{N}$ is said to be *period* if there are some

[3] The proof can also be adopted, with a small modification, to show that, for some decidable classes of Datalog with existential rules, such as *sticky* sets (Calì et al. 2012) and the *shy* fragment (Leone et al. 2012), the addition of default negation causes the fragment to be undecidable. The details are beyond the scope of this paper.

integers $h, v > 0$ such that $\tau(i+h, j) = \tau(i, j+v) = \tau(i, j)$ for all $i, j \in \mathbb{N}$. It is well-known that there is no algorithm to check whether or not a domino system D has a (period) tiling for $\mathbb{N} \times \mathbb{N}$ (Berger 1966; Gurevich and Koryakov 1972).

Proposition 2

There is no algorithm to check whether or not a $\forall$-safe E-disjunctive program has a (finite) stable model.[4]

Proof

Given a domino system $D = (k, H, V)$, construct a program Π_D as follows:

1. $num(c) \leftarrow$
2. $\exists Y succ(X, Y) \leftarrow num(X)$
3. $num(Y) \leftarrow succ(X, Y)$
4. $\leftarrow p_i(X, Y), p_j(X, Y) \qquad (0 \leq i < j < k)$
5. $p_0(X, Y); \ldots; p_{k-1}(X, Y) \leftarrow num(X), num(Y)$
6. $\leftarrow p_i(X,Y), p_j(Z,Y), succ(X,Z)\ (0 \leq i,j < k, (i,j) \notin H)$
7. $\leftarrow p_i(X,Y), p_j(X,Z), succ(Y,Z)\ (0 \leq i,j < k, (i,j) \notin V)$

This program is clearly $\forall$-safe. Let $\mathbf{p}$ be the tuple of all predicates appearing in Π_D. We claim that D has a tiling for $\mathbb{N} \times \mathbb{N}$ iff Π_D has a stable model (which is also a minimal model as no default negation occurs). If this is true, according to (Berger 1966), it is undecidable to check whether or not a $\forall$-safe E-disjunctive program has a stable model. We now prove this claim.

($\Rightarrow$) Suppose $\tau : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{Z}_k$ is a tiling of D for $\mathbb{N} \times \mathbb{N}$. We need to show that Π_D has a minimal/stable model. Let $\mathcal{A}$ be a structure with the domain $\mathbb{N}$ and interpreting c as 0; interpreting num as the set $\mathbb{N}$; interpreting $succ$ as the relation $\{(i, i+1) : i \in \mathbb{N}\}$; and for $0 \leq i < n$, interpreting p_i as the relation $\{(m, n) : \tau(m, n) = i\}$. It is not difficult to verify that $\mathcal{A}$ is a minimal/stable model of Π_D.

($\Leftarrow$) Suppose $\mathcal{A}$ is a minimal/stable model of Π_D. Let $(a_i)_{i \in \mathbb{N}}$ be a sequence of elements in $Dom(\mathcal{A})$ such that $a_0 = c^{\mathcal{A}}$ and that $(a_i, a_{i+1}) \in succ^{\mathcal{A}}$ for all $i \in \mathbb{N}$. The existence of such a sequence is guaranteed by the rule 2. Let τ be a function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{Z}_k$ such that $\tau(m, n) = i$ iff $(a_m, a_n) \in p_i^{\mathcal{A}}$ for all $m, n \in \mathbb{N}$ and $0 \leq i < k$. According to the rules 4 and 5, τ is well-defined. By the rules 6 and 7, τ should be a tiling of D for $\mathbb{N} \times \mathbb{N}$.

For the case of finite stable models, it suffices to show that D has a period tiling for $\mathbb{N} \times \mathbb{N}$ iff Π_D has a finite stable model. This can be obtained by a slight modification of the proof for the arbitrary case. $\square$

Note that, for each D, Π_D can be rewritten without disjunction by shifting disjunctions to the body of rules. The soundness follows from the splitting lemma (Ferraris et al. 2009) and the fact that the program consists of rules which are head-cycle-free.

[4] The same claim can be proved by adopting the proof of Theorem 15 of (Calì et al. 2008), in the following way. First, query answering with a Datalog$^{\exists}$ program for conjunctive queries can be reduced to a problem of model checking. Given the non-guarded Datalog$^{\exists}$ program Π used in the above proof (with a minor translation to make it a $\forall$-safe E-program) and a conjunctive query, let Π' be the resulting program for model checking, which is a $\forall$-safe E-program. Then, one needs to show a lemma that Π' has a classic model iff Π' has a stable model (in this case a minimal model). This is because in general the left hand side does not imply the right hand side.

6.1 A Decidable Fragment

In general, a $\forall$-safe E-disjunctive program may not possess the SPP. In this section, we identify a fragment of E-disjunctive programs, which is decidable based on the *small model property*, i.e., the size of any stable model of a program in this class is bounded.

Definition 7
Let Π be an E-disjunctive program and $\mathbf{p}$ the tuple of all predicates occurring in Π. Then Π is *E-stratified* if there is a function ℓ, called an *E-level mapping of* Π, that maps each predicate in $\mathbf{p}$ to a positive integer such that:

1. if r is a rule in Π, p is a predicate having positive occurrence in the body of r, and q is a predicate occurring in the head, then $\ell(p) \leq \ell(q)$;
2. in the above case, if there is an individual variable occurring in the parameters of q and bounded by an existential quantifier, then $\ell(p) < \ell(q)$.

Definition 8
An E-disjunctive program Π is *safe* if it is both $\forall$-safe and E-stratified.

For example, the program that encodes the strategic companies problem in Example 3 is safe.

Given any E-disjunctive program Π, let Π^* denote the program obtained from Π by, for each predicate $p \in \mathbf{p}$, substituting p^* for all positive occurrences of p in the head or the body of any rule in Π, where p^* is a new predicate variable for predicate constant p. For simplicity, without confusion we also use Π to denote its first-order representation (i.e., occurrences of not are replaced by $\neg$, and similarly for conjunction and disjunction). Let $\mathrm{SM}^*_{\mathbf{p}}[\Pi]$ stand for $\Pi \wedge \forall \mathbf{p}^*(\mathbf{p}^* < \mathbf{p} \rightarrow \neg\Pi^*)$. By the definitions of SM and SM^*, it is easy to show the following proposition.

Proposition 3
Let Π be an E-disjunctive program with a tuple of predicates $\mathbf{p}$. Then $\mathrm{SM}^*_{\mathbf{p}}[\Pi]$ is equivalent to $\mathrm{SM}_{\mathbf{p}}[\Pi]$.

We now present two key lemmas whose proofs can be found in Appendix.

Proposition 4
Let Π be a safe E-disjunctive program with a tuple of predicates $\mathbf{p}$, ℓ an E-level mapping of Π, and $\mathcal{A}$ a stable model of Π. Then $|\epsilon(\mathcal{A})| \in O((k \cdot l)^{m^n})$, where k and l are the numbers of individual constants and rules appearing in Π respectively, m is the maximum number of variables in any rule in Π, $n = \max\{\ell(p) \mid p \in \mathbf{p}\}$, and $\epsilon(\mathcal{A})$ is the set of elements from $Dom(\mathcal{A})$ having occurrences in some ground atoms in $\mathcal{A}$.

Proposition 5
Let Π be a safe E-disjunctive program and ℓ an E-level mapping of Π. Then Π has a stable model iff it has a stable model $\mathcal{A}$ with $|Dom(\mathcal{A})| \in O((k \cdot l)^{m^n})$, where k, l, m and n are the same as those in Proposition 4.

By Proposition 5, we immediately have the following theorem.

Theorem 3
It is decidable to check whether or not a safe E-disjunctive program has a stable model.

7 Related Work and Discussion

For Datalog$^\exists$ programs, a number of fragments are known to be decidable, which include the *guarded* fragment, which requires $\forall$-variables in a rule to appear in a single positive body literal of the rule, and its variants (Calì et al. 2009; Gottlob et al. 2012; Alviano et al. 2012). Also see (Fagin et al. 2005) for the fragment of *weakly acyclic* Datalog$^\exists$, (Calì et al. 2012) for *sticky* sets, and (Leone et al. 2012) for the *shy* fragment.

In general, it is an interesting question whether conditions similar to those for decidable Datalog$^\exists$ fragments can be extended to accommodate disjunction and/or negation under the stable model semantics. When disjunction and negation are not considered, safe E-programs are weakly acyclic. Actually, we can define the notion of *weak E-stratification* and show a similar decidability result. This is to say that the condition of being weakly acyclic can be extended to E-disjunctive programs to guarantee decidability, whenever such an E-disjunctive program is $\forall$-safe. On the other hand, as commented earlier in Section 6, the proof of Proposition 2 can be modified slightly to show that such an extension is not possible for sticky sets of (Calì et al. 2012), nor for the shy fragment of (Leone et al. 2012), even for stratified negation. The question whether guarded E-disjunctive programs are decidable remains open.

E-disjunctive programs can be seen as a generalization of disjunctive Datalog of (Alviano et al. 2012) under the safety condition defined in this paper. It can be shown that, given a safe but negation-free E-disjunctive program Π, a set of facts D and a (boolean) conjunctive query q, q is true in every model of $\Pi \cup D$ iff q is true in every stable model of $\Pi \cup D$.

Some defeasible DLs have been proposed recently. A main difference between ASP and various forms of defeasible DLs is that the former aims at a flexible style of declarative knowledge representation, while the latter are designed with a specific focus and by using a specific technique, e.g., based on a form of circumscription (Bonatti et al. 2011), or based on a notion of rational closure for defeasible inheritance networks (Casini and Straccia 2011). An interesting future topic is how E-disjunctive programs may capture some of these defeasible DLs.

Our results here can be generalized to programs with equality (which may appear in rule bodies). In particular, we can similarly define E-disjunctive programs with equality and show a similar decidable fragment based on $\forall$-safety and E-stratification. The idea is that, given a program Π, equality under stable models can be eliminated by introducing a fresh predicate enforcing identity, namely $\forall X \; e(X, X)$ (cf. Lemma 8 of (Zhang and Ying 2010)). Then we substitute $e(s, t)$ for each equality atom $s = t$ appearing in Π.

In this paper we have considered arbitrary structures. In order to access all elements in the domain of an interpretation by name, the *standard names assumption* (SNA) (Motik and Rosati 2010) has been introduced. In such interpretations, called *SNA interpretations*, the sets of ground atoms that are assigned to true are in 1-1 correspondence with subsets of the Herbrand base. For the results in Sections 4-5, as they are true for arbitrary structures they are true for SNA interpretations. It is not difficult to check that all the decidability results in Section 6 are still valid under SNA. For example, the proof of Proposition 2 can also be used to show that the domino system D has a tiling iff Π_D has an SNA stable model, and the bound in Proposition 4 holds for all stable models including SNA stable models.

For finite Herbrand structures an efficient system for computing stable models for E-disjunctive programs already exists, the **claspD** system, where $\exists$-variables can be encoded by the feature of condition on variable instantiations. Also see (Zhang et al. 2011) for a translation for arbitrary but finite structures.

References

ALVIANO, M., FABER, W., LEONE, N., AND MANNA, M. 2012. Disjunctive datalog with existential quantifiers: Semantics, decidability, and complexity issues. *Theory and Practice of Logic Programming 12,* 4-5, 701–718.

BARTHOLOMEW, M. AND LEE, J. 2010. A decidable class of groundable formulas in the general theory of stable models. In *Proc. KR-2010*.

BERGER, R. 1966. The undecidability of the domino problem. *Memoirs of the American Mathematical Society 66,* 1, 1–72.

BONATTI, P. A., FAELLA, M., AND SAURO, L. 2011. Defeasible inclusions in low-complexity DLs. *J. Artif. Intell. Res. 42*, 719–764.

CABALAR, P., PEARCE, D., AND VALVERDE, A. 2009. A revised concept of safety for general answer set programs. In *Proc. LPNMR*. 58–70.

CADOLI, M., EITER, T., AND GOTTLOB, G. 1997. Default logic as a query language. *IEEE Trans. Knowl. Data Eng. 9,* 3, 448–463.

CALÌ, A., GOTTLOB, G., AND KIFER, M. 2008. Taming the infinite chase: Query answering under expressive relational constraints. In *KR*. 70–80.

CALÌ, A., GOTTLOB, G., AND LUKASIEWICZ, T. 2009. A general datalog-based framework for tractable query answering over ontologies. In *Proc. PODS*. 77–86.

CALÌ, A., GOTTLOB, G., AND PIERIS, A. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell. 193*, 87–128.

CASINI, G. AND STRACCIA, U. 2011. Defeasible inheritance-based description logics. In *Proc. IJCAI-11*. 813–818.

EITER, T., GOTTLOB, G., AND MANNILA, H. 1997. Disjunctive datalog. *ACM Trans. Database Syst. 22,* 3, 364–418.

FAGES, F. 1994. Consistency of clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science 1*, 51–60.

FAGIN, R., KOLAITIS, P. G., MILLER, R. J., AND POPA, L. 2005. Data exchange: semantics and query answering. *Theor. Comput. Sci. 336,* 1, 89–124.

FERRARIS, P., LEE, J., AND LIFSCHITZ, V. 2011. Stable models and circumscription. *Artificial Intelligence 175,* 1, 236–263.

FERRARIS, P., LEE, J., LIFSCHITZ, V., AND PALLA, R. 2009. Symmetric splitting in the general theory of stable models. In *IJCAI-09*. 797–803.

GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *Proc. ICLP'88*. 1070–1080.

GOTTLOB, G., HERNICH, A., KUPKE, C., AND LUKASIEWICZ, T. 2012. Equality-friendly well-founded semantics and applications to description logics. In *Proc. AAAI-12*.

GUREVICH, Y. AND KORYAKOV, I. 1972. Remarks on Berger's paper on the domino problem. *Siberian Mathematical Journal 13*, 319–321.

LEE, J., LIFSCHITZ, V., AND PALLA, R. 2008. Safe formulas in the general theory of stable models (preliminary report). In *Proc. ICLP*. 672–676.

LEE, J. AND MENG, Y. 2011. First-order stable model semantics and first-order loop formulas. *J. Artif. Intell. Res. (JAIR) 42*, 125–180.

LEE, J. AND PALLA, R. 2012. Reformulating the situation calculus and the event calculus in the general theory of stable models and in answer set programming. *J. Artif. Intell. Res. (JAIR) 43*, 571–620.

LEONE, N., MANNA, M., TERRACINA, G., AND VELTRI, P. 2012. Efficiently computable datalog; programs. In *KR*.

LEONE, N., PFEIFER, G., FABER, W., EITER, T., GOTTLOB, G., PERRI, S., AND SCARCELLO, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log. 7,* 3, 499–562.

LIN, F. AND ZHOU, Y. 2011. From answer set logic programming to circumscription via logic of GK. *Artificial Intelligence 175,* 1, 264–277.

Motik, B. and Rosati, R. 2010. Reconciling description logics and rules. *Journal of the ACM 57,* 5, 1–62.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3-4, 241–273.

Nieuwenhuis, R., Oliveras, A., and Tinelli, C. 2006. Solving SAT and SAT modulo theories: From an abstract davis–putnam–logemann–loveland procedure to DPLL(T). *J. ACM 53,* 6, 937–977.

You, J.-H., Shen, Y.-D., and Wang, K. 2012. Well-supported semantics for logic programs with generalized rules. In *Correct Reasoning: Essays on Logic-Based AI in Honor of Vladimir Lifschitz,*. LNCS 7265, 576–591.

Zhang, H. and Ying, M. 2010. Decidable fragments of first-order language under stable model semantics and circumscription. In *Proc. AAAI-10*.

Zhang, H., Zhang, Y., Ying, M., and Zhou, Y. 2011. Translating first-order theories into logic programs. In *Proc. IJCAI-11*. 1126–1131.

Zhou, Y. and Zhang, Y. 2011. Progression semantics for disjunctive logic programs. In *Proc. AAAI-11*. 286–291.