Expressiveness of Logic Programs under the General Stable Model Semantics

HENG ZHANG and YAN ZHANG

The stable model semantics had been recently generalized to non-Herbrand structures by several works, which provides a unified framework and solid logical foundations for answer set programming. This paper focuses on the expressiveness of normal and disjunctive logic programs under the general stable model semantics. A translation from disjunctive logic programs to normal logic programs is proposed for infinite structures. Over finite structures, some disjunctive logic programs are proved to be intranslatable to normal logic programs if the arities of auxiliary predicates and functions are bounded in a certain way. The equivalence of the expressiveness of normal logic programs and disjunctive logic programs over arbitrary structures is also shown to coincide with that over finite structures, and coincide with whether NP is closed under complement. Moreover, to obtain a more explicit picture of the expressiveness, some intertranslatability results between logic program classes and fragments of second-order logic are established.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving —*logic programming and nonmonotonic reasoning*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*predicate logic, representation languages*

General Terms: Languages, Theory

Additional Key Words and Phrases: Answer set programming, expressiveness, complexity, nonmonotonic reasoning, second-order logic

1. INTRODUCTION

Logic programming with default negation is an elegant and efficient formalism for Knowledge Representation and Reasoning, which incorporates the abilities of classical logic, inductive definition and commonsense reasoning. Nowadays, the most prominent semantics for this formalism is the stable model semantics proposed by Gelfond and Lifschitz [Gelfond and Lifschitz 1988]. Logic programming based on this semantics, which is known as Answer Set Programming (ASP), has then emerged as a flourishing paradigm for declarative programming in the last two decades.

The original stable model semantics focuses only on Herbrand structures in which the unique name assumption is made. For a certain class of applications, this assumption will simplify the representation. However, there are many applications where the knowl-

Authors' address: Heng Zhang, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China, and School of Software, Tianjin University, Tianjin 300072; Yan Zhang, School of Computing, Engineering and Mathematics, Western Sydney University, Locked Bag 1797, Penrith, NSW 2751, Australia, and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Science, Beijing 100190, China.

edge can be more naturally represented over non-Herbrand structures including arithmetical structures. To overcome this limit, the general stable model semantics, which generalizes the original semantics to arbitrary structures, was then proposed via second-order logic [Ferraris et al. 2011], via circumscription [Lin and Zhou 2011], and via Gödel's 3-valued logic [Pearce and Valverde 2005], respectively. This new stable model semantics provides us a unified framework for answer set programming, armed with powerful tools from classical logic.

The main goal of this work is to identify the expressiveness of logic programs, which is one of the central topics in Knowledge Representation and Reasoning. We will focus on two important classes of logic programs – normal logic programs (NLP) and disjunctive logic programs (DLP). Over Herbrand structures, the expressiveness of logic programs under query equivalence has been thoroughly studied in last three decades. A comprehensive survey for these works can be found in [Dantsin et al. 2001]. Our task described in this paper, however, is quite different. On the one hand, we will work on the general stable model semantics so that non-Herbrand structures will be considered. On the other hand, instead of considering query equivalence, the expressiveness in our work will be based on model equivalence. This setting is important since ASP solvers are usually used to generate models. Note that model equivalence always implies query equivalence, but the converse is in general false.

We also hope this work contributes to the effective implementation of answer set solvers. Translating logic programs into classical logics is a usual approach to implement answer set solvers. For example, in the propositional case, there have been a number of works that implemented answer set solving by reducing the existence of answer sets to the satisfiability of classical propositional logic, see, e.g., [Lin and Zhao 2004; Lierler and Maratea 2004]. In this work, we are interested in translating normal logic programs to first-order sentences so that the state-of-the-art SMT solvers can be used for answer set solving. On the other hand, our work also considers the optimization of logic programs. It is clear that a language is simpler in the syntax, then it is more likely to have an efficient implementation. Therefore, in this work, we will also investigate whether a rich language can be compiled (translated) to a simple language or not. In particular, since the arity of auxiliary symbol is the most important factor to introduce nondeterminism [Immerman 1999], we will try to find translations in which the maximum arity of auxiliary symbols is as small as possible.

Our contribution in this paper is fourfold. Firstly, we show that, over infinite structures, every disjunctive program can be equivalently translated to a normal one. Secondly, we prove that, if only finite structures are considered, for each integer n greater than 1 there is a disjunctive program with intensional predicates of arities less than n that cannot be equivalently translated to any normal program with auxiliary predicates of arities less than $2n$. Thirdly, we show that disjunctive and normal programs are of the same expressiveness over arbitrary structures if, and only if, they are of the same expressiveness over finite structures, if, and only if, the complexity class NP is closed under complement. Lastly, to understand the exact expressiveness of logic programs, we also prove that the intertranslatability holds between some classes of logic programs and some fragments of second-order logic.

The rest of this paper is organized as follows. Section 2 presents necessary concepts, notions, definitions and background knowledge that we will need through out this paper. Section 3 studies the expressiveness of logic programs over infinite structures where we propose a translation from DLP to NLP. Section 4 then focuses the expressiveness over finite structures. In particular, we show that over finite structures, DLP and NLP have the same expressiveness if, and only if, NP is closed under complement. A more subtle intranslatability property from DLP to NLP is also proved in this section. Based on the results from Sections 3 and 4, Section 5 compares the expressiveness of NLP and DLP over arbitrary structures. Finally, Section 6 concludes the paper with some remarks.

2. PRELIMINARIES

A *vocabulary* consists of a finite set of predicate constants and a finite set of function constants. Logical symbols are as usual, including a countable set of predicate variables and a countable set of function variables. Every constant or variable is equipped with a natural number, called its *arity*. Nullary function constants and variables are called *individual constants* and *variables*, respectively. Nullary predicate constants are called *propositional constants*. Sometimes we do not distinguish between predicate constants and predicate variables, and simply call them *predicates*; and likewise we sometimes refer to function constants and function variables as *functions* if no confusion occurs. Atoms, formulas, sentences and theories of a vocabulary υ (or shortly, υ-atoms, υ-formulas, υ-sentences and υ-theories) are built from constants in υ, variables, equality $=$, connectives $\bot, \top, \wedge, \vee, \rightarrow$, and quantifiers $\exists, \forall$ in a standard way. Every *positive clause* of υ is a finite disjunction of υ-atoms. Given a sentence φ and a theory Σ, let $\upsilon(\varphi)$ and $\upsilon(\Sigma)$ denote the sets of predicate and function constants that occur in φ and Σ, respectively.

Suppose $\mathsf{Q} \in \{\forall, \exists\}$, $\tau = \{X_1, \ldots, X_n\}$, and $\vec{x} = x_1 \cdots x_m$, where X_i ranges over predicate and function variables, and x_j ranges over individual variables. We let $\mathsf{Q}\tau$ and $\mathsf{Q}\vec{x}$ be shorthands of the quantifier blocks $\mathsf{Q}X_1 \cdots \mathsf{Q}X_n$ and $\mathsf{Q}x_1 \cdots \mathsf{Q}x_m$, respectively. A quantifier is called *second-order* if it involves either a predicate variable or a function variable of a positive arity. Let $\Sigma^{1\mathrm{F}}_{n,k}$ be the class of sentences of the form $\mathsf{Q}_1\tau_1 \cdots \mathsf{Q}_n\tau_n\varphi$, where Q_i is $\exists$ if i is odd, and $\forall$ otherwise; τ_i is a finite set of variables of arities $\leq k$; and no second-order quantifier appears in φ. Let $\Sigma^1_{n,k}$ denote the class defined the same as $\Sigma^{1\mathrm{F}}_{n,k}$ except no function variable allowed in any τ_i. Let $\Sigma^{1\mathrm{F}}_n$ (respectively, Σ^1_n) be the union of $\Sigma^{1\mathrm{F}}_{n,k}$ (respectively, $\Sigma^1_{n,k}$) for all $k \geq 0$. Given a class Λ defined as above, let $\Lambda[\forall^*\exists^*]$ (respectively, $\Lambda[\forall^*]$) be the class of sentences in Λ with first-order part of the form $\forall\vec{x}\exists\vec{y}\vartheta$ (respectively, $\forall\vec{x}\vartheta$), where $\vec{x}, \vec{y}$ are tuples of individual variables, and ϑ is quantifier-free.

EXAMPLE 1. *Let φ denote the second-order formula*

$$\exists X \forall xy (X(a) \wedge (X(x) \wedge E(x,y) \rightarrow X(y)) \wedge \neg X(b)) \tag{1}$$

where E is a binary predicate constant, X is a unary predicate variable, and a, b are two individual constants. As defined, it is clear that φ is a Σ^1_1-sentence and, moreover, it is also in $\Sigma^{1\mathrm{F}}_1$. Towards a more explicit classification, φ is also a $\Sigma^1_{1,1}[\forall^*\exists^*]$-sentence because the first-order part is universal, and only one unary predicate variable is quantified.

EXAMPLE 2. *Let ψ denote the second-order formula*

$$\exists f (\forall xy (f(x) = f(y) \rightarrow x = y) \wedge \exists x \forall y \neg (x = f(y))) \tag{2}$$

where f is a unary function variable. Clearly, ψ is a $\Sigma^{1\mathrm{F}}_1$-sentence, but it is not in Σ^1_1.

Every *structure* $\mathbf{A}$ of υ (or shortly, υ-*structure* $\mathbf{A}$) is accompanied by a nonempty set A, called the *domain* of $\mathbf{A}$, and interprets each n-ary predicate constant P in υ as an n-ary relation, denoted $P^{\mathbf{A}}$, on A, and interprets each n-ary function constant f in υ as an

n-ary function, denoted $f^{\mathbf{A}}$, on A. A structure is *finite* if its domain is finite, and *infinite* otherwise. Let FIN denote the class of finite structures, and INF denote the class of infinite structures. A *restriction* of a structure $\mathbf{A}$ to a vocabulary σ is the structure obtained from $\mathbf{A}$ by discarding all interpretations for constants which are not in σ. Given a vocabulary $\upsilon \supset \sigma$ and a σ-structure $\mathbf{B}$, every υ*-expansion* of $\mathbf{B}$ is a structure $\mathbf{A}$ of υ such that $\mathbf{B}$ is a restriction of $\mathbf{A}$ to σ. Given a structure $\mathbf{A}$ and a set τ of predicates, let $\textsc{Ins}(\mathbf{A}, \tau)$ denote the set of *ground atoms* $P(\vec{a})$ for all tuples $\vec{a} \in P^{\mathbf{A}}$ and all predicate constants P in τ.

Every *assignment* in a structure $\mathbf{A}$ is a function that maps each individual variable to an element of A and that maps each predicate (respectively, function) variable to a relation (respectively, function) on A of the same arity. For convenience, we assume that the definition of assignments extends to terms natually. Given a formula φ and an assignment α in $\mathbf{A}$, we write $\mathbf{A} \models \varphi[\alpha]$ if α *satisfies* φ in $\mathbf{A}$ in the standard way. In particular, if φ is a sentence, we simply write $\mathbf{A} \models \varphi$ and say that $\mathbf{A}$ is a *model* of φ, or in other words, φ *is true* in $\mathbf{A}$. We use $\mathsf{Mod}(\varphi)$ to denote the set of all models of φ. Given formulas φ, ψ and a class $\mathcal{C}$ of structures, we say φ is *equivalent* to ψ over $\mathcal{C}$, or write $\varphi \equiv_{\mathcal{C}} \psi$ for short, if for every $\mathbf{A}$ in $\mathcal{C}$ and every assignment α in $\mathbf{A}$, α satisfies φ in $\mathbf{A}$ if, and only if, α satisfies ψ in $\mathbf{A}$. In particular, if $\mathcal{C}$ is the class of arbitrary structures, the words "over $\mathcal{C}$" and the subscript $\mathcal{C}$ can be dropped. Given a quantifier-free formula φ and an assignment α in $\mathbf{A}$, let $\varphi[\alpha]$ denote the *ground formula* obtained from φ by (i) substituting $P(\alpha(t_1), \ldots, \alpha(t_k))$ for $P(t_1, \ldots, t_k)$ if $P(t_1, \ldots, t_k)$ is an atomic formula and P is a predicate constant of arity k, followed by (ii) substituting $\top$ for $t_1 = t_2$ if $\alpha(t_1) = \alpha(t_2)$, and by (iii) substituting $\bot$ for $t_1 = t_2$ otherwise.

A class of structures is also called a *property*. Let $\mathcal{C}$ and $\mathcal{D}$ be any two properties. We say that $\mathcal{D}$ is *defined* by a sentence φ over $\mathcal{C}$, or equivalently, φ *defines* $\mathcal{D}$ over $\mathcal{C}$, if each structure $\mathbf{A}$ from $\mathcal{C}$ is in $\mathcal{D}$ if, and only if, $\mathbf{A}$ is a model of φ; that is, $\mathcal{D} = \mathsf{Mod}(\varphi) \cap \mathcal{C}$. $\mathcal{D}$ is *definable* in a class Σ of sentences over $\mathcal{C}$ if there is a sentence in Σ that defines $\mathcal{D}$ over $\mathcal{C}$. Given two classes Σ, Λ of sentences, we write $\Sigma \leq_{\mathcal{C}} \Lambda$ if each property definable in Σ over $\mathcal{C}$ is also definable in Λ over $\mathcal{C}$; we write $\Sigma \simeq_{\mathcal{C}} \Lambda$ if both $\Sigma \leq_{\mathcal{C}} \Lambda$ and $\Lambda \leq_{\mathcal{C}} \Sigma$ hold. Intuitively, $\Sigma \leq_{\mathcal{C}} \Lambda$ asserts that, over the structure class $\mathcal{C}$, the fragment Λ is at least as expressive as Σ; and $\Sigma \simeq_{\mathcal{C}} \Lambda$ asserts that, over the structure class $\mathcal{C}$, Σ and Λ are of the same expressiveness. Again, in above definitions, if $\mathcal{C}$ is the class of arbitrary structures, the words "over $\mathcal{C}$" and the subscript $\mathcal{C}$ might be omitted.

EXAMPLE 3 (EXAMPLE 1 CONTINUED). *Let φ be the same as in Example 1, that is,*

$$\varphi = \exists X \forall xy (X(a) \wedge (X(x) \wedge E(x,y) \rightarrow X(y)) \wedge \neg X(b)). \tag{3}$$

Let υ denote the vocabulary $\{E, a, b\}$. It is clear that every finite υ-structure $\mathbf{A}$ can be regarded as a directed graph $G_{\mathbf{A}}$ with two distinguished nodes $a^{\mathbf{A}}$ and $b^{\mathbf{A}}$ such that the node set of $G_{\mathbf{A}}$ is the domain A and the edge set of $G_{\mathbf{A}}$ is the binary relation $E^{\mathbf{A}}$.

Let $\mathsf{Unreach}$ *denote the class of finite υ-structures $\mathbf{A}$ such that $b^{\mathbf{A}}$ is unreachable from $a^{\mathbf{A}}$ in the directed graph $G_{\mathbf{A}}$. It is easy to see that every finite υ-structure is in* $\mathsf{Unreach}$ *if, and only if, it is a model of φ; that is, φ defines* $\mathsf{Unreach}$ *over the class of finite structures.*

EXAMPLE 4 (EXAMPLE 2 CONTINUED). *Let ψ be the same as in Example 2, that is,*

$$\psi = \exists f (\forall xy (f(x) = f(y) \rightarrow x = y) \wedge \exists x \forall y \neg (x = f(y))). \tag{4}$$

Let σ denote the empty vocabulary $\emptyset$. One can check that every σ-structure $\mathbf{A}$ is a model of ψ if, and only if, $\mathbf{A}$ is infinite. Thus, ψ defines INF *over the class of arbitrary structures.*

2.1 Logic Programs and Stable Models

Every *disjunctive logic program* (or simply called *disjunctive program*) consists of a set of *rules*, each of which is a first-order formula of the following form[1]:

$$\theta_1 \wedge \cdots \wedge \theta_m \to \theta_{m+1} \vee \cdots \vee \theta_n \tag{5}$$

where $0 \leq m \leq n$ and $n > 0$; θ_i is an atom without involving any equality if $m < i \leq n$; θ_i is a *literal* (i.e., an atom or the negation of an atom) if $1 \leq i \leq m$. Given a rule, the disjunctive part is called its *head*, and the conjunctive part is called its *body*. Given a disjunctive program Π, a predicate is called *intensional* (w.r.t. Π) if it appears in the head of some rule in Π, and *extensional* otherwise. A formula is called *intensional* (w.r.t. Π) if it does not involve any extensional predicate of Π. Let $\upsilon(\Pi)$ denote the set of predicate and function constants that appear in Π. Note that, in this paper, all negations in intensional literals w.r.t. Π will be assumed as *default negations*.

Let Π be a disjunctive program. Then Π is called *normal* if the head of each rule contains at most one atom, Π is *plain* if the negation of any intensional atom does not appear in the body of any rule, Π is *propositional* if it does not involve any predicate of a positive arity, and Π is *finite* if it contains only a finite set of rules. In general, we will assume that all disjunctive programs in this paper are finite when they involve predicates with positive arities, while propositional logic programs may contain infinite sets of rules.

EXAMPLE 5. *Let Π be a disjunctive program consisting of the following rules*

$$P(a), \tag{6}$$
$$P(x) \wedge E(x,y) \to P(y), \tag{7}$$
$$\neg P(b) \to P(b), \tag{8}$$

where P and E are unary and binary predicates, respectively, and both a, b are individual constants. Clearly, Π is normal, and P is the only intensional predicate of Π. Note that the body of the rule (6) is empty. In this case, we usually omit the connective $\to$.

Given any disjunctive program Π, let $\mathrm{SM}(\Pi)$ denote the second-order sentence

$$\widehat{\Pi} \wedge \forall \tau^*(\tau^* < \tau \to \neg \widehat{\Pi}^*) \tag{9}$$

where:

(1) τ is the set of all intensional predicates w.r.t. Π;

(2) τ^* is the set of predicate variables P^* for all predicates $P \in \tau$, where for each predicate constant $Q \in \tau$ we introduce a fresh predicate variable Q^*;

(3) $\tau^* < \tau$ denotes the formula

$$\bigwedge_{P \in \tau} \forall \vec{x}(P^*(\vec{x}) \to P(\vec{x})) \wedge \neg \bigwedge_{P \in \tau} \forall \vec{x}(P(\vec{x}) \to P^*(\vec{x})); \tag{10}$$

(4) $\widehat{\Pi}$ is the conjunction of sentences $\gamma_\forall$ for all rules $\gamma \in \Pi$ where $\psi_\forall$ denotes the first-order universal closure of ψ whenever ψ is a rule;

[1] Note that, in this paper, we do not require the rules to be safe. The main reason is as follows: The general stable model semantics does not rely on the grounding technique, and an intended domain will be always specified; thus, the termination of the computation on a logic program does not depend on the safety of this program.

(5) $\widehat{\Pi}^*$ denotes the conjunction of $\gamma^*_\forall$ for all $\gamma \in \Pi$ where each $\gamma^*_\forall$ is the formula obtained from $\gamma_\forall$ by substituting P^* for all positive occurrences of P in the head or in the body.

A structure $\mathbf{A}$ is called a *stable model* of Π if it is a model of $\mathrm{SM}(\Pi)$. For more details about the transformational semantics, please refer to [Ferraris et al. 2011]. Note that, since we only consider disjunctive programs, the transformation presented here is slightly simpler than the transformation in [Ferraris et al. 2011] where first-order sentences are focused. For the class of disjunctive programs, the equivalence of both transformations can be easily proved.

Given two properties $\mathcal{C}$ and $\mathcal{D}$, we say $\mathcal{D}$ is *defined* by a disjunctive program Π over $\mathcal{C}$ via the set τ of *auxiliary constants* if the formula $\exists\tau\mathrm{SM}(\Pi)$ defines $\mathcal{D}$ over $\mathcal{C}$, where τ is a set of predicates and functions occurring in Π.[2] Given $n \geq 0$, let DLP_n (respectively, $\mathrm{DLP}^{\mathrm{F}}_n$) be the class of sentences $\exists\tau\mathrm{SM}(\Pi)$ for all disjunctive programs Π and all finite sets τ of predicate (respectively, predicate and function) constants of arities $\leq n$. Let DLP (respectively, $\mathrm{DLP}^{\mathrm{F}}$) be the union of DLP_n (respectively, $\mathrm{DLP}^{\mathrm{F}}_n$) for all integers $n \geq 0$. Furthermore, in above definitions, if Π is restricted to be normal, we then obtain the notations NLP_n, $\mathrm{NLP}^{\mathrm{F}}_n$, NLP and $\mathrm{NLP}^{\mathrm{F}}$, respectively.

EXAMPLE 6 (EXAMPLE 5 CONTINUED). *Let Π be the logic program that is presented in Example 5. Then $\widehat{\Pi}$ denotes the first-order formula*

$$P(a) \wedge \forall xy(P(x) \wedge E(x,y) \rightarrow P(y)) \wedge (\neg P(b) \rightarrow P(b)), \tag{11}$$

and $\widehat{\Pi}^*$ denotes the formula

$$P^*(a) \wedge \forall xy(P^*(x) \wedge E(x,y) \rightarrow P^*(y)) \wedge (\neg P(b) \rightarrow P^*(b)). \tag{12}$$

By definition, we know that $\mathrm{SM}(\Pi)$ is the second-order sentence

$$\widehat{\Pi} \wedge \forall P^*(\forall x(P^*(x) \rightarrow P(x)) \wedge \neg\forall x(P(x) \rightarrow P^*(x)) \rightarrow \neg\widehat{\Pi}^*). \tag{13}$$

Let τ be the set $\{P\}$, and let υ denote the vocabulary $\{E, a, b\}$. Then it is easy to see that every finite υ-structure $\mathbf{A}$ is a model of $\exists P\,\mathrm{SM}(\Pi)$ if, and only if, $\mathbf{A} \in \mathsf{Unreach}$ does not hold (which is defined in Example 3). Let Reach denote the complement of $\mathsf{Unreach}$. Therefore, the class Reach is defined by Π via τ (as the set of auxiliary constants).

Given a rule γ, let γ^-_{B} be the set of conjuncts in the body of γ in which no intensional predicate positively occurs, and let γ^+ be the rule obtained from γ by removing all literals in γ^-_{B}. Given a disjunctive program Π and a structure $\mathbf{A}$, let $\Pi^{\mathbf{A}}$ be the set of rules $\gamma^+[\alpha]$ for all assignments α in $\mathbf{A}$ and all rules γ in Π such that α satisfies γ^-_{B} in $\mathbf{A}$. Now, $\Pi^{\mathbf{A}}$ can be regarded as a propositional program where each ground atom as a proposition. This procedure is called the *first-order Gelfond-Lifschitz reduction* due to the following result.

PROPOSITION 1 (PROPOSITION 4 IN [ZHANG AND ZHANG 2013B]). *Let Π be a disjunctive program and τ be the set of intensional predicates. Then an $\upsilon(\Pi)$-structure $\mathbf{A}$ is a stable model of Π iff $\textsc{Ins}(\mathbf{A}, \tau)$ is a minimal (w.r.t. the set inclusion) model of $\Pi^{\mathbf{A}}$.*

To make the logic program more readable, now let us present a splitting lemma. Note that it directly follows from the first splitting lemma presented in [Ferraris et al. 2009].

[2]To simplify the presentation, we slightly abuse the notion without confusion: Each predicate (function) constant that appears in τ in the formula $\exists\tau\mathrm{SM}(\Pi)$ is now regarded as a predicate (function) variable of the same airty.

PROPOSITION 2 (SPLITTING LEMMA IN [FERRARIS ET AL. 2009]). *Let* Π *be a disjunctive program, and let* $\{\Pi_1, \Pi_2\}$ *be a partition of* Π *such that all intensional predicates of* Π_1 *is extensional w.r.t.* Π_2 *and that no extensional predicate of* Π_1 *has any occurrence in* Π_2. *Then we have that* $\mathrm{SM}(\Pi)$ *is equivalent to* $\mathrm{SM}(\Pi_1) \wedge \mathrm{SM}(\Pi_2)$.

2.2 Progression Semantics

In this subsection, we review a progression semantics proposed by Zhang and Zhang [2013b], which generalizes Lobo et al.'s fixed point semantics [1992] by allowing default negations and arbitrary structures.

Now, let us present the semantics. For convenience, two positive clauses that contain the same set of atoms will be regarded as the same. Let Π be a propositional, possibly infinite and plain disjunctive program. Let $\textsc{pc}(\upsilon(\Pi))$ denote the set of all positive clauses of $\upsilon(\Pi)$ and take $\Lambda \subseteq \textsc{pc}(\upsilon(\Pi))$. We define $\Gamma_\Pi(\Lambda)$ as the following positive clause set:

$$\left\{ H \vee C_1 \vee \cdots \vee C_k \;\middle|\; \begin{array}{c} k \geq 0 \;\&\; H, C_1, \ldots, C_k \in \textsc{pc}(\upsilon(\Pi)) \\ \&\; \exists p_1, \ldots, p_k \in \upsilon(\Pi) \text{ s.t.} \\ \begin{bmatrix} p_1 \wedge \cdots \wedge p_k \rightarrow H \in \Pi \;\& \\ C_1 \vee p_1, \ldots, C_k \vee p_k \in \Lambda \end{bmatrix} \end{array} \right\}. \tag{14}$$

It is easy to check that Γ_Π is a monotone operator on $\textsc{pc}(\upsilon(\Pi))$.

With this notation, a progression operator for first-order programs can be defined via the first-order Gelfond-Lifschitz reduction. Given any disjunctive program Π and any $\upsilon(\Pi)$-structure $\mathbf{A}$, we define $\Gamma_\Pi^{\mathbf{A}}$ as the operator $\Gamma_{\Pi^{\mathbf{A}}}$; that is, let $\Gamma_\Pi^{\mathbf{A}}(\Lambda)$ denote $\Gamma_{\Pi^{\mathbf{A}}}(\Lambda)$ for all positive clause set $\Lambda \subseteq \textsc{pc}(\upsilon(\Pi^{\mathbf{A}}))$. Furthermore, we define

$$\Gamma_\Pi^{\mathbf{A}} \uparrow_n = \begin{cases} \emptyset & \text{if } n = 0; \\ \Gamma_\Pi^{\mathbf{A}}(\Gamma_\Pi^{\mathbf{A}} \uparrow_{n-1}) & \text{if } n > 0. \end{cases} \tag{15}$$

Finally, let $\Gamma_\Pi^{\mathbf{A}} \uparrow_\omega$ denote the union of $\Gamma_\Pi^{\mathbf{A}} \uparrow_n$ for all integers $n \geq 0$. To illustrate these definitions, a simple example is given as follows.

EXAMPLE 7. *Let* Π *be the disjunctive program consisting of the following rules:*

$$S(x) \vee T(x), \tag{16}$$

$$T(x) \wedge E(x, y) \rightarrow T(y). \tag{17}$$

Let $\upsilon = \{E\}$*, and let* $\mathbf{A}$ *be a structure of* υ *defined as follows:*

(*1*) *the domain* A *is the set* $\{a_i \mid i \in \mathbb{N}\}$ *where* $\mathbb{N}$ *denotes the set of natural numbers;*

(*2*) *the predicate* E *is interpreted as the successor relation, i.e.,* $\{(a_i, a_{i+1}) \mid i \in \mathbb{N}\}$.

Then, $\Pi^{\mathbf{A}}$ *is a propositional disjunctive program consisting of the following rules:*

$$S(a_i) \vee T(a_i) \qquad (i \in \mathbb{N}), \tag{18}$$

$$T(a_i) \rightarrow T(a_{i+1}) \qquad (i \in \mathbb{N}). \tag{19}$$

Therefore, the progression of Π *on* $\mathbf{A}$ *is then defined as follows:*

$$\begin{aligned} \Gamma_\Pi^{\mathbf{A}} \uparrow_0 &= \{\}, \\ \Gamma_\Pi^{\mathbf{A}} \uparrow_1 &= \{S(a_i) \vee T(a_i) \mid i \in \mathbb{N}\}, \end{aligned}$$

$$
\begin{aligned}
& \vdots \\
\Gamma_{\Pi}^{\mathbf{A}} \uparrow_n \;&=\; \{S(a_i) \vee T(a_j) \mid i, j \in \mathbb{N} \,\&\, i \leq j < i+n\}, \\
& \vdots \\
\Gamma_{\Pi}^{\mathbf{A}} \uparrow_\omega \;&=\; \{S(a_i) \vee T(a_j) \mid i, j \in \mathbb{N} \,\&\, i \leq j\}.
\end{aligned}
$$

The following proposition shows that the general stable model semantics can be equivalently redefined by the progression operator that we have defined.

THEOREM 1 (THEOREM 1 IN [ZHANG AND ZHANG 2013B]). *Let Π be a disjunctive program, τ be the set of intensional predicates of Π, and $\mathbf{A}$ be a structure of $\upsilon(\Pi)$. Then $\mathbf{A}$ is a stable model of Π iff $\textsc{Ins}(\mathbf{A}, \tau)$ is a minimal model of $\Gamma_{\Pi}^{\mathbf{A}} \uparrow_\omega$.*

REMARK 1. *In the above theorem, if Π is a normal program, we have that $\mathbf{A}$ is a stable model of Π if, and only if, $\textsc{Ins}(\mathbf{A}, \tau) = \Gamma_{\Pi}^{\mathbf{A}} \uparrow_\omega$.*

3. INFINITE STRUCTURES

Now we study the expressiveness of logic programs over infinite structures. We first propose a translation that reduces each disjunctive program to a normal program over infinite structures. The main idea is to encode ground positive clauses by elements in the intended domain. With the encoding, we then simulate the progression of the given disjunctive program by the progression of a normal program.

We first show how to encode a positive clause by a domain element. Let A be an infinite set. Every *encoding function* on A is an injective function from $A \times A$ into A. Let enc be an encoding function on A and c an element in A such that c does not belong to the range of enc. To simplify the statement, let $enc(a_1, \ldots, a_k; c)$ be short for the expression

$$enc(\cdots enc(enc(c, a_1), a_2), \cdots, a_k) \tag{20}$$

for any $k \geq 0$ and any set of elements $a_1, \ldots, a_k \in A$. In the above expression, the special element c is used as a flag to indicate that the encoded tuple will be started after c, and we call c the *encoding flag* of this encoding. Intuitively, by $b = enc(\vec{a}; c)$ we means that b is the encoding of $\vec{a}$ related to the encoding function enc and the encoding flag c.

Let A^* denote the set of finite tuples of elements in A, and let

$$enc(A; c) = \{b \in A \mid \exists \vec{a} \in A^* \text{ s.t. } b = enc(\vec{a}; c)\}. \tag{21}$$

The *merging function* mrg on A related to enc and c is then defined as the function from $enc(A; c) \times enc(A; c)$ into $enc(A; c)$ such that

$$mrg(enc(\vec{a}_1; c), enc(\vec{a}_2; c)) = enc(\vec{a}_1\vec{a}_2; c) \tag{22}$$

for all tuples $\vec{a}_1$ and $\vec{a}_2$ in A^*. Suppose $b_i = enc(\vec{a}_i; c)$ for $1 \leq i \leq k$. Again, to simplify the statement, we let $mrg(b_1, \ldots, b_k)$ be short for the expression

$$mrg(\cdots mrg(mrg(b_1, b_2), b_3), \cdots, b_k). \tag{23}$$

It is easy to see that the following holds:

$$mrg(b_1, \ldots, b_k) = enc(\vec{a}_1 \cdots \vec{a}_k; c). \tag{24}$$

In other words, if $b = mrg(b_1, \dots, b_k)$, b is then an element that encodes the tuple obtained by joining all the tuples encoded by $b_1, \dots, b_k$ sequentially. Note that the merging function mrg is unique if both the encoding function enc and the encoding flag c have been fixed.

EXAMPLE 8. *Let A denote the set of all positive integers and assume that A is the domain on which we will focus. Let P_1, P_2, P_3 be three predicates of arities $2, 3, 1$, respectively. Now we show how to encode ground positive clauses by elements in A.*

Let $e : A \times A \to A$ be a function such that

$$e(m, n) = 2^m + 3^n \tag{25}$$

for all $m, n \in A$. It is easy to check that e is an encoding function on A, and integers $1, 2, 3, 4$ are not in the range of e. For $i \in \{1, 2, 3\}$, fix i to be the encoding flag for the encodings of atoms built from P_i. Then the ground atom $P_2(1, 3, 5)$ can be encoded by

$$e(1, 3, 5; 2) = e(e(e(2, 1), 3), 5) = 2^{155} + 3^5. \tag{26}$$

Let 4 be the encoding flag for encodings of positive clauses. Then the positive clause

$$P_2(1, 3, 5) \vee P_3(2) \vee P_1(2, 4) \tag{27}$$

can be encoded by $e(e(1, 3, 5; 2), e(2; 3), e(2, 4; 1); 4)$.

In classical logic, two positive clauses are equivalent if, and only if, they contain the same set of atoms. Assume that c is the encoding flag for encodings of positive clauses and enc is the encoding function. To capture the equivalence between two positive clauses, some encoding relations related to enc and c are needed. We define them as follows:

$$in \ = \{(enc(\vec{a}; c), b) \mid \vec{a} \in A^* \wedge b \in \text{ELEM}(\vec{a})\}, \tag{28}$$

$$subc \ = \{(enc(\vec{a}; c), enc(\vec{b}; c)) \mid \vec{a}, \vec{b} \in A^* \wedge \text{ELEM}(\vec{a}) \subseteq \text{ELEM}(\vec{b})\}, \tag{29}$$

$$equ \ = \{(enc(\vec{a}; c), enc(\vec{b}; c)) \mid \vec{a}, \vec{b} \in A^* \wedge \text{ELEM}(\vec{a}) = \text{ELEM}(\vec{b})\}, \tag{30}$$

where $\text{ELEM}(\vec{a})$ and $\text{ELEM}(\vec{b})$ denote the sets of elements in $\vec{a}$ and $\vec{b}$, respectively. Intuitively, $in(a, b)$ asserts that the atom encoded by b appears in the positive clause encoded by a; $subc(a, b)$ asserts that the positive clause encoded by a is a subclause of that encoded by b; and $equ(a, b)$ asserts that the positive clauses encoded by a and b are equivalent.

With the above method for encoding, we can now define the translation. Let Π be a disjunctive program. We first construct a class of normal programs related to Π as follows:

1. Let C_Π denote the set that consists of an individual constant c_P for each predicate constant P that occurs in Π, and an individual constant c_ϵ. Here, c_ϵ will be interpreted as the encoding flag for positive clauses, and c_P will be interpreted as the encoding flag for atoms built from P. Let Π_1 be the logic program that consists of the rule

$$Enc(x, y, c) \ \to \ \bot \tag{31}$$

for each individual constant $c \in C_\Pi$, and the following rules:

$$\neg \underline{Enc}(x, y, z) \ \to \ Enc(x, y, z) \tag{32}$$

$$\neg Enc(x, y, z) \ \to \ \underline{Enc}(x, y, z) \tag{33}$$

$$Enc(x, y, z) \wedge Enc(u, v, z) \wedge \neg x = u \ \to \ \bot \tag{34}$$

$$Enc(x, y, z) \wedge Enc(u, v, z) \wedge \neg y = v \ \to \ \bot \tag{35}$$

$$Enc(x,y,z) \rightarrow Defined(x,y) \tag{36}$$
$$\neg Defined(x,y) \rightarrow Defined(x,y) \tag{37}$$
$$Enc(x,y,z) \wedge Enc(x,y,u) \wedge \neg z = u \rightarrow \bot \tag{38}$$

Informally, rules (36)–(38) define that Enc is the graph[3] of a function; rules (34)–(35) state that Enc is injective. Thus, Enc should be the graph of an encoding function. In addition, the rule (31) assures that each $c \in C_\Pi$ is not in the range of Enc.

2. Let Π_2 be the logic program that consists of the following rules:

$$y = c_\epsilon \rightarrow Mrg(x,y,x) \tag{39}$$
$$Mrg(x,u,v) \wedge Enc(u,w,y) \wedge Enc(v,w,z) \rightarrow Mrg(x,y,z) \tag{40}$$
$$Enc(x,u,y) \rightarrow In(y,u) \tag{41}$$
$$Enc(x,z,y) \wedge In(x,u) \rightarrow In(y,u) \tag{42}$$
$$x = c_\epsilon \rightarrow Subc(x,y) \tag{43}$$
$$Subc(u,y) \wedge Enc(u,v,x) \wedge In(y,v) \rightarrow Subc(x,y) \tag{44}$$
$$Subc(x,y) \wedge Subc(y,x) \rightarrow Equ(x,y) \tag{45}$$

Informally, rules (39)–(40) state that Mrg is the graph of the merging function related to Enc and c_ϵ; rules (41)–(42) implement an inductive version of the definition presented in (28); rules (43)–(44) implement an inductive version of the definition presented in (29); and rules (41)–(45) then assert that Equ is the equivalence relation between positive clauses.

3. Let Π_3 be the logic program that consists of the rule

$$True(x) \wedge Equ(x,y) \rightarrow True(y) \tag{46}$$

and the rule

$$\begin{bmatrix} True(x_1) \wedge \cdots \wedge True(x_k) \wedge \\ Enc(y_1, \ulcorner\theta_1\urcorner, x_1) \wedge \cdots \wedge Enc(y_k, \ulcorner\theta_k\urcorner, x_k) \\ \wedge\, Mrg(\ulcorner\gamma_{\mathrm{H}}\urcorner, y_1, \ldots, y_k, z) \wedge \gamma_{\mathrm{B}}^- \end{bmatrix} \rightarrow True(z) \tag{47}$$

for each rule $\gamma \in \Pi$, where:

(1) $\theta_1, \ldots, \theta_k$ list all the intensional atoms that have strictly positive occurrences in the body of γ for some $k \geq 0$;

(2) γ_{H} is the head of γ, and γ_{B}^- is the conjunction of literals occurring in the body of γ but not in the list $\theta_1, \ldots, \theta_k$;

(3) $Enc(s_1, \ulcorner\theta\urcorner, s_2)$ denotes the formula

$$u_0^i = c_P \wedge Enc(u_0^i, t_1, u_1^i) \wedge \cdots \wedge Enc(u_{m-1}^i, t_m, u_m^i) \wedge Enc(s_1, u_m^i, s_2) \tag{48}$$

if s_1, s_2 are two terms, and θ is an atom of the form $P(t_1, \ldots, t_m)$ for some predicate constant P that occurs in Π, where each u_j^i is a fresh individual variable;

(4) $Mrg(\ulcorner\gamma_{\mathrm{H}}\urcorner, y_1, \ldots, y_k, z)$ denotes the formula

$$\begin{aligned} Enc(v_0, \ulcorner\zeta_1\urcorner, v_1) \wedge \cdots \wedge Enc(v_{n-1}, \ulcorner\zeta_n\urcorner, v_n) \wedge Mrg(w_0, y_1, w_1) \wedge \cdots \\ \wedge\, Mrg(w_{k-1}, y_k, w_k) \wedge v_0 = c_\epsilon \wedge w_0 = v_n \wedge z = w_k \end{aligned} \tag{49}$$

[3]Given a k-ary function f on some set A, the *graph* of f is defined as the relation $\{(\vec{a}, f(\vec{a})) : \vec{a} \in A^k\}$.

if $\gamma_{\text{H}} = \zeta_1 \vee \cdots \vee \zeta_n$ for some atoms $\zeta_1, \ldots, \zeta_n$ and some integer $n \geq 0$.

Intuitively, the rule (46) assures that the progression is closed under the equivalence of positive clauses; the rule (47) simulates the progression operator for the original program. As each positive clause is encoded by an element in the intended domain, the processes of decoding and encoding should be carried out before and after the simulation, respectively.

EXAMPLE 9. *Let γ denote the following rule:*

$$P(v) \wedge \neg Q(v) \rightarrow R(v) \vee S(v) \tag{50}$$

and suppose P, Q, R, S are intensional w.r.t. the underlying program. Then, according to the above translation, we can use the following normal rule (which is defined by (47), but with a slight simplification) to simulate the rule γ:

$$\begin{bmatrix} True(x_1) \wedge Enc(c_P, v, u_1) \wedge Enc(y_1, u_1, x_1) \wedge \\ Enc(c_R, v, u_2) \wedge Enc(c_\epsilon, u_2, w_1) \wedge Enc(c_S, v, u_3) \wedge \\ Enc(w_1, u_3, w_2) \wedge Mrg(w_2, y_1, z) \wedge \neg Q(v) \end{bmatrix} \rightarrow True(z). \tag{51}$$

4. Let Π_4 be the logic program that consists of the rule

$$x = c_\epsilon \ \rightarrow \ False(x) \tag{52}$$

and the rule

$$False(x) \wedge Enc(x, \lceil\theta\rceil, y) \wedge \neg\theta \ \rightarrow \ False(y) \tag{53}$$

for every intensional atom θ of the form $P(\vec{z}_P)$, where $\vec{z}_P$ is a tuple of pairwise distinct individual variables $z_1 \cdots z_{k_P}$ that are different from x and y, and k_P is the arity of P.

This program is intended to define the predicate $False$ as follows: $False(a)$ holds in the intended structure if, and only if, a encodes a positive clause that is false in the structure.

REMARK 2. *Suppose $\mathbf{A}$ is the intended structure. By the definition of Π_3, if $True(a)$ is true in $\mathbf{A}$, then a should be an element in A that encodes a positive clause, say C, in $\Gamma_\Pi^{\mathbf{A}} \uparrow_\omega$. By the definition of $\Gamma_\Pi^{\mathbf{A}}$, it is not difficult to see that $\mathbf{A} \models C$, which means that $False(a)$ will be false in this case. Thus, definitions of $True$ and $False$ are consistent.*

5. Let Π_5 be the logic program consisting of the rule

$$True(c_\epsilon) \ \rightarrow \ \bot \tag{54}$$

and the following rule

$$True(x) \wedge Enc(y, \lceil\theta\rceil, x) \wedge False(y) \ \rightarrow \ \theta \tag{55}$$

for each atom θ of the form same as that in Π_4.

Informally, this program asserts that a ground atom is true in the intended structure if, and only if, there is a positive clause containing this atom such that the clause is true and all the other atoms in this clause are false in the intended structure.

Now, we let $\Pi^\diamond$ denote the union of $\Pi_1, \ldots, \Pi_5$. This then completes the definition of the translation. The soundness of this translation is assured by the following theorem.

THEOREM 2. *Let Π be a disjunctive program. Then over infinite structures,* $\mathrm{SM}(\Pi)$ *is equivalent to* $\exists\pi\mathrm{SM}(\Pi^\diamond)$*, where π is the set of constants occurring in $\Pi^\diamond$ but not in Π.*

To prove this result, some notations and lemmas are needed. Let υ_i and τ be the sets of intensional predicates of Π_i and Π, respectively. Let $\sigma = \upsilon_1 \cup \upsilon_2 \cup \upsilon(\Pi)$. Given a structure $\mathbf{A}$ of $\upsilon(\Pi)$, every *encoding expansion* of $\mathbf{A}$ is a σ-expansion $\mathbf{B}$ of $\mathbf{A}$ such that

(1) $\mathbf{B}$ interprets Enc as the graph of an encoding function enc on A such that no element among $c_\epsilon^{\mathbf{B}}$ and $c_P^{\mathbf{B}}$ (for all $P \in \tau$) belongs to the range of enc, and interprets $\underline{Enc}$ as the complement of the graph of enc, and interprets $Defined$ as $A \times A$;

(2) $\mathbf{B}$ interprets Mrg as the graph of the merging function related to enc and $c_\epsilon^{\mathbf{B}}$, and interprets $In, Subc, Equ$ as the encoding relations $in, subc, equ$ related to enc and $c_\epsilon^{\mathbf{B}}$, respectively.

In the rest of the proof for Theorem 2, unless otherwise mentioned, we assume $\mathbf{A}$ is a structure of $\upsilon(\Pi)$, $\mathbf{B}$ is an encoding expansion of $\mathbf{A}$, and enc is the encoding function defined by the predicate Enc in the structure $\mathbf{B}$. Furthermore, we define

$$[\![P(a_1, \ldots, a_k)]\!] = enc(a_1, \ldots, a_k; c_P^{\mathbf{B}}), \tag{56}$$

$$[\![\theta_1 \vee \cdots \vee \theta_n]\!] = enc([\![\theta_1]\!], \ldots, [\![\theta_n]\!]; c_\epsilon^{\mathbf{B}}). \tag{57}$$

Given a set Σ of ground positive clauses, let $[\![\Sigma]\!]$ be the set of elements $[\![C]\!]$ for all $C \in \Sigma$. Let $\Delta^n(\mathbf{B})$ be the set of elements $a \in A$ such that $True(a) \in \Gamma_{\Pi_3}^{\mathbf{B}} \uparrow_n$.

LEMMA 1. $[\![\Gamma_\Pi^{\mathbf{A}} \uparrow_\omega]\!] = \bigcup_{n \geq 0} \Delta^n(\mathbf{B})$.

PROOF. "$\subseteq$": By definition, it suffices to show that $[\![\Gamma_\Pi^{\mathbf{A}} \uparrow_n]\!] \subseteq \Delta^{2n}(\mathbf{B})$ for all $n \geq 0$. We show this by an induction on n. The case for $n = 0$ is trivial. Let $n > 0$ and assume that $[\![\Gamma_\Pi^{\mathbf{A}} \uparrow_{n-1}]\!] \subseteq \Delta^{2(n-1)}(\mathbf{B})$. Now, our task is to show that $[\![\Gamma_\Pi^{\mathbf{A}} \uparrow_n]\!] \subseteq \Delta^{2n}(\mathbf{B})$. Let $C \in \Gamma_\Pi^{\mathbf{A}} \uparrow_n$. By definition, there is a rule $p_1 \wedge \cdots \wedge p_k \to H$, denoted by γ_p, in $\Pi^{\mathbf{A}}$, and a sequence of clauses $C_1 \vee p_1, \ldots, C_k \vee p_k$ in $\Gamma_\Pi^{\mathbf{A}} \uparrow_{n-1}$ such that $C \equiv C'$, where C' denotes the clause $H \vee C_1 \vee \cdots \vee C_k$. Consequently, there is a rule γ in Π and an assignment α in $\mathbf{A}$ such that α satisfies γ_{B}^- in $\mathbf{A}$ and that $\gamma^+[\alpha] = \gamma_p$. Let a denote $[\![C]\!]$, a' denote $[\![C']\!]$, and a_i denote $[\![C_i \vee p_i]\!]$ for $1 \leq i \leq k$. On the other hand, by the inductive assumption, each a_i should be an element in $\Delta^{2(n-1)}(\mathbf{B})$, or equivalently $True(a_i) \in \Gamma_{\Pi_3}^{\mathbf{B}} \uparrow_{2(n-1)}$; by definition, there exists a rule of the form (47) corresponding to γ in Π_3. Consequently, we have that $True(a') \in \Gamma_{\Pi_3}^{\mathbf{B}} \uparrow_{2n-1}$. As $Equ(a', a)$ is clearly true in $\mathbf{B}$, according to the rule (46) we then have that $True(a) \in \Gamma_{\Pi_3}^{\mathbf{B}} \uparrow_{2n}$, which implies $a \in \Delta^{2n}(\mathbf{B})$.

"$\supseteq$": It suffices to show that $[\![\Gamma_\Pi^{\mathbf{A}} \uparrow_n]\!] \supseteq \Delta^n(\mathbf{B})$ for all $n \geq 0$. Similarly, we show it by an induction on n. The case for $n = 0$ is trivial. Let $n > 0$ and assume that $[\![\Gamma_\Pi^{\mathbf{A}} \uparrow_{n-1}]\!] \supseteq \Delta^{n-1}(\mathbf{B})$. Let $a \in \Delta^n(\mathbf{B})$. We then have $True(a) \in \Gamma_{\Pi_3}^{\mathbf{B}} \uparrow_n$. By definition, $True(a)$ must be generated by either rule (46) or rule (47). If the first case is true, there must exist an element $b \in \Gamma_{\Pi_3}^{\mathbf{B}} \uparrow_{n-1}$ such that $Equ(a, b)$ is true in $\mathbf{B}$. By the inductive assumption, there is a positive clause C such that $[\![C]\!] = b$ and $C \in \Gamma_\Pi^{\mathbf{A}} \uparrow_{n-1}$. By the definition of equ, it is also clear that a encodes a clause C_0 that contains exactly the set of atoms in C. By the definition of the progression operator, it is clear that $C_0 \in \Gamma_\Pi^{\mathbf{A}} \uparrow_{n-1} \subseteq \Gamma_\Pi^{\mathbf{A}} \uparrow_n$.

Now, it remains to consider the case that $True(a)$ is generated by the rule (47). By definition, there should be (i) a sequence of elements $b_1, \ldots, b_k \in B$ such that $True(b_i) \in \Gamma_{\Pi_3}^{\mathbf{B}} \uparrow_{n-1}$ for $1 \leq i \leq k$, (ii) a rule $\gamma \in \Pi$ such that $\theta_1, \ldots, \theta_k$ are the set of intensional atoms positively appearing in the body, (iii) a sequence of elements $a_1, \ldots, a_k \in B$, and (iv) an assignment α in $\mathbf{B}$ such that $Mrg([\![H]\!], a_1, \ldots, a_k, a)$ and all $Enc(a_i, [\![p_i]\!], b_i)$ are true in $\mathbf{B}$, where $p_i = \theta_i[\alpha]$ and H denotes the clause $\gamma_{\mathrm{H}}[\alpha]$. By the inductive assumption, there exist a sequence of clauses $C_1 \vee p_1, \dots, C_k \vee p_k$ in $\Gamma^{\mathbf{A}}_{\Pi} \uparrow_{n-1}$ such that $[\![C_i \vee p_i]\!] = b_i$ and $[\![C_i]\!] = a_i$. Let C denote the clause $H \vee C_1 \vee \cdots \vee C_k$. By definition, C should be in $\Gamma^{\mathbf{A}}_{\Pi} \uparrow_n$. It is also clear that $a = [\![C]\!]$, which implies $a \in [\![\Gamma^{\mathbf{A}}_{\Pi} \uparrow_n]\!]$ as desired. $\square$

Again, unless otherwise mentioned, let us fix $\mathbf{C}$ as an $\upsilon(\Pi^\diamond)$-expansion of $\mathbf{B}$ that interprets $True$ as the set $[\![\Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega]\!]$, and that interprets the predicate $False$ as the set

$$\{[\![C]\!] \mid C \in \textsc{gpc}(\tau, A)\ \&\ \textsc{Ins}(\mathbf{A}, \tau) \models \neg C\} \tag{58}$$

where $\textsc{gpc}(\tau, A)$ denotes the set of all ground positive clauses built from predicates in τ and elements in A. Such a structure is also called a *progression expansion* of $\mathbf{A}$.

LEMMA 2. $\textsc{Ins}(\mathbf{A}, \tau)$ *is a minimal model of* $\Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$ *iff* $\textsc{Ins}(\mathbf{C}, \tau)$ *is a minimal model of* $\Pi^{\mathbf{C}}_5$.

Roughly speaking, the soundness of Lemma 2 is assured by the result that every head-cycle-free disjunctive program is equivalent to a normal program obtained by shifting [Ben-Eliyahu and Dechter 1994]. Note that every set of positive clauses is head-cycle-free, and Π_4 and Π_5 are designed for the simulation of shifting. Now, let us present the proof.

PROOF OF LEMMA 2. We only show the direction "$\Longrightarrow$". The converse can be proved by a routine check in a similar way. Assume that $\textsc{Ins}(\mathbf{A}, \tau)$ is a minimal model of $\Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$. We first show that $\textsc{Ins}(\mathbf{C}, \tau)$ is a model of $\Pi^{\mathbf{C}}_5$. Let $\mathbf{B}$ denote the restriction of $\mathbf{C}$ to σ. It is clear that $\mathbf{B}$ is an encoding expansion of $\mathbf{A}$. By assumption, $\textsc{Ins}(\mathbf{A}, \tau)$ is a model of $\Gamma^{\mathbf{A}}_{\Pi}$. So, we must have that $\bot \notin \Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$. By Lemma 1, the element $c^{\mathbf{C}}_\epsilon$, which encodes the empty clause $\bot$, is then not in $\Delta^n(\mathbf{B})$. Equivalently, $True(c_\epsilon)$ is false in $\mathbf{C}$. This means that the rule (54) is satisfied by $\mathbf{C}$. Let γ be any rule of the form (55) and α an arbitrary assignment in $\mathbf{C}$ such that α satisfies the body of γ in $\mathbf{C}$. By definition, there should exist a clause $C \in \Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$ such that $\alpha(x) = [\![C]\!]$. As $Enc(y, \lceil\theta\rceil, x)$ is satisfied by α in $\mathbf{C}$, by definition there exists a subclause C_0 of C such that $\alpha(y) = [\![C_0]\!]$ and $C = C_0 \vee p$ where p denotes $\theta[\alpha]$. Since α also satisfies $False(y)$ in $\mathbf{C}$ and $\mathbf{C}$ interprets $False$ as the relation (58), we conclude that $\textsc{Ins}(\mathbf{A}, \tau) \models \neg C_0$. On the other hand, we have $\textsc{Ins}(\mathbf{A}, \tau) \cap C \neq \emptyset$ because $\textsc{Ins}(\mathbf{A}, \tau)$ satisfies C. Combining these conclusions, it must hold that $p \in \textsc{Ins}(\mathbf{A}, \tau)$, which implies $p \in \textsc{Ins}(\mathbf{C}, \tau)$ immediately. Thus, $\textsc{Ins}(\mathbf{C}, \tau)$ is indeed a model of $\Pi^{\mathbf{C}}_5$.

Let π be the set of intensional propositional constants of $\Pi^{\mathbf{C}}_5$. Clearly, each intensional propositional constant of $\Pi^{\mathbf{A}}$ is in π. Next, we want to show that $\textsc{Ins}(\mathbf{C}, \tau)$ is a π-minimal model of $\Pi^{\mathbf{C}}_5$. Let M be any model of $\Pi^{\mathbf{C}}_5$ such that $M \subseteq \textsc{Ins}(\mathbf{C}, \tau)$ and $M \setminus \pi = \textsc{Ins}(\mathbf{C}, \tau) \setminus \pi$. It suffices to show that $M \cap \pi \supseteq \textsc{Ins}(\mathbf{C}, \tau) \cap \pi$. We claim that for each atom $p \in \pi \cap \textsc{Ins}(\mathbf{A}, \tau)$ there exists at least one clause, say C_p, in $\Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$ such that $C_p \cap \textsc{Ins}(\mathbf{A}, \tau) = \{p\}$. Otherwise, let $N = \textsc{Ins}(\mathbf{A}, \tau) \setminus \{p\}$; it is obvious that N is also a model of $\Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$, a contradiction. With this claim, for each atom $p \in \pi \cap \textsc{Ins}(\mathbf{C}, \tau) = \pi \cap \textsc{Ins}(\mathbf{A}, \tau)$ there should be a rule γ of the form (55) and an assignment α in $\mathbf{C}$ such that $\theta[\alpha] = p$, $\alpha(x) = [\![C_p]\!]$ and $\mathbf{C} \models Enc(y, \lceil\theta\rceil, x)[\alpha]$. It is clear that $\gamma[\alpha] \in \Pi^{\mathbf{C}}_5$. Let C_0 be the clause obtained from C_p by removing p. By the definition of C_p, it is clear that each atom in C_0 should be false in $\mathbf{A}$ (so it is false in $\mathbf{C}$ too). As $\alpha(y)$ encodes C_0, the ground atom $False(y)[\alpha]$ should be in $\textsc{Ins}(\mathbf{C}, \tau) \setminus \pi = M \setminus \pi$. It is also easy to check that $True(x)[\alpha], Enc(y, \lceil\theta\rceil, x)[\alpha] \in \textsc{Ins}(\mathbf{C}, \tau) \setminus \pi = M \setminus \pi$. Since M satisfies $\gamma[\alpha]$, we must have that $p = \theta[\alpha] \in M$. Thus, we can conclude that $M \cap \pi \supseteq \textsc{Ins}(\mathbf{C}, \tau) \cap \pi$. $\square$

With these lemmas, we can then prove Theorem 2.

PROOF OF THEOREM 2. By Proposition 2, it suffices to show that

$$\mathrm{SM}(\Pi) \equiv_{\mathsf{INF}} \exists\pi(\mathrm{SM}(\Pi_1) \wedge \cdots \wedge \mathrm{SM}(\Pi_5)). \tag{59}$$

"$\Longrightarrow$": Let $\mathbf{A}$ be an infinite model of $\mathrm{SM}(\Pi)$, and $\mathbf{B}$ be an encoding expansion of $\mathbf{A}$. As A is infinite, such an expansion always exists. It is easy to check that $\mathbf{B}$ is a stable model of both Π_1 and Π_2. Let $\mathbf{C}$ be the progression expansion of $\mathbf{A}$ that is also an expansion of $\mathbf{B}$. By Theorem 1, $\textsc{Ins}(\mathbf{B},\tau) = \textsc{Ins}(\mathbf{A},\tau)$ should be a minimal model of $\Gamma^{\mathbf{A}}_{\Pi}\uparrow_{\omega}$. By Lemma 1 and definition, $\textsc{Ins}(\mathbf{B},\tau)$ is also a minimal model of $\Gamma^{\mathbf{B}}_{\Pi_3}\uparrow_{\omega}$. By Theorem 1 again, $\mathbf{B}$ is then a stable model of Π_3, which implies that so is $\mathbf{C}$. It is also easy to check that $\mathbf{C}$ is a stable model of Π_4. On the other hand, since $\textsc{Ins}(\mathbf{A},\tau)$ is a minimal model of $\Gamma^{\mathbf{A}}_{\Pi}\uparrow_{\omega}$, by Lemma 2, $\textsc{Ins}(\mathbf{C},\tau)$ should be a minimal model of $\Pi^{\mathbf{C}}_5$, which means that $\mathbf{C}$ is a stable model of Π_5 by Proposition 1. Thus, $\mathbf{A}$ is a model of the right-hand side of (59).

"$\Longleftarrow$": Let $\mathbf{A}$ be an infinite model of the right-hand side of (59). Then there exists an $\upsilon(\Pi^{\diamond})$-expansion $\mathbf{C}$ of $\mathbf{A}$ such that $\mathbf{C}$ satisfies $\mathrm{SM}(\Pi_i)$ for all $i \in \{1,\ldots,5\}$. Let $\mathbf{B}$ be the restrictions of $\mathbf{C}$ to σ. Then, by a routine check, it is easy to show that $\mathbf{B}$ is an encoding expansion of $\mathbf{A}$. As $\mathbf{C}$ is a stable model of Π_3, by Theorem 1, $\textsc{Ins}(\mathbf{C},\upsilon_3)$ is then a minimal model of $\Gamma^{\mathbf{C}}_{\Pi_3}\uparrow_{\omega} = \Gamma^{\mathbf{B}}_{\Pi_3}\uparrow_{\omega}$. Furthermore, by Lemma 1 and the conclusion that $\mathbf{C}$ satisfies $\mathrm{SM}(\Pi_4)$, we then have that $\mathbf{C}$ is a progression expansion of $\mathbf{A}$. On the other hand, since $\mathbf{C}$ is also a stable model of Π_5, by Proposition 1 we can conclude that $\textsc{Ins}(\mathbf{C},\tau)$ is a minimal model of $\Pi^{\mathbf{C}}_5$. Thus, by Lemma 2 we immediately have that $\textsc{Ins}(\mathbf{A},\tau)$ is a minimal model of $\Gamma^{\mathbf{A}}_{\Pi}\uparrow_{\omega}$. By Theorem 1, $\mathbf{A}$ must be a stable model of Π, which completes the proof. $\square$

REMARK 3. *Note that, given any finite domain A, there is no injective function from $A \times A$ into A. Thus, we cannot expect that the above translation works on finite structures.*

COROLLARY 1. DLP $\simeq_{\mathsf{INF}}$ NLP.

Now, let us focus on the relationship between logic programs and second-order logic. The following proposition says that, over infinite structures, normal programs are more expressive than the existential second-order logic, which then strengthens a result in [Asuncion et al. 2012] where such a separation over arbitrary structures was obtained.

PROPOSITION 3. NLP $\not\leq_{\mathsf{INF}} \Sigma^1_1$.

To show this, our main idea is to find a property that can be defined by a normal program but not by any existential second-order sentence. A property that satisfies the mentioned conditions is defined as follows. Let υ_{R} be the vocabulary consisting of a binary predicate E and two individual constants c and d. Let Reach_i be the class of infinite υ_{R}-structures in each of which there is a finite path from c to d via edges in E. Now, we show the result.

PROOF OF PROPOSITION 3. We first show that Reach_i is definable in NLP over infinite structures. Let Π be a normal program that consists of the following rules:

$$P(c), \tag{60}$$

$$P(x) \wedge E(x,y) \rightarrow P(y), \tag{61}$$

$$\neg P(d) \rightarrow \bot. \tag{62}$$

It is easy to see that $\exists P\mathrm{SM}(\Pi)$ defines the property Reach_i over infinite structures.

Next, we prove that Reach_i is undefinable in Σ^1_1 over infinite structures. Towards a contradiction, assume that there is a first-order sentence φ and a finite set τ of predicates such that $\exists\tau\varphi$ is in Σ^1_1 and defines Reach_i over infinite structures. Let R be a binary predicate that has no occurrence in τ, and let ψ denote the formula

$$\forall x \exists y R(x,y) \wedge \forall x \neg R(x,x) \wedge \forall x \forall y \forall z (R(x,y) \wedge R(y,z) \rightarrow R(x,z)). \tag{63}$$

Intuitively, it asserts that the relation R is transitive and irreflexive, and each element in the domain has a successor w.r.t. R. It is obvious that such a relation exists iff the domain is infinite. Thus, the formula $\exists\tau\varphi \wedge \exists R\psi$ defines Reach_i over arbitrary structures.

Moreover, let $\gamma_0(x,y)$ be $x = y$; for $n > 0$ let $\gamma_n(x,y)$ denote the formula

$$\exists z_n(\gamma_{n-1}(x,z_n) \wedge E(z_n,y)), \tag{64}$$

where each $\gamma_n(x,y)$ asserts that there is a path of length n from x to y. Let Λ be the set of sentences $\neg\gamma_n(c,d)$ for all $n \geq 0$. Now we prove a property as follows.

Claim. $\Lambda \cup \{\exists\tau\varphi, \exists R\psi\}$ is satisfiable.

To show this, it suffices to show that the first-order theory $\Lambda \cup \{\varphi, \psi\}$ is satisfiable. Let Φ be a finite subset of Λ, and let $n = \max\{m \mid \neg\gamma_m(c,d) \in \Phi\}$. Let $\mathbf{A}_n$ be an infinite model of ψ of the vocabulary $\upsilon(\varphi) \cup \upsilon(\psi)$ in which the minimal length of paths from c to d via edges in E is greater than n. Then $\mathbf{A}_n$ is clearly a model of $\Phi \cup \{\varphi, \psi\}$. Due to the arbitrariness of Φ, by the compactness we then have the desired claim.

Let $\mathbf{A}$ be any model of $\Lambda \cup \{\exists\tau\varphi, \exists R\psi\}$. Then according to $\exists R\psi$, $\mathbf{A}$ should be infinite, and by Λ, there is no path from c to d via E in $\mathbf{A}$. However, according to $\exists\tau\varphi$, every infinite model of it should be c-to-d reachable, a contradiction. Thus, the property Reach_i is then undefinable in Σ^1_1 over infinite structures. This completes the proof immediately. $\square$

The following separation immediately follows from the proof of Theorem 4.1 in [Eiter et al. 1996]. Although their statement refers to the whole class of arbitrary structures, the proof still works if only infinite structures are considered.

PROPOSITION 4. $\Sigma^1_2 \not\leq_{\mathsf{INF}}$ DLP.

4. FINITE STRUCTURES

In this section we focus on the expressiveness of logic programs over finite structures. We first consider the relationship between disjunctive and normal programs. Unfortunately, in the general case, it is not hard to obtain the following result. Note that the direction "only if" is already implicit in [Asuncion et al. 2012].

PROPOSITION 5. DLP $\simeq_{\mathsf{FIN}}$ NLP *iff* NP $=$ coNP.[4]

PROOF. By Fagin's Theorem [Fagin 1974] and Stockmeyer's logical characterization of the polynomial-time hierarchy [Stockmeyer 1977],[5] we have that $\Sigma^1_2 \simeq_{\mathsf{FIN}} \Sigma^1_1$ iff $\Sigma^p_2 =$ NP. By a routine complexity-theoretic argument, it is also true that $\Sigma^p_2 =$ NP iff NP $=$ coNP. On the other hand, by Proposition 7, Leivant's normal form [Leivant 1989] and the definition of SM, we can conclude that DLP $\simeq_{\mathsf{FIN}} \Sigma^1_2$. By Proposition 6, it holds that NLP $\simeq_{\mathsf{FIN}} \Sigma^1_1$. Combining these conclusions, we then have the desired proposition. $\square$

[4] The proposition was first presented in an earlier version of this paper [Zhang and Zhang 2013a]. Recently, such an equivalence was also observed by Zhou [2015]. For traditional logic programs under the query equivalence, a similar result follows from the expressiveness results proved by [Eiter et al. 1997; Schlipf 1995].

[5] In their characterizations of complexity classes, no function constant of positive arity is allowed. However, this restriction can be removed since functions can be easily simulated by predicates and first-order quantifiers.

This result shows how difficult it is to separate normal programs from disjunctive programs over finite structures. To know more about the relationship, we will try to prove a weaker separation between these two classes. To achieve this goal, we need to study the relationship between logic programs and second-order logic.

For the class of normal programs, we have the following characterization:

PROPOSITION 6. $\mathrm{NLP}^{\mathrm{F}}_n \simeq_{\mathsf{FIN}} \Sigma^{1\mathrm{F}}_{1,n}[\forall^*]$ *for all* $n > 1$.

To prove it, we have to develop a translation that turns normal programs to first-order sentences. The main idea is to extend the Clark completion by simulating the progression.

Now, let us give the translation. Let Π be a normal program and n be the maximum arity of intensional predicates of Π. Without loss of generality, assume the head of every rule in Π is of the form $P(\vec{x})$, where P is a k-ary predicate for some $k \geq 0$, and $\vec{x}$ is the tuple of distinct individual variables $x_1, \ldots, x_k$. Let $\prec$ be a new binary predicate and ϖ a universal first-order sentence asserting that $\prec$ is a strict linear order. Given two tuples $\vec{s}$ and $\vec{t}$ of terms with the same length, let $\vec{s} \prec \vec{t}$ be a quantifier-free formula asserting that $\vec{s}$ is less than $\vec{t}$ w.r.t. the lexicographic order extended from $\prec$ naturally. For example, if $\vec{s}$ and $\vec{t}$ denote the tuples (s_1, s_2) and (t_1, t_2), respectively, then one can use the following quantifier-free formula to express that $\vec{s}$ is less than $\vec{t}$ w.r.t. the extended strict linear order:

$$s_1 \prec t_1 \vee (s_1 = t_1 \wedge s_2 \prec t_2). \tag{65}$$

Let τ be the set of intensional predicates of Π. Let c be the least integer that is not less than $\log_2 |\tau| + n$. We fix P as a k-ary predicate in τ and let $\lambda = P(x_1, \ldots, x_k)$. Suppose $\gamma_1, \ldots, \gamma_l$ list all rules in Π whose heads are λ, and suppose each γ_i is of the form

$$\eta^i \wedge \theta^i_1 \wedge \cdots \wedge \theta^i_{m_i} \rightarrow \lambda \tag{66}$$

where $\theta^i_1, \ldots, \theta^i_{m_i}$ list all positive intensional conjuncts in the body of γ_i, η^i is the conjunction of other conjuncts that occur in the body of γ_i, $m_i \geq 0$, and $\vec{y}_i$ is the tuple of all individual variables that occur in γ_i but not in λ.

Next, we let ψ_P denote the following first-order sentence:

$$\forall x_1 \cdots \forall x_k \left[\lambda \rightarrow \bigvee_{i=1}^{l} \exists \vec{y}_i \left[\eta^i \wedge \bigwedge_{j=1}^{m_i} \left(\theta^i_j \wedge DLess(\theta^i_j, \lambda)\right)\right]\right] \tag{67}$$

where, for all intensional atoms θ and θ_0, let $\textsc{ord}(\theta)$ denote the tuple $(o^c_Q(\vec{t}), \cdots, o^1_Q(\vec{t}))$ if θ is of the form $Q(\vec{t})$; let $o^1_Q, \ldots, o^c_Q$ be fresh function constants whose arities are the same as that of Q; and let $DLess(\theta, \theta_0)$ denote the quantifier-free formula $\textsc{ord}(\theta) \prec \textsc{ord}(\theta_0)$.

Now we define φ_Π as a conjunction of the sentences ϖ and $\widehat{\Pi}$, and ψ_P for all $P \in \tau$. Let σ denote the set of function constants o^s_Q for all $Q \in \tau$ and all $s \in \{1, \ldots, c\}$. Clearly, $\exists \sigma \varphi_\Pi$ is equivalent to a sentence in $\Sigma^{1\mathrm{F}}_{1,n}[\forall^*]$ by introducing Skolem functions if n (i.e., the maximum arity of intensional predicates of Π) is greater than 1.

EXAMPLE 10 (EXAMPLE 5 CONTINUED). *Let* Π *denote the following normal program which is obtained from the program presented in Example 5 by a normalization:*

$$y = a \rightarrow P(y), \tag{68}$$

$$P(x) \wedge E(x, y) \rightarrow P(y), \tag{69}$$

$$y = b \wedge \neg P(b) \rightarrow P(y). \tag{70}$$

Clearly, the only intensional predicate constant is P. Let $\tau = \{P\}$. Then we have that

$$c = \log_2 |\tau| + n = 0 + 1 = 1, \tag{71}$$

where n denotes the maximum arity of predicate constants that occur in τ. According to the translation defined above, ψ_P is the following first-order formula:

$$\forall y(P(y) \to y = a \vee \exists x(E(x,y) \wedge P(x) \wedge o(x) \prec o(y)) \vee (y = b \wedge \neg P(b))) \tag{72}$$

where o is a fresh unary function. Moreover, φ_Π is the formula $\varpi \wedge \widehat{\Pi} \wedge \psi_P$. Clearly, $\exists o \varphi_\Pi$ is equivalent to the $\Sigma^{1\mathrm{F}}_{1,n}[\forall^*]$-sentence $\exists o \exists f(\varpi \wedge \widehat{\Pi} \wedge \vartheta)$, where ϑ denotes the formula

$$\forall y(P(y) \to y = a \vee (E(f(y),y) \wedge P(f(y)) \wedge o(f(y)) \prec o(y)) \vee (y = b \wedge \neg P(b))) \tag{73}$$

and f is a unary Skolem function introduced to eliminate the existential quantifier $\exists x$.

Next, let us show the soundness of the presented translation:

LEMMA 3. *Given any finite structure $\mathbf{A}$ of $\upsilon(\Pi)$ with at least two elements in the domain, we have $\mathbf{A} \models \mathrm{SM}(\Pi)$ iff $\mathbf{A} \models \exists \sigma \varphi_\Pi$.*

PROOF. "$\Longrightarrow$": Let $\mathbf{A}$ be a finite stable model of Π with at least two elements in the domain. Let N be the cardinality of A, τ be the set of intensional predicates of Π, n be the maximum arity of intensional predicates in τ, and c be the least integer that is not less than $\log_2 |\tau| + n$. Without loss of generality, let us assume that A is the set of natural numbers less than N. Note that every finite structure with domain size N is isomorphic to a structure over this domain. Given any ground intensional atom p, we define

$$\ell(p) = \max\{m < N^c \mid p \notin \Gamma^{\mathbf{A}}_\Pi \uparrow_m\}. \tag{74}$$

Let $\mathbf{B}$ be an $\upsilon(\varphi_\Pi)$-expansion of $\mathbf{A}$ in which

(1) the predicate constant $\prec$ is interpreted as the relation $\{(a,b) \in A \times A \mid a < b\}$;
(2) for each predicate constant $P \in \tau$ of arity $k \geq 0$ and each integer $i \in \{1, \ldots, c\}$, the function constant o^i_P is interpreted as a function g of arity k such that $g(\vec{a}) = d_i$ for all k-tuples $\vec{a}$ on A if $(d_c, \ldots, d_1)$ is the representation of $\ell(P(\vec{a}))$ in the base-N numeral system. (For example, suppose $N = 8, c = 3$ and $\ell(P(\vec{a})) = 22$; then the desired representation of $\ell(P(\vec{a}))$ in the base-8 numeral system is $(0, 2, 6)$.)

By a routine check, one can show that $\mathbf{B}$ is a model of φ_Π. From this, we know that $\mathbf{A}$ is indeed a model of $\exists \sigma \varphi_\Pi$, which proves the desired direction.

"$\Longleftarrow$": Let $\mathbf{A}$ be a finite model of $\exists \sigma \varphi_\Pi$. We want to show that $\mathbf{A}$ is a stable model of Π. By Theorem 1, it suffices to show that $\textsc{Ins}(\mathbf{A}, \tau) = \Gamma^{\mathbf{A}}_\Pi \uparrow_\omega$. Clearly, there exists a model $\mathbf{B}$ of φ_Π that is an $\upsilon(\varphi_\Pi)$-expansion of $\mathbf{A}$. By the formula ϖ, we know that $\mathbf{B}$ must interpret $\prec$ as a strict linear order on B. We first show that

$(*)$ $\Gamma^{\mathbf{A}}_\Pi \uparrow_s \subseteq \textsc{Ins}(\mathbf{A}, \tau)$ for all $s \geq 0$.

This can be done by an induction on s. The case of $s = 0$ is trivial. Let $s > 0$ and assume that $\Gamma^{\mathbf{A}}_\Pi \uparrow_{s-1} \subseteq \textsc{Ins}(\mathbf{A}, \tau)$. Our task is to show that $\Gamma^{\mathbf{A}}_\Pi \uparrow_s \subseteq \textsc{Ins}(\mathbf{A}, \tau)$. Let p be a ground atom in $\Gamma^{\mathbf{A}}_\Pi \uparrow_s$. By definition, there is a rule $\gamma_i \in \Pi$ of the form (same as (66))

$$\eta^i \wedge \theta^i_1 \wedge \cdots \wedge \theta^i_{m_i} \to \lambda \tag{75}$$

and an assignment α in $\mathbf{A}$ such that (i) $\lambda[\alpha] = p$, (ii) α satisfies η^i in $\mathbf{A}$ (so equivalently, in $\mathbf{B}$), and (iii) for each atom θ^i_j, it holds that $\theta^i_j[\alpha] \in \Gamma^{\mathbf{A}}_\Pi \uparrow_{s-1}$. By the inductive assumption, we have $\theta^i_j[\alpha] \in \textsc{Ins}(\mathbf{A}, \tau)$, which means that $\mathbf{B} \models \theta^i_j[\alpha]$. As α clearly satisfies the rule γ_i in $\mathbf{B}$, we conclude that α satisfies λ in $\mathbf{B}$, which implies that

$$p = \lambda[\alpha] \in \textsc{Ins}(\mathbf{B}, \tau) = \textsc{Ins}(\mathbf{A}, \tau). \tag{76}$$

So, the claim $(*)$ is true. From this, we obtain that $\Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega \subseteq \textsc{Ins}(\mathbf{A}, \tau)$ as desired.

Now, it remains to show that $\textsc{Ins}(\mathbf{A}, \tau) \subseteq \Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$. Towards a contradiction, assume it is not true. We then have that $\Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega \subsetneq \textsc{Ins}(\mathbf{A}, \tau)$ by the previous conclusion. For all ground atoms p_1 and p_2 in $\textsc{Ins}(\mathbf{A}, \tau)$, we write $p_1 < p_2$ if $DLess(p_1, p_2)$ is true in $\mathbf{B}$. Let p be a $<$-minimal atom in $\textsc{Ins}(\mathbf{A}, \tau) \setminus \Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$ and suppose $p = P(\vec{a})$ for some $P \in \tau$. Let α be an assignment in $\mathbf{B}$ such that $\alpha(\vec{x}) = \vec{a}$. By definition, α should satisfy ψ_P (in which $\lambda[\alpha] = p$) in $\mathbf{B}$. So, there is an integer $i \in \{1, \ldots, l\}$ and an assignment α_0 in $\mathbf{B}$ such that

(1) $\alpha_0(\vec{x}) = \vec{a}$,
(2) $\eta^i[\alpha_0]$ is true in $\mathbf{B}$, and
(3) for all integers $j \in \{1, \ldots, m_i\}$, $q_j \in \textsc{Ins}(\mathbf{B}, \tau)$ (or equivalently, $q_j \in \textsc{Ins}(\mathbf{A}, \tau)$) and $q_j < \lambda[\alpha_0]$, where q_j denotes the ground atom $\theta^i_j[\alpha_0]$.

Since $\lambda[\alpha_0] = \lambda[\alpha] = p$ and p is a $<$-minimal atom in $\textsc{Ins}(\mathbf{A}, \tau) \setminus \Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$, we conclude that $q_j \in \Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$ for all integers $j \in \{1, \ldots, m_i\}$. According to the definition of ψ_P, the rule γ_i (of the form (66)) is in Π, which implies that

$$\gamma^+_i[\alpha_0] = q_1 \wedge \cdots \wedge q_{m_i} \to p \in \Pi^{\mathbf{A}}. \tag{77}$$

By definition, we have that $p \in \Gamma^{\mathbf{A}}_{\Pi} \uparrow_\omega$, a contradiction as desired. $\square$

REMARK 4. *Let m denote the number of intensional predicates in Π, and let n denote the maximal arity of intensional predicates in Π. The maximal arity of auxiliary constants in our translation is only n, which is optimal if Conjecture 1 in [Durand et al. 2004]*[6] *is true. Note that the maximal arity of auxiliary constants of the ordered completion in [Asuncion et al. 2012] is $2n$. Moreover, the number of auxiliary constants in our translation is $m \cdot (\lceil \log_2 m \rceil + n)$, while that of the ordered completion is m^2.*

REMARK 5. *Similar to the work in [Asuncion et al. 2012], one can develop an answer set solver based on our translation by calling some SMT solver. From the theoretical comparison given in the last remark, we believe that the approach proposed here is rather promising. In addition, as a strict partial order is available in almost all the SMT solvers (e.g., built-in arithmetic relations), our translation can be easily optimized.*

Now we are in the position to prove Proposition 6.

PROOF OF PROPOSITION 6. "$\geq_{\mathsf{FIN}}$": Let φ be any sentence in $\Sigma^{1\mathrm{F}}_{1,n}[\forall^*]$. It is obvious that φ can be written as an equivalent sentence of the form $\exists\tau\forall\vec{x}(\eta_1 \wedge \cdots \wedge \eta_k)$ for some $k \geq 0$, where each η_i is a disjunction of literals, i.e., atoms or negated atoms, and τ a finite set of functions or predicates of arity $\leq n$. Let Π be a logic program consisting of the rule $\tilde{\eta}_i \to \bot$ for each $i \in \{1, \ldots, k\}$, where $\tilde{\eta}_i$ is obtained from η_i by substituting θ for each negated atom $\neg\theta$, followed by substituting $\neg\theta$ for each atom θ, and followed by substituting $\wedge$ for $\vee$. It is easy to check that $\exists\tau\mathrm{SM}(\Pi)$ is in $\mathrm{NLP}^{\mathrm{F}}_n$ and equivalent to φ.

[6]Conjecture 1 in [Durand et al. 2004] implies that $\mathrm{ESO}^{\mathrm{F}}_n[\forall^*] \simeq_{\mathsf{FIN}} \mathrm{ESO}^{\mathrm{F}}_n[\forall^n]$, where $\mathrm{ESO}^{\mathrm{F}}_n[\forall^n]$ denotes the class of sentences in $\mathrm{ESO}^{\mathrm{F}}_n[\forall^*]$ that involve at most n individual variables.

"$\leq_{\mathsf{FIN}}$": Let $\mathcal{C}^{=1}$ (respectively, $\mathcal{C}^{>1}$) denote the class of finite structures with exactly one (respectively, at least two) element(s) in the domain. Let Π be a normal program and τ be a finite set of predicates and functions such that $\exists\tau\mathrm{SM}(\Pi)$ is in $\mathrm{NLP}^{\mathrm{F}}_n$. It is trivial to construct a sentence, say ζ, in $\Sigma^{1\mathrm{F}}_{1,n}[\forall^*]$ such that $\exists\tau\mathrm{SM}(\Pi)$ is equivalent to ζ over $\mathcal{C}^{=1}$. (Note that, if the domain consists of only one element, a first-order logic program will regress to a propositional one.) By Lemma 3, there is also a sentence ψ in $\Sigma^{1\mathrm{F}}_{1,n}[\forall^*]$ such that $\exists\tau\mathrm{SM}(\Pi)$ is equivalent to ψ over $\mathcal{C}^{>1}$. Let φ denote the following sentence:

$$(\exists x\forall y(x = y) \wedge \zeta) \vee (\exists x\exists z(\neg x = z) \wedge \psi)\,. \tag{78}$$

Informally, the formula φ first test whether the intended domain is a singleton or not. ζ will be activated if the answer is true, and ψ will be activated otherwise. Thus, it is easy to show that $\exists\tau\mathrm{SM}(\Pi)$ is equivalent to φ over finite structures. It is also clear that φ can be written to be an equivalent sentence in $\Sigma^{1\mathrm{F}}_{1,n}[\forall^*]$. Note that every first-order quantifier can be regarded as a second-order quantifier over a function variable of arity 0. $\square$

REMARK 6. *Again, if one assumes Conjecture 1 in [Durand et al. 2004], by the main result of [Grandjean 1985],* $\mathrm{NLP}^{\mathrm{F}}_k$ *then exactly captures the class of languages computable in $O(n^k)$-time (where n denotes the size of the input) in Nondeterministic Random Access Machines (NRAMs), which implies that whether an extensional database can be expanded to a stable model of a disjunctive program can be checked in $O(n^k)$-time in an NRAM.*

By Proposition 6 and the fact that functions can be simulated by introducing auxiliary predicates in both logic programs and second-order logic, we have the following result:

COROLLARY 2. $\mathrm{NLP} \simeq_{\mathsf{FIN}} \Sigma^1_1$.

To establish the mentioned separation, we still need to investigate the translatability from a fragment of second-order logic to disjunctive programs. For convenience, in the rest of this paper, we fix $Succ$ as a binary predicate, $First$ and $Last$ as two unary predicates, and υ_{S} as the set that consists of these predicates. Unless otherwise mentioned, every logic program or formula to be considered is always assumed to contains no predicate from υ_{S}.

A structure $\mathbf{A}$ is called a *successor structure* if all of the following hold:

(1) the vocabulary of $\mathbf{A}$ contains all the predicates in υ_{S}, and
(2) $Succ^{\mathbf{A}}$ is a binary relation S on A such that the transitive closure of S is a strict linear order, and that $|\{b\,|\,(a,b)\in S\}| \leq 1$ and $|\{b\,|\,(b,a)\in S\}| \leq 1$ for all $a \in A$, and
(3) $First^{\mathbf{A}}$ (respectively, $Last^{\mathbf{A}}$) consists of the least element (respectively, the largest element) in A w.r.t. the strict linear order defined by $Succ^{\mathbf{A}}$.

Note that, by definition, both the least and largest elements must exist in a successor structure. This means that every successor structure is always finite.

Let SUC denote the class of successor structures, and let $\Sigma^1_{2,n}[\forall^n\exists^*]$ denote the class of sentences in $\Sigma^1_{2,n}[\forall^*\exists^*]$ that involve at most n universal quantifiers. Now let us show that:

LEMMA 4. $\Sigma^1_{2,n}[\forall^n\exists^*] \leq_{\mathsf{SUC}} \mathrm{DLP}_n$ *for all* $n > 0$.

PROOF. Fix $n > 0$, and let $\exists\tau\forall\sigma\varphi$ be a sentence in $\Sigma^1_{2,n}[\forall^n\exists^*]$, where τ and σ are two finite sets of predicates of arities $\leq n$. It suffices to show that there is a disjunctive program Π and a set π of auxiliary predicates such that $\forall\sigma\varphi \equiv_{\mathsf{SUC}} \exists\pi\mathrm{SM}(\Pi)$ and $\exists\pi\mathrm{SM}(\Pi)$ belongs to DLP_n. Without loss of generality, let us assume that φ is of the form

$$\forall\vec{x}\exists\vec{y}(\vartheta_1(\vec{x},\vec{y}) \vee \cdots \vee \vartheta_m(\vec{x},\vec{y})), \tag{79}$$

where $m \geq 0$, each ϑ_i is a finite conjunction of literals, and the length of $\vec{x}$ is exactly n.

Before presenting the program Π, let us first define some notations. Suppose $\vec{u}$ and $\vec{v}$ are two tuples of individual variables $u_1 \cdots u_k$ and $v_1 \cdots v_k$, respectively. Let $First(\vec{u})$ denote the conjunction of $First(u_i)$ for all $i \in \{1, \ldots, n\}$, and $Last(\vec{u})$ denote the conjunction of $Last(u_i)$ for all $i \in \{1, \ldots, n\}$. Given $i \in \{1, \ldots, n\}$, let $Succ_i(\vec{u}, \vec{v})$ denote the formula

$$\begin{aligned} u_1 = v_1 \wedge \cdots \wedge u_{i-1} = v_{i-1} \wedge Succ(u_i, v_i) \\ \wedge\ Last(u_{i+1}) \wedge First(v_{i+1}) \wedge \cdots \wedge Last(u_n) \wedge First(v_n). \end{aligned} \tag{80}$$

Now, by employing the well-known saturation technique (see, e.g., the proof of Theorem 6.3 in [Eiter et al. 1997]), we define the desired logic program Π as follows:

$$X(\vec{z}) \vee \tilde{X}(\vec{z}) \qquad (X \in \sigma) \tag{81}$$
$$Last(\vec{x}) \wedge D(\vec{x}) \to X(\vec{z}) \qquad (X \in \sigma) \tag{82}$$
$$Last(\vec{x}) \wedge D(\vec{x}) \to \tilde{X}(\vec{z}) \qquad (X \in \sigma) \tag{83}$$
$$First(\vec{x}) \wedge \tilde{\vartheta}_i(\vec{x}, \vec{y}) \to D(\vec{x}) \qquad (1 \leq i \leq m) \tag{84}$$
$$Succ_j(\vec{v}, \vec{x}) \wedge D(\vec{v}) \wedge \tilde{\vartheta}_i(\vec{x}, \vec{y}) \to D(\vec{x}) \qquad (1 \leq i \leq m, 1 \leq j \leq n) \tag{85}$$
$$Last(\vec{x}) \wedge \neg D(\vec{x}) \to \bot \tag{86}$$

where D is an n-ary fresh predicate; for each predicate $X \in \sigma$ of arity $n \geq 0$, let $\tilde{X}$ be a fresh predicate of arity n; and $\tilde{\vartheta}_i$ is the formula obtained from ϑ_i by substituting $\tilde{X}$ for the expression $\neg X$ whenever $X \in \sigma$. Let π denote the predicate set

$$\sigma \cup \{D\} \cup \{\tilde{X} : X \in \sigma\}. \tag{87}$$

It is easy to see that $\exists\pi\mathrm{SM}(\Pi)$ belongs to DLP^n. Next we show that $\exists\pi\mathrm{SM}(\Pi) \equiv_{\mathsf{SUC}} \forall\sigma\varphi$ by employing a similar idea used in the proof of Theorem 6.3 in [Eiter et al. 1997].

Let $\mathbf{A}$ be a successor structure of $\upsilon(\forall\tau\varphi) \cup \upsilon_{\mathrm{s}}$ that satisfies $\forall\sigma\varphi$. Our task now is to prove that $\mathbf{A}$ is a model of $\exists\pi\mathrm{SM}(\Pi)$. Let $\mathbf{B}$ be an $\upsilon(\Pi)$-expansion of $\mathbf{A}$ in which each of the predicates among D, X, $\tilde{X}$ ($X \in \sigma$) is interpreted as the full relation on A of a proper arity. To obtain the desired conclusion, it is sufficient to show that $\mathbf{B}$ is a stable model of Π. It is clear that $\mathbf{B}$ is a model of $\widehat{\Pi}$. Towards a contradiction, let us assume that $\mathbf{B}$ is not a stable model of Π, which implies that there is some assignment β in $\mathbf{B}$ such that

$$\mathbf{B} \models (\pi^* < \pi)[\beta] \qquad \& \qquad \mathbf{B} \models \widehat{\Pi}^*[\beta], \tag{88}$$

where both notations π^* and $\widehat{\Pi}^*$ are defined in the same way as those in Section 2. It is not difficult to see that there is some tuple $\vec{a} \in A^n$ such that $\vec{a} \notin \beta(D^*)$. Let $\vec{b}$ be an arbitrary tuple on A of length $|\vec{y}|$. According to rules (84)–(85) it must be true that

$$\mathbf{B} \models (\neg\tilde{\vartheta}_1(\vec{a}, \vec{b}) \wedge \cdots \wedge \neg\tilde{\vartheta}_m(\vec{a}, \vec{b}))[\beta]. \tag{89}$$

Let α be an assignment in $\mathbf{A}$ such that $\alpha(X) = \beta(X^*)$ for all $X \in \sigma$. We can infer that

$$\mathbf{A} \models (\neg\vartheta_1(\vec{a}, \vec{b}) \wedge \cdots \wedge \neg\vartheta_m(\vec{a}, \vec{b}))[\alpha], \tag{90}$$

which is impossible since we have that $\mathbf{A} \models \forall\sigma\varphi$, a contradiction as desired.

For the converse, let us assume that $\mathbf{A}$ is a model of $\exists\pi\mathrm{SM}(\Pi)$. Towards a contradiction, we also assume that $\mathbf{A}$ is not a model of $\forall\sigma\varphi$. By the former, we know that there is a stable model $\mathbf{B}$ of Π that is an $\upsilon(\Pi)$-expansion of $\mathbf{A}$. By the definition of Π, it is easy to see that

$\mathbf{B}$ interprets each predicate among $D, X, \tilde{X}$ ($X \in \sigma$) as the full relation on A of a proper arity. By the assumption that $\mathbf{A} \models \neg\forall\sigma\varphi$, there is an assignment α in $\mathbf{A}$ such that

$$\mathbf{A} \models \neg\forall\vec{x}\exists\vec{y}(\vartheta_1(\vec{x},\vec{y}) \vee \cdots \vee \vartheta_m(\vec{x},\vec{y}))[\alpha]. \tag{91}$$

With this conclusion, we know that there exists a tuple $\vec{a} \in A^n$ such that for all tuples $\vec{b}$ on A of length $|\vec{y}|$ and all integers $i \in \{1,\ldots,m\}$ we have that $\mathbf{A} \models \neg\vartheta_i(\vec{a},\vec{b})[\alpha]$. Let β be an assignment in $\mathbf{B}$ such that $\beta(D^*) = A^n \setminus \{\vec{a}\}$ and that

$$\beta(X^*) = \alpha(X) \qquad \& \qquad \beta(\tilde{X}^*) = A^k \setminus \alpha(X) \tag{92}$$

for all predicates $X \in \sigma$ of arity k. Then, it is a routine task to check that

$$\mathbf{B} \models (\pi^* < \pi)[\beta] \qquad \& \qquad \mathbf{B} \models \widehat{\Pi}^*[\beta]. \tag{93}$$

From these, we conclude that $\mathbf{B}$ is not a model of $\mathrm{SM}(\Pi)$, a contradiction as desired. $\square$

Next, we show that this result can be generalized to finite structures. To do this, we need a logic program to define the class of successor structures. Now we define it as follows.

Let Π_{S} denote the normal program that consists of the following rules.

$$\neg\underline{Less}(x,y) \to Less(x,y) \tag{94}$$
$$\neg Less(x,y) \to \underline{Less}(x,y) \tag{95}$$
$$Less(x,y) \wedge Less(y,z) \to Less(x,z) \tag{96}$$
$$Less(x,y) \wedge Less(y,x) \to \bot \tag{97}$$
$$\neg Less(x,y) \wedge \neg Less(y,x) \wedge \neg x = y \to \bot \tag{98}$$
$$Less(x,y) \to \underline{First}(y) \tag{99}$$
$$\neg\underline{First}(x) \to First(x) \tag{100}$$
$$Less(x,y) \to \underline{Last}(x) \tag{101}$$
$$\neg\underline{Last}(x) \to Last(x) \tag{102}$$
$$Less(x,y) \wedge Less(y,z) \to \underline{Succ}(x,z) \tag{103}$$
$$\neg\underline{Succ}(x,y) \wedge Less(x,y) \to Succ(x,y) \tag{104}$$

Informally, rules (94)–(95) are choice rules to guess a binary relation $Less$; rules (96), (97) and (98) restrict $Less$ to be transitive, antisymmetric, and total, respectively, so that $Less$ is a strict linear order; rules (99)–(100) and rules (101)–(102) then assert that $First$ and $Last$ consist of the least and the last elements, respectively, if they indeed exist; and the last two rules state that $Succ$ defines the relation of immediate successors w.r.t. $Less$.

The following lemma, which can be proved by a routine check, asserts that the program Π_{S} exactly defines the class of successor structures as we expect.

LEMMA 5. *Let $\sigma \supseteq \upsilon_{\mathrm{S}}$ be a vocabulary and $\mathbf{A}$ be a σ-structure. Then $\mathbf{A}$ is a successor structure iff both $\mathbf{A}$ is finite and $\mathbf{A}$ is a model of $\exists\tau\mathrm{SM}(\Pi_{\mathrm{S}})$, where $\tau = \upsilon(\Pi_{\mathrm{S}}) \setminus \upsilon_{\mathrm{S}}$.*

Now we can then prove the following result:

PROPOSITION 7. $\Sigma^1_{2,n}[\forall^n\exists^*] \leq_{\mathsf{FIN}} \mathrm{DLP}_n$ *for all* $n > 1$.

PROOF. Let $n > 1$ be an integer, and φ be a sentence in $\Sigma^1_{2,n}[\forall^n\exists^*]$. Let Π_0 be the disjunctive program constructed in the proof of Lemma 4 which expresses φ, and let σ be the set of predicates that appear in Π_0 but neither in υ_{S} nor in $\upsilon(\varphi)$. Let $\Pi = \Pi_0 \cup \Pi_{\mathrm{S}}$ and let τ be the set of predicates that appear in Π_s but not in υ_s. Next, our task is to show that φ is equivalent to $\exists\tau\exists\sigma\mathrm{SM}(\Pi)$ over finite structures. By definition and Proposition 2, it suffices to show that $\varphi \equiv_{\mathsf{FIN}} \psi$, where ψ denotes the following formula:

$$\exists\upsilon_s(\exists\tau\mathrm{SM}(\Pi_s) \wedge \exists\sigma\mathrm{SM}(\Pi_0)). \tag{105}$$

Let υ denote the union of $\upsilon(\varphi)$ and υ_s. Now we prove the new statement as follows.

"$\Longrightarrow$": Let $\mathbf{A}$ be a finite model of φ. Clearly, there exists an υ-expansion $\mathbf{B}$ of $\mathbf{A}$ such that $\mathbf{B}$ is a successor structure. By Lemma 5, $\mathbf{B}$ should be a model of $\exists\tau\mathrm{SM}(\Pi_s)$, and by the proof of Lemma 4, $\mathbf{B}$ is also a model of $\exists\sigma\mathrm{SM}(\Pi_0)$. Hence, $\mathbf{A}$ is a model of ψ.

"$\Longleftarrow$": Let $\mathbf{A}$ be a finite model of ψ. Then there is an υ-expansion $\mathbf{B}$ of $\mathbf{A}$ such that $\mathbf{B}$ satisfies both $\exists\tau\mathrm{SM}(\Pi_s)$ and $\exists\sigma\mathrm{SM}(\Pi_0)$. By Lemma 5, $\mathbf{B}$ is a successor structure, and then by Lemma 4, $\mathbf{B}$ must be a model of φ. This means that $\mathbf{A}$ is a model of φ. $\Box$

Fix υ_n as the vocabulary $\{P_n\}$, where P_n is an n-ary predicate constant. Let Parity^n denote the class of finite υ_n-structures in each of which P_n is interpreted as a set consisting of an even number of n-tuples. The following result was proved by Ajtai.

THEOREM 3 (IMPLICIT BY THEOREM 2.1 IN [AJTAI 1983]). *Let n be a positive integer. Then Parity^n is not definable in $\Sigma^1_{1,n-1}$ over finite structures.*

With all of the above results, we are now in the position to establish a weaker separation:

THEOREM 4. $\mathrm{DLP}_n \not\leq_{\mathsf{FIN}} \mathrm{NLP}^{\mathsf{F}}_{2n-1}$ *for all $n > 1$.*

PROOF. Fix $n > 1$. Now, let us show that Parity^{2n} is definable in DLP_n over finite structures. We first show that Parity^{2n} is definable in $\Sigma^1_{2,n}[\forall^n\exists^*]$ over successor structures. Let $First, Last$ and $Succ_i$ be defined the same on term tuples as those in the proof of Lemma 4, and let $Succ(\vec{s},\vec{t})$ denote the disjunction of $Succ_i(\vec{s},\vec{t})$ for all $i \in \{1,\dots,n\}$ if $\vec{s}$ and $\vec{t}$ are n-tuples. Let X, Y be predicate variables of arity n, and φ_1 be the formula

$$\begin{bmatrix} \forall\vec{z}(First(\vec{z}) \to (Y(\vec{z}) \leftrightarrow\ P_{2n}(\vec{x},\vec{z}))) \wedge \\ \forall\vec{y}\vec{z}(Succ(\vec{y},\vec{z}) \to (P_{2n}(\vec{x},\vec{z}) \leftrightarrow Y(\vec{y}) \oplus Y(\vec{z}))) \end{bmatrix} \to \exists\vec{z}(Last(\vec{z}) \wedge (X(\vec{x}) \leftrightarrow Y(\vec{z}))), \tag{106}$$

where $\psi \oplus \chi$ denotes the formula $(\psi \leftrightarrow \neg\chi)$ if ψ and χ are formulas. Intuitively, φ_1 asserts that $X(\vec{a})$ is true iff the cardinality of $\{\vec{b} \mid (\vec{a},\vec{b}) \in P_{2n}\}$ is odd. Let φ_2 denote the formula

$$\begin{bmatrix} \forall\vec{z}(First(\vec{z}) \to (X(\vec{z}) \leftrightarrow\ Y(\vec{z}))) \wedge \\ \forall\vec{y}\vec{z}(Succ(\vec{y},\vec{z}) \to (X(\vec{z}) \leftrightarrow Y(\vec{y}) \oplus Y(\vec{z}))) \end{bmatrix} \to \exists\vec{z}(Last(\vec{z}) \wedge \neg Y(\vec{z})). \tag{107}$$

Intuitively, φ_2 states that X consists of an even number of n-tuples on the intended domain. Now, let $\varphi = \exists X\forall Y\forall\vec{x}(\varphi_1 \wedge \varphi_2)$. Obviously, φ can be rewritten as an equivalent sentence in $\Sigma^1_{2,n}[\forall^n\exists^*]$. By a routine check, it is not hard to see that, for every successor structure $\mathbf{A}$ of $\upsilon(\varphi)$, the restriction of $\mathbf{A}$ to υ_{2n} belongs to Parity^{2n} iff $\mathbf{A}$ is a model of φ.

According to the proof of Lemma 4, there exists a disjunctive program Π_0 and a finite set τ of predicates of arities $\leq n$ such that $\exists\tau\mathrm{SM}(\Pi_0)$ is equivalent to φ over successor structures and no predicate in υ_s is intensional w.r.t. Π_0. Let Π be the union of Π_s and Π_0. Let σ be the set of predicates in $\upsilon(\Pi) \setminus \upsilon_{2n}$. It is easy to show that, over finite structures, Parity^{2n} is defined by $\exists\sigma\mathrm{SM}(\Pi)$, so it is definable in DLP_n.

Next, we show that Parity^{2n} is not definable in $\mathrm{NLP}^{\mathsf{F}}_{2n-1}$ over finite structures, which yields the desired theorem immediately. By Proposition 6, it sufficies to prove that Parity^{2n}

is not definable in $\Sigma^{1\mathrm{F}}_{1,2n-1}$ over finite structures. Towards a contradiction, assume it is not true. By a proof similar to Theorem 3.9 (a) in [Durand et al. 1998], it is easy to prove:

Claim. Let $m \geq 1$ and suppose the property Parity^m is definable in $\Sigma^{1\mathrm{F}}_{1,m-1}$ over finite structures. Then the property Parity^{2m} is definable in $\Sigma^{1\mathrm{F}}_{1,2m-2}$ over finite structures.

With the above claim, we can now conclude that Parity^{4n} is definable in $\Sigma^{1\mathrm{F}}_{1,4n-2}$ over finite structures. Since every function variable of arity $k \geq 0$ can always be simulated by a predicate variable of arity $k+1$, Parity^{4n} should be definable in $\Sigma^{1}_{1,4n-1}$ over finite structures, which contradicts with Theorem 3. This completes the proof. $\square$

5. ARBITRARY STRUCTURES

Based on the results presented in the previous two sections, we can then compare the expressiveness of disjunctive programs and normal programs over arbitrary structures.

THEOREM 5. DLP $\simeq$ NLP *iff* DLP $\simeq_{\mathsf{FIN}}$ NLP.

PROOF. The direction "$\Longrightarrow$" is trivial. Now let us consider the converse. Assume that DLP $\simeq_{\mathsf{FIN}}$ NLP, and let Π be a disjunctive program. Then there exists a normal program $\Pi^\star$ such that $\mathrm{SM}(\Pi) \equiv_{\mathsf{FIN}} \exists\sigma\mathrm{SM}(\Pi^\star)$, where σ is the set of predicates that occur in $\Pi^\star$ but not in Π. By Theorem 2, there is a normal program $\Pi^\diamond$ such that $\mathrm{SM}(\Pi) \equiv_{\mathsf{INF}} \exists\tau\mathrm{SM}(\Pi^\diamond)$. Without loss of generality, we assume that $\sigma \cap \tau = \emptyset$. To show DLP $\simeq$ NLP, our idea is to design a normal program testing whether the intended structure is finite. We let $\Pi^\star$ work if it is true, and let $\Pi^\diamond$ work otherwise. To do this, we introduce a new predicate $Finite$ of arity 0, and let Π_{T} be the union of Π_{S} (see Section 4) and the set of rules as follows:

$$First(x) \;\rightarrow\; Num(x), \tag{108}$$
$$Num(x) \wedge Succ(x,y) \;\rightarrow\; Num(y), \tag{109}$$
$$Num(x) \wedge Last(x) \;\rightarrow\; Finite. \tag{110}$$

Let $\pi = \upsilon(\Pi_{\mathrm{T}}) \setminus \{Finite\}$. Now let us present a property as follows.

Claim. Suppose $\mathbf{A} \models \exists\pi\mathrm{SM}(\Pi_{\mathrm{T}})$. Then $\mathbf{A}$ is finite iff $\mathbf{A} \models Finite$.

The direction "$\Longrightarrow$" of this claim follows from Lemma 5. So, it remains to show the converse. Suppose $\mathbf{A}$ satisfies $Finite$. Let υ_0 denote the union of $\upsilon(\Pi_{\mathrm{T}})$ and the vocabulary of $\mathbf{A}$. Then, there exists an υ_0-expansion $\mathbf{B}$ of $\mathbf{A}$ such that $\mathbf{B}$ is a stable model of Π_{T}. Hence, $Less^{\mathbf{B}}$ should be a strict linear order on A; the element in $First^{\mathbf{B}}$ (respectively, $Last^{\mathbf{B}}$), if it exists, should be the least (respectively, largest) element in A w.r.t. $Less^{\mathbf{B}}$; and $Succ^{\mathbf{B}}$ should be the relation defining the direct successors w.r.t. $Less^{\mathbf{B}}$. As $Finite$ is true in $\mathbf{A}$, there must exist an integer $n \geq 0$ and n elements $a_1, \ldots, a_n$ from A such that $First(a_1), Last(a_n)$ and each $Succ(a_i, a_{i+1})$ are true in $\mathbf{B}$. We assert that, for each $a \in A$ there exists some index $i \in \{1, \ldots, n\}$ such that $a = a_i$. If not, let b be one of such elements. As $Less^{\mathbf{B}}$ is a strict linear order, there exists $j \in \{1, \ldots, n-1\}$ such that both $Less(a_j, b)$ and $Less(b, a_{j+1})$ are true in $\mathbf{B}$. But this is impossible since $Succ(a_j, a_{j+1})$ is true in $\mathbf{B}$. So, we have $A = \{a_1, \ldots, a_n\}$. This implies that $\mathbf{A}$ is finite as desired.

Next, let us construct the desired program. Let $\Pi^\star_0$ (respectively, $\Pi^\diamond_0$) denote the program obtained from $\Pi^\star$ (respectively, $\Pi^\diamond$) by adding $Finite$ (respectively, $\neg Finite$) to the body of each rule as a conjunct. Let $\Pi^\dagger = \Pi^\star_0 \cup \Pi^\diamond_0 \cup \Pi_{\mathrm{T}}$. Let $\nu = \upsilon(\Pi^\dagger) \setminus \upsilon(\Pi)$. Now, we show that $\exists\nu\mathrm{SM}(\Pi^\dagger)$ is equivalent to $\mathrm{SM}(\Pi)$ over arbitrary structures. By definition and Proposition 2, it suffices to show that $\mathrm{SM}(\Pi)$ is equivalent to the formula

$$\exists Finite(\exists\sigma\mathrm{SM}(\Pi_0^\star) \wedge \exists\tau\mathrm{SM}(\Pi_0^\diamond) \wedge \exists\pi\mathrm{SM}(\Pi_{\mathrm{T}})). \tag{111}$$

Let $\mathbf{A}$ be a structure of $\upsilon(\Pi)$. As a strict total order always exists on domain A, we can construct an $\upsilon(\Pi) \cup \upsilon(\Pi_{\mathrm{T}})$-expansion $\mathbf{B}$ of $\mathbf{A}$ such that $\mathbf{B}$ is a stable model of Π_{T}. By the above claim, $\mathbf{B} \models Finite$ if, and only if, $\mathbf{A}$ is finite. Let us first assume that $\mathbf{A}$ is finite. By definition, it is clear that $\exists\sigma\mathrm{SM}(\Pi_0^\star)$ is satisfied by $\mathbf{B}$ if, and only if, $\exists\sigma\mathrm{SM}(\Pi^\star)$ is satisfied by $\mathbf{A}$, and $\exists\sigma\mathrm{SM}(\Pi_0^\diamond)$ is always true in $\mathbf{B}$. This means that $\exists\nu\mathrm{SM}(\Pi^\dagger)$ is equivalent to $\mathrm{SM}(\Pi)$ over finite structures. By a symmetrical argument, we can show that the equivalence also holds over infinite structures. This then completes the proof. $\square$

REMARK 7. *In classical logic, it is well-known that separating languages over arbitrary structures is usually easier than that over finite structures [Ebbinghaus and Flum 1999]. In logic programming, it also seems that arbitrary structures are better-behaved than finite structures. For example, there are some preservation theorems that work on arbitrary structures, but not on finite structures [Ajtai and Gurevich 1994]. Therefore, an interesting question then arises as whether it is possible to develop some techniques on arbitrary structures so that stronger separations of* DLP *from* NLP *are achievable.*

COROLLARY 3. DLP $\simeq$ NLP *iff* $\mathrm{NP} = \mathrm{coNP}$.

Next, we give a characterization for disjunctive programs.

PROPOSITION 8. DLP $\simeq \Sigma_2^1[\forall^*\exists^*]$.

PROOF. The direction "$\leq$" trivially follows from the second-order definition of stable model semantics. So, we only need to show the converse. To do this, it suffices to prove that, for every second-order sentence φ that is of the following form:

$$\forall\sigma\forall\vec{x}\exists\vec{y}(\vartheta_1(\vec{x},\vec{y}) \vee \cdots \vee \vartheta_m(\vec{x},\vec{y})), \tag{112}$$

there is a disjunctive program Π and a set τ of auxiliary predicates such that φ is equivalent to $\exists\tau\mathrm{SM}(\Pi)$, where σ is a finite set of predicates; $\vec{x}$ and $\vec{y}$ are two finite tuples of individual variables; and each ϑ_i is a conjunction of atoms or negated atoms. Notice that, if φ is equivalent to $\exists\tau\mathrm{SM}(\Pi)$, then any $\Sigma_2^1[\forall^*\exists^*]$-sentence of the form $\exists\pi\varphi$ (where π is a set of predicates) is equivalent to $\exists\pi\exists\tau\mathrm{SM}(\Pi)$, which proves the desired proposition.

Now, let us prove the equivalence between φ and $\exists\tau\mathrm{SM}(\Pi)$. Let n denote the length of $\vec{x}$. Again by employing the saturation technique (see, e.g., the proof of Theorem 6.3 in [Eiter et al. 1997]), we can construct a logic program Π as follows:

$$T_X(\vec{x},\vec{z}) \vee F_X(\vec{x},\vec{z}) \qquad (X \in \sigma) \tag{113}$$

$$D(\vec{x}) \to F_X(\vec{x},\vec{z}) \qquad (X \in \sigma) \tag{114}$$

$$D(\vec{x}) \to T_X(\vec{x},\vec{z}) \qquad (X \in \sigma) \tag{115}$$

$$\tilde{\vartheta}_i(\vec{x},\vec{y}) \to D(\vec{x}) \qquad (1 \leq i \leq m) \tag{116}$$

$$\neg D(\vec{x}) \to \bot \tag{117}$$

where, for each $X \in \sigma$, T_X and F_X are two distinct fresh predicates of arity $(n+k)$ if k denotes the arity of X; each $\tilde{\vartheta}_i$ is the formula obtained from ϑ_i by substituting $F_X(\vec{x},\vec{t})$

for $\neg X(\vec{t})$ and followed by substituting $T_X(\vec{x}, \vec{t})$ for $X(\vec{t})$ whenever $X \in \sigma$ and $\vec{t}$ is a tuple of terms of a proper length; and D is a fresh predicate of arity n. Let τ denote the set

$$\sigma \cup \{D\} \cup \bigcup_{X \in \sigma} \{T_X, F_X\}. \tag{118}$$

Clearly, $\exists\tau\mathrm{SM}(\Pi)$ belongs to DLP. It remains to show that $\exists\tau\mathrm{SM}(\Pi)$ is equivalent to φ.

Let $\mathbf{A}$ be a model of φ, and take α as an arbitrary assignment in $\mathbf{A}$. Now our task is to show that $\mathbf{A}$ is a model of $\exists\tau\mathrm{SM}(\Pi)$. Let $\mathbf{B}$ be an $\upsilon(\Pi)$-expansion of $\mathbf{A}$ that interpretes each predicate among $D, T_X, F_X (X \in \sigma)$ as the full relation of a proper arity. To obtain the desired conclusion, it suffices to prove that $\mathbf{B}$ is a stable model of Π. It is easy to check that $\mathbf{B}$ is a model of $\widehat{\Pi}$. Towards a contradiction, let us assume that $\mathbf{B}$ is not a stable model of Π, which implies that there exists an assignment β in $\mathbf{B}$ such that

$$\mathbf{B} \models (\pi^* < \pi)[\beta] \qquad \& \qquad \mathbf{B} \models \widehat{\Pi}^*[\beta], \tag{119}$$

where denotes the set $\tau \setminus \sigma$, and both notations π^* and $\widehat{\Pi}^*$ are defined in the same way as those in Section 2. From these conclusions, we know that there is some n-tuple on A, say $\vec{a}$, such that $\vec{a} \notin \beta(D)^*$. From the conclusion that $\mathbf{B} \models \widehat{\Pi}^*[\beta]$, we can infer that

$$\mathbf{B} \models \bigwedge_{i=1}^{m} \neg\exists\vec{y}\tilde{\vartheta}_i^*(\vec{a}, \vec{y})[\beta], \tag{120}$$

where $\tilde{\vartheta}_i^*$ denotes the formula obtained from $\tilde{\vartheta}_i$ by substituting P^* for P whenever

$$P \in \bigcup_{X \in \sigma} \{T_X, F_X\}. \tag{121}$$

Let α be an assignment in $\mathbf{A}$ such that $\alpha(X) = \beta(T_X^*)$ for all predicates $X \in \sigma$. According to the definition of $\tilde{\vartheta}$, it is not difficult to see that

$$\mathbf{A} \models \bigwedge_{i=1}^{m} \neg\exists\vec{y}\vartheta_i(\vec{a}, \vec{y})[\alpha], \tag{122}$$

which implies that $\mathbf{A}$ is not a model of φ, a contradiction as desired.

For the converse, let us assume that $\mathbf{A}$ is a model of $\exists\pi\mathrm{SM}(\Pi)$. Towards a contradiction, we also assume that $\mathbf{A}$ is not a model of φ. By the former, we know that there is a stable model $\mathbf{B}$ of Π that is an $\upsilon(\Pi)$-expansion of $\mathbf{A}$. By the definition of Π, it is easy to see that $\mathbf{B}$ interprets each predicate among D, T_X, F_X $(X \in \sigma)$ as the full relation on A of a proper arity. By the assumption that $\mathbf{A} \models \neg\varphi$, there is an assignment α in $\mathbf{A}$ such that

$$\mathbf{A} \models \neg\exists\vec{y}(\vartheta_1(\vec{x}, \vec{y}) \vee \cdots \vee \vartheta_m(\vec{x}, \vec{y}))[\alpha]. \tag{123}$$

With this, we can infer that there exists an index $i \in \{1, \ldots, m\}$ such that

$$\mathbf{A} \models \neg\exists\vec{y}\vartheta_i(\vec{x}, \vec{y})[\alpha]. \tag{124}$$

Let $\vec{a} = \alpha(\vec{x})$, and let β be an assignment in $\mathbf{B}$ such that $\beta(D^*) = A^n \setminus \{\vec{a}\}$ and that

$$\beta(T_X^*) \ = \ \{(\vec{a}, \vec{b}) \mid \vec{b} \in \alpha(X)\} \cup \{(\vec{a}_0, \vec{b}) \mid \vec{a}_0 \in A^n \setminus \{\vec{a}\} \,\&\, \vec{b} \in A^k\}, \tag{125}$$

$$\beta(F_X^*) \ = \ \{(\vec{a}, \vec{b}) \mid \vec{b} \in A^k \setminus \alpha(X)\} \cup \{(\vec{a}_0, \vec{b}) \mid \vec{a}_0 \in A^n \setminus \{\vec{a}\} \,\&\, \vec{b} \in A^k\} \tag{126}$$

for all predicates $X \in \sigma$ of arity k. Then, it is a routine task to check that

$$\mathbf{B} \models (\pi^* < \pi)[\beta] \qquad \& \qquad \mathbf{B} \models \widehat{\Pi}^*[\beta], \tag{127}$$

where π denotes the set of predicates D and T_X, F_X for all $X \in \sigma$, and both notations π^* and $\widehat{\Pi}^*$ are defined in the same way as those in Section 2. From this, we then conclude that $\mathbf{B}$ is not a model of $\mathrm{SM}(\Pi)$, i.e., a stable model of Π, a contradiction as desired. $\square$

6. CONCLUSION AND RELATED WORK

We have undertaken a comprehensive study on the expressiveness of logic programs under the general stable model semantics. From the results we proved in this paper and other existing results, now we can draw an almost complete picture for the expressiveness of logic programs and some related fragments of second-order logic. As shown in Figure 1, the expressiveness hierarchy in each table is related to a structure class, while the syntactical classes in a same block is proved to be of the same expressiveness over the corresponding structure class. The closer a block is to the top, the more expressive the classes in the block are. In addition, a dashed line between two blocks indicates that the corresponding separation is true if, and only if, the complexity class NP is not closed under complement.

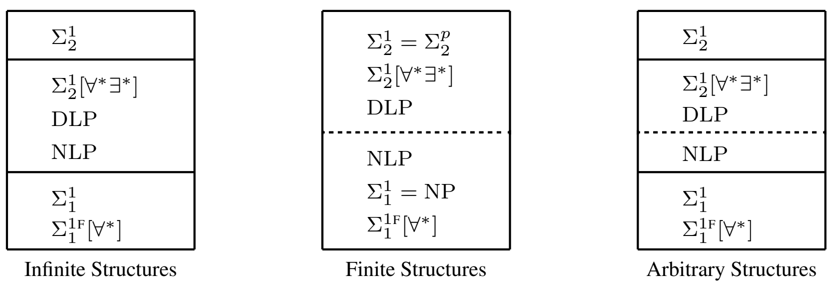

Fig. 1. Expressiveness Hierarchies Related to Logic Programs

Without involving the well-known conjecture in Complexity Theory, we established the intranslatability from disjunctive programs to normal programs over finite structures if the arities of auxiliary predicate and function symbols are bounded in a certain sense. This can be regarded as a strong evidence that disjunctive programs are more expressive than normal programs over finite structures. As a byproduct, we also developed a succinct translation from normal logic programs to first-order sentences. This may be viewed as an alternative to the work of the ordered completion for translating a normal program into a first-order sentence over finite structures proposed in [Asuncion et al. 2012]. It is also worth noting that both the number and the maximum arity of auxiliary symbols involving in our translation are significantly smaller than those in the ordered completion.

There are also several previous works that contribute to Figure 1, which are listed as follows. The translatability from Σ_1^1 to $\Sigma_1^{1\mathrm{F}}[\forall^*]$ follows from the well-known existence of Skolem's normal form. The translatability from Σ_2^1 to $\Sigma_2^1[\forall^*\exists^*]$ over finite structures is due to [Leivant 1989]. The separation of Σ_2^1 from $\Sigma_2^1[\forall^*\exists^*]$ (on both arbitrary and infinite structures) is implicit in [Eiter et al. 1996]. The translatability from DLP to $\Sigma_2^1[\forall^*\exists^*]$ over arbitrary structures (so also over infinite structures and over finite structures) directly follows from the second-order transformation [Ferraris et al. 2011]. The intranslatability from NLP to Σ_1^1 over arbitrary structures is due to [Asuncion et al. 2012].

Over Herbrand structures, [Schlipf 1995; Eiter and Gottlob 1997] proved that normal programs, disjunctive programs and universal second-order logic are of the same expressiveness under the query equivalence. Their proofs employ an approach from Computability Theory. However, this approach seems difficult to be applied to general infinite structures. In the propositional case, there have been a lot of works on the translatability and expressiveness of logic programs, e.g., [Eiter et al. 2004; Janhunen 2006]. It should be noted that the picture of expressiveness and translatability in there is quite different from that in the first-order case. For example, it is not difficult to show that every boolean function can be defined by a normal program if auxiliary propositional symbols are allowed, and thus developing a translation from propositional disjunctive programs to propositional normal programs is always possible if we do not consider the succinctness. There have been also some works (see, e.g., [Eiter et al. 2013]) that focus on translatability between classes of propositional logic programs under the strong equivalence and uniform equivalence. As a future work, it would be interesting to generalize our work to these equivalence.

Translations from logic programs into first-order theories had been also investigated in several earlier works. Chen et al. [2006] proved that, over finite structures, every normal program with variables can be equivalently translated to a possibly infinite first-order theory. This result was later extended to disjunctive programs [Lee and Meng 2011]. In addition, Lee and Meng [2011] identified the intranslatability from logic programs into possibly infinite first-order theories over arbitrary structures, and proposed some sufficient conditions which assure the translatability over arbitrary structures. Instead of using possibly infinite theories, translations in this paper only involve finite theories. The translatability between logic programs and first-order theories was also considered in [Zhang et al. 2011], but first-order theories there are equipped with the general stable model semantics.

It is also worth mentioning some related works that focus on identifying sufficient conditions for the translatability between first-order logic programs and first-order theories. A possibly incomplete list is as follows. Ferraris et al. [2011] showed that every tight logic program is equivalent to a first-order sentence. Chen et al. [2011] proposed a notion called loop-separability that assures the first-order definability over finite structures. Zhang and Zhang [2013b] established some semantic characterizations for the first-order expressibility of negation-free disjunctive programs over arbitrary structures. Ben-Eliyahu and Dechter [1994] proved that every head-cycle-free disjunctive program is equivalent to a normal program. Very recently, Zhou [2015] proposed a semantic notion called choice-boundedness that assures the translatability from disjunctive programs to normal programs. All these results provide another view to understand the expressiveness of logic programs.

Acknowledgements

The authors are grateful to several referees for their useful comments. The work of the first author was supported in part by the innovative research funds of Huazhong University of Science and Technology under grant 2016YXMS073 and the National Natural Science Foundation of China under grants U1401258 and 61572221. The work of the second author was supported in part by the research grant from the State Key Laboratory of Computer Science, Institute of Software, Chinese Adademy of Sciences, under the project SYSKF1506.

REFERENCES

AJTAI, M. 1983. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic 24*, 1–48.

AJTAI, M. AND GUREVICH, Y. 1994. Datalog vs first-order logic. *Journal of Computer and System Sciences 49*, 562–588.

ASUNCION, V., LIN, F., ZHANG, Y., AND ZHOU, Y. 2012. Ordered completion for first-order logic programs on finite structures. *Artificial Intelligence 177–179*, 1–24.

BEN-ELIYAHU, R. AND DECHTER, R. 1994. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence 12,* 1–2, 53–87.

CHEN, Y., LIN, F., WANG, Y., AND ZHANG, M. 2006. First-order loop formulas for normal logic programs. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning*. 298–307.

CHEN, Y., LIN, F., ZHANG, Y., AND ZHOU, Y. 2011. Loop-separable programs and their first-order definability. *Artificial Intelligence 175*, 890–913.

DANTSIN, E., EITER, T., GOTTLOB, G., AND VORONKOV, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys 33,* 3, 374–425.

DURAND, A., GRANDJEAN, E., AND OLIVE, F. 2004. New results on arity vs. number of variables. *Research report 20–2004, LIF, Marseille, France*.

DURAND, A., LAUTEMANN, C., AND SCHWENTICK, T. 1998. Subclasses of binary NP. *Journal of Logic and Computation 8,* 2, 189–207.

EBBINGHAUS, H.-D. AND FLUM, J. 1999. *Finite Model Theory*, 2 ed. Springer-Verlag, New York.

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

EITER, T., FINK, M., TOMPITS, H., AND WOLTRAN, S. 2004. On eliminating disjunctions in stable logic programming. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning*. 447–458.

EITER, T. AND GOTTLOB, G. 1997. Expressiveness of stable model semantics for disjunctive logic programs with functions. *The Journal of Logic Programming 33*, 167–178.

EITER, T., GOTTLOB, G., AND GUREVICH, Y. 1996. Normal forms for second-order logic over finite structures, and classication of NP optimization problems. *Annals of Pure and Applied Logic 78*, 111–125.

EITER, T., GOTTLOB, G., AND MANNILA, H. 1997. Disjunctive datalog. *ACM Transactions on Database Systems 22*, 364–418.

FAGIN, R. 1974. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation*. SIAM-AMS Proceedings, vol. 7. 43–73.

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

FERRARIS, P., LEE, J., LIFSCHITZ, V., AND PALLA, R. 2009. Symmetric splitting in the general theory of stable models. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. 797–803.

GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium on Logic Programming*. 1070–1080.

GRANDJEAN, E. 1985. Universal quantifiers and time complexity of random access machines. *Mathematical Systems Theory 18,* 2, 171–187.

IMMERMAN, N. 1999. *Descriptive complexity*. Graduate texts in computer science. Springer.

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

LEE, J. AND MENG, Y. 2011. First-order stable model semantics and first-order loop formulas. *Journal of Artificial Intelligence Research 42*, 125–180.

LEIVANT, D. 1989. Descriptive characterizations of computational complexity. *Journal of Computer and System Sciences 39*, 51–83.

LIERLER, Y. AND MARATEA, M. 2004. Cmodels-2: SAT-based answer set solver enhanced to non-tight programs. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning*. 346–350.

LIN, F. AND ZHAO, Y. 2004. ASSAT: computing answer sets of a logic program by sat solvers. *Artificial Intelligence 157,* 1-2, 115–137.

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

LOBO, J., MINKER, J., AND RAJASEKAR, A. 1992. *Foundations of Disjunctive Logic Programming*. The MIT Press, Cambridge.

PEARCE, D. AND VALVERDE, A. 2005. A first order nonmonotonic extension of constructive logic. *Studia Logica 80,* 2/3, 321–346.

SCHLIPF, J. S. 1995. The expressive powers of the logic programming semantics. *Journal of Computer and System Sciences 51,* 1, 64–86.

STOCKMEYER, L. J. 1977. The polynomial-time hierarchy. *Theoretical Computer Science 3*, 1–22.

ZHANG, H. AND ZHANG, Y. 2013a. Disjunctive logic programs versus normal logic programs. *CoRR abs/1304.0620 (available at http://arXiv.org/abs/1304.0620).*

ZHANG, H. AND ZHANG, Y. 2013b. First-order expressibility and boundedness of disjunctive logic programs. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. 1198–1204.

ZHANG, H., ZHANG, Y., YING, M., AND ZHOU, Y. 2011. Translating first-order theories into logic programs. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. 1126–1131.

ZHOU, Y. 2015. First-order disjunctive logic programming vs normal logic programming. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. 3292–3298.