Disjunctive Logic Programs versus Normal Logic Programs

Heng Zhang and Yan Zhang
Artificial Intelligence Research Group
School of Computing, Engineering and Mathematics
University of Western Sydney, Australia

Abstract
This paper focuses on the expressive power of disjunctive and normal logic programs under the stable model semantics. A translation from disjunctive logic programs into normal logic programs is proposed and then proved to be sound over infinite structures. The equivalence of expressive power of two kinds of logic programs over arbitrary structures is shown to coincide with that over finite structures, and coincide with whether or not NP is closed under complement. Over finite structures, the intranslatability from disjunctive logic programs to normal logic programs is also proved if arities of auxiliary predicates and functions are bounded in a certain way.

1 Introduction
Normal logic programs provide us an elegant and efficient language for knowledge representation, which incorporates the abilities of classical logic, mathematical induction and nonmonotonic reasoning. Disjunctive logic programs extend this language by introducing epistemic disjunction to the rule head, motivated to represent more knowledge, in particular, indefinite knowledge. The most popular semantics for them is the stable model semantics, which was originally proposed by [Gelfond and Lifschitz, 1988]. Logic programming based on it is known as answer set programming, a flourishing paradigm of declarative programming emerged recently.

Identifying the expressive power of languages is one of the central topics in the area of knowledge representation and reasoning. In this paper we try to compare the expressive power of these two kinds of logic programs under the stable model semantics. The expressive power of logic programs as a database query language has been thoroughly studied in last three decades. For a survey, please refer to [Dantsin et al., 2001]. However, except for a few work, the results for normal and disjunctive logic programs are limited to Herbrand structures. As encoding knowledge in Herbrand domains is unnatural and inflexible in many cases, our work will focus on infinite structures, finite structures and arbitrary structures. The semantics employed here is the general stable model semantics, which was developed by [Ferraris et al., 2011; Lin and Zhou, 2011] via a second-order translation, and provides us a unified framework for answer set programming.

Our contributions in this paper are as follows. Firstly, we show that, over infinite structures, every disjunctive logic program can be equivalently translated to a normal logic program. Secondly, we prove that disjunctive and normal logic programs are of the same expressive power over arbitrary structures if and only if they are of the same expressive power over finite structures, and if and only if complexity class NP is closed under complement. Thirdly, we show that for each integer $k > 1$ there is a disjunctive logic program with intentional predicates of arities $\leq k$ that can not be equivalently translated to any normal program with auxiliary predicates and functions of arities $< 2k$. To prove them, the relationship between logic programs and classical logic is also studied.

2 Preliminaries

Vocabularies are assumed to be sets of predicate constants and function constants. Every constant is equipped with a natural number, its arity. Nullary function constants are also called individual constants, and nullary predicate constants are called proposition constants. For some technical reasons, a vocabulary is allowed to contain an arbitrary infinite set of proposition constants. Logical symbols are defined as usual, including a countable set of predicate variables, a countable set of function variables and a countable set of individual variables. Predicate (function) constants and variables are simply called predicates (functions) if no confusion occurs. Terms, formulae and sentences of a vocabulary $\nu$ (or shortly, $\nu$-terms, $\nu$-formulae and $\nu$-sentences) are built from $\nu$, equality, variables, connectives and quantifiers in a standard way. For each formula $\varphi$ and each set $\Sigma$ of formulae, let $\nu(\varphi)$ and $\nu(\Sigma)$ be the sets of all constants occurring in $\varphi$ and $\Sigma$ respectively. Let $Q^\varphi$ and $Q^\Sigma$ denote quantifier blocks $QX_1\cdots QX_n$ and $Qx_1\cdots Qx_m$ respectively if $\tau$ is the set of $X_i$ for all integers $1 \leq i \leq n$, $\bar{x} = x_1\cdots x_m$, $Q$ is $\forall$ or $\exists$, $X_i$ and $x_i$ are predicate/function and individual variables respectively.

Every structure $\mathcal{A}$ of a vocabulary $\nu$ (or shortly, $\nu$-structure $\mathcal{A}$) is accompanied by a nonempty set $\mathcal{A}$, the domain of $\mathcal{A}$, and interprets each $n$-ary predicate constant $P$ in $\nu$ as an $n$-ary relation $P^\mathcal{A}$ on $\mathcal{A}$, and interprets each $n$-ary function constant $f$ in $\nu$ as an $n$-ary function $f^\mathcal{A}$ on $\mathcal{A}$. A structure is finite if its domain is finite; otherwise it is infinite. A restriction of a structure $\mathcal{A}$ to a vocabulary $\sigma$ is the structure obtained from
A by discarding all interpretations for constants which do not belong to σ. Furthermore, given a vocabulary v, a structure A is called an 𝜈-expansion of some σ-structure B if σ ⊆ v, the vocabulary of A is v, and B is a restriction of A to σ.

Every assignment in a structure A is a function that maps each individual variable to an element of A and that maps each predicate variable to a relation on A of the same arity. Given a (second-order) formula ϕ and an assignment α in A, we write (A, α) |= ϕ if α satisfies ϕ in A in the standard way. In particular, if ϕ is a sentence, we simply write A |= ϕ, and say A is a model of ϕ, or in other words, A satisfies ϕ. Given two (second-order) formulae ϕ and ψ and a class C of structures, we say ϕ is equivalent to ψ over C, or write ϕ ≡C ψ for short, if for every structure A in C and every assignment α in A, α satisfies ϕ in A if and only if α satisfies ψ in A.

Suppose τ is a set of predicates and A is a domain, i.e., a nonempty set. Let Gτ(A, A) denote the set of P(α) for all predicates P ∈ τ and all n-tuples a on A where n is the arity of P. Let GPC(τ, A) be the set of finite disjunctions built from atoms in Gτ(A, A). Each element in Gτ(A, A) is called a grounded atom (grounded positive clause) of τ over A. Given a structure A, let INS(A, τ) be the set of grounded atoms P(α) such that P ∈ τ and P(α) is true in A.

Let FIN denote the class of all finite structures, and let INF denote the class of all infinite structures. Suppose Σ and Π are two sets of second-order formulae and let C be a class of structures. We write Σ ⊆ C or Π ⊆ C if for each formula φ ∈ Σ, there is a formula ψ in C such that φ |=C ψ. We write Σ ≃ C or Π ≃ C if both Σ ⊆ C and Π ⊆ C hold. In particular, if C is the class of all arbitrary structures, the subscripts Σ or Π can be dropped.

### 2.1 Logic Programs

Every disjunctive logic program is a set of rules of the form

\[\zeta_1 \land \cdots \land \zeta_m \rightarrow \zeta_{m+1} \lor \cdots \lor \zeta_n\]

where \(0 \leq m \leq n\), and for each integer \(m < i \leq n\), \(\zeta_i\) is an atom without equality; for each integer \(1 \leq j \leq m\), \(\zeta_j\) is a literal, i.e., an atom or its negation. The disjunctive part of the rule is called its head, and the conjunctive part called its body. Let Π be a disjunctive logic program. Then each intensional predicate of Π is a predicate constant that occurs in the head of some rule in Π. Atoms built from intensional predicates of Π are called intensional atoms of Π.

Let Π be a disjunctive logic program. Then Π is normal if the head of each rule contains at most one atom, Π is plain if there is no negation of any intensional atom of Π occurring in any of its rules, Π is propositional if no predicate of positive arity occurs in any of its rules, and Π is finite if it contains only a finite set of rules. In particular, if we do not mention, a logic program is always assumed to be finite.

Given a disjunctive logic program Π, let SM(Π) denote the formula \(\varphi \land \forall^+ \tau^+(\tau^+ < \tau \rightarrow \neg \varphi^\tau)\), where \(\tau^+\) is the set of all intensional predicate constants of Π; \(\tau^+\) is the set of predicate variables \(P^*\) for all \(P \in \tau\); \(\tau^+ < \tau\) is the formula \(\land_{P \in \tau} \forall \bar{x}(P^*(\bar{x}) \rightarrow P(\bar{x})) \land \land_{P \in \tau} \forall \bar{x}(P(\bar{x}) \rightarrow P^*(\bar{x}))\); \(\varphi\) is the conjunction of all sentences \(\psi(\gamma)\) such that \(\gamma\) is a rule in Π and \(\forall(\gamma)\) is the universal closure of \(\gamma\); \(\varphi^\tau\) is the conjunction of \(\forall(\gamma^\tau)\) such that \(\gamma^\tau\) is a rule in Π and \(\gamma^\tau\) is the rule obtained from \(\gamma\) by substituting \(P^*(\bar{l})\) for all positive occurrences of \(P(\bar{l})\) in its head or in its body if \(P\) in \(\tau\). A structure A is a stable model of Π if it is a model of SM(Π).

Now, given a class C of structures, or in other words, a property, we can define it by a logic program in the following way: the models of second-order formula \(\exists r\)SM(Π) are exactly the structures in C, where \(\tau\) is a set of predicate and function constants occurring in Π. Constants in \(\tau\) are called auxiliary constants. Given \(n \geq 0\), let DLP\(^n\) (DLP\(^n\)) be the set of formulae \(\exists r\)SM(Π) for all disjunctive logic programs Π and all finite sets \(\tau\) of predicate (predicate and function, respectively) constants of arities \(\leq n\). Let DLP (DLP\(_\tau\)) be the union of DLP\(^n\) (DLP\(^n\)) respectively for all \(n \geq 0\). In above definitions, if \(\tau\) is restricted to be normal, we then obtain the notations NLP\(^n\), NLP\(_\tau\), NLP and NL\(_\tau\) respectively.

Given a rule γ, a structure A and an assignment α in A, let \(\gamma[α]\) be the rule obtained from γ by substituting P(α) for all atoms P(\(\bar{a}\)) where \(\bar{a} = α(\bar{l})\), let \(\gamma[α]\) be the set of all conjuncts in the body of γ in which no intensional predicate positively occurs, and let \(\gamma[α]\) be the rule obtained from γ by removing all literals in γ[α]. Given a disjunctive logic program Π, let SM(Π) be the set of rules \(\gamma[α]\) for all assignments α in A and all rules γ in Π such that α satisfies γ[α] in A. The following proposition shows that the general stable model semantics can be redefined by the above first order GL-reduction:

### Proposition 1

(Zhang and Zhang, 2013, Proposition 4), Let Π be a disjunctive logic program with a set \(\tau\) of intensional predicates. Then an \(\forall(\Pi)\)-structure A is a stable model of Π iff INS(A, τ) is a minimal model (via set inclusion) model of Π[\(\forall(\Pi)\]).

### 2.2 Progression Semantics

In this paper, every clause and the clauses obtained from it by laws of commutation, association and identity for \(\lor\) are regarded to be the same. Now we review a progression semantics proposed by [Zhang and Zhang, 2013], which generalizes the fixed point semantics of [Lobo et al., 1992] to arbitrary structures and to logic programs with default negation.

Suppose Π is a propositional, possibly infinite and plain disjunctive logic program, and Σ is a set of finite disjunctions of atoms in \(\forall(\Pi)\). We define \(Γ[Σ(Π)]\) as the set of all positive clauses \(H \lor C_1 \lor \cdots \lor C_n\) such that \(\forall n \geq 0\) and there are a rule \(p_1 \lor \cdots \lor p_k \rightarrow H\) in Π and a sequence of positive clauses \(C_1 \lor \cdots \lor C_n \lor p_k\) in Σ. It is easy to verify that \(Γ[Σ]\) is a monotonic function on the sets of positive clauses of \(\forall(\Pi)\).

Now, by the first-order GL-reduction defined above, a progressional operator for first-order logic programs is then defined. Let Π be a disjunctive logic program and let A be a structure of \(\forall(\Pi)\). We define \(Γ[Π[\overline{A}]\) as the operator \(Γ[Π[\overline{A}].\) Furthermore, define \(Γ[Π[\overline{A}]\) \(= \emptyset\) as the empty set, and define \(Γ[Π[\overline{A}]\) \(\uparrow n\) as the union of \(Γ[Π[\overline{A}]\) \(\uparrow n-1\) and \(Γ[Π[\overline{A}]\) \(\uparrow n-1\) for all integers \(n > 0\). Finally, let \(Γ[Π[\overline{A}]\) \(\uparrow ω\) be the union of \(Γ[Π[\overline{A}]\) \(\uparrow n\) for all integers \(n \geq 0\). The following proposition provides us a progression semantics for disjunctive logic programs:

### Proposition 2

(Zhang and Zhang, 2013, Theorem 1), Let Π be a disjunctive logic program, \(\tau\) the set of all intensional predicates of Π, and A a structure of \(\forall(\Pi)\). Then A is a stable model of Π iff INS(A, τ) is a minimal model of \(Γ[Π[\overline{A}]\) \(\uparrow ω\).

**Remark 1.** In above proposition, it is easy to see that, if Π is normal, A is a stable model of Π if and only if INS(A, τ) = \(Γ[Π[\overline{A}]\) \(\uparrow ω\).
3 Infinite Structures

In this section, we propose a translation that turns each disjunctive logic program to an equivalent normal logic program over infinite structures. The main idea is to encode grounded positive clauses by elements in the intended domain. With the encoding, we then simulate the progression of given disjunctive logic program by the progression of a normal program.

Firstly, we show how to encode each clause by an element. Let $A$ be an infinite set. Every encoding function $\gamma$ on $A$ is an injective function from $A \times A$ into $A$. Let $\epsilon : A \rightarrow A$ be an encoding function on $A$ and $c$ an element in $A$ such that $\epsilon(a, b) \neq c$ for all elements $a, b \in A$. For the sake of convenience, we let $\epsilon(a_1, \ldots, a_k; c)$ denote the following expression

$$\epsilon(\cdots(\epsilon(c, a_1), a_2), \cdots, a_k)$$

for any integer $k \geq 0$ and any set of elements $a_1, a_2, \ldots, a_k \in A$. Let $\epsilon(A, c)$ denote the set $\{\epsilon(a; c) : a \in A^*\}$ where $A^*$ is the set of all finite tuples of elements in $A$. The merging function $\text{mrg}$ on $A$ related to $\epsilon$ and $c$ is the function from $\epsilon(A, c) \times \epsilon(A, c)$ to $\epsilon(A, c)$ that satisfies

$$\text{mrg}(\epsilon(a; c), \epsilon(b; c)) = \epsilon(a, b; c)$$

for all tuples $a, b \in A^*$. The extracting function $\text{ext}$ on $A$ related to $\epsilon$ and $c$ is the function from $\epsilon(A, c) \times A$ into $\epsilon(A, c)$ that satisfies $\text{ext}(\epsilon(a; c), b) = \epsilon(\epsilon(a; c), b)$, where $\epsilon(a)$ is the tuple obtained from $a$ by removing all occurrences of $b$.

It is clear that both the merging function and the extracting function are unique if $\epsilon$ and $c$ are fixed.

As mentioned before, the order of disjuncts in a clause does not change the semantics. To omit the order, we need some encoding predicates related to $\epsilon$ and $c$. The predicate $\text{in}$ is a subset of $\epsilon(A, c) \times A$ such that $\epsilon(a; c), b \in \text{in}$ iff $b$ occurs in $a$; the predicate $\text{subc}$ is a binary relation on $\epsilon(A, c)$ such that $(\epsilon(a, c), \epsilon(b, c)) \in \text{subc}$ iff all the elements in $a$ occur in $b$. The predicate $\text{eq}u$ is a binary relation on $\epsilon(A, c)$ such that $(a, b) \in \text{eq}u$ iff $(a, b) \in \text{subc}$ and $(b, c) \in \text{subc}$.

Example 1. Let $Z^+$ be the set of all positive integers, and define $e(m, n) = 2^m + 3^n$ for all integers $m, n \in Z^+$. Then $e$ is clearly an encoding function on $Z^+$, and integers $1, 2, 3, 4$ are not in the range of $e$. Suppose $P_1, P_2, P_3$ are predicates. Then a grounded atom $P_2(1, 3, 5)$ can be encoded as $e(1, 3, 5, 2)$, i.e. $e(e(2, 1, 3), 5)$ that equals to $2^{135} + 3^5$; the positive clause $P_2(1, 3, 5) \lor P_2(2, 1, 3)$ can be encoded as $e(e(1, 3, 5, 2), e(2, 3), e(2, 4, 1, 3)), \ldots$, where, for each $1 \leq i \leq 3$, integer $i$ is to be used for the ending flag of atoms $P_i(\cdots)$, and integer $4$ is for the ending flag of clauses.

With this method for encoding, we can then define a translation. Let $\Pi$ be a disjunctive logic program. We first construct a class of logic programs related to $\Pi$ as follows:

1. Let $C_1$ be the set consisting of an individual constant $eP$ for each predicate constant $P$ that occurs in $\Pi$, and of an individual constant $c$. Let $\Pi_1$ consist of the rule

$$\text{ENC}(x, y, c) \rightarrow \bot$$

for each individual constant $c \in C_1$, and the following rules:

$$\neg\text{ENC}(x, y, z) \rightarrow \text{ENC}(x, y, z) \quad (2)$$

$$\neg\text{ENC}(x, y, z) \rightarrow \text{ENC}(x, y, z) \quad (3)$$

$$\text{ENC}(x, y, z) \land \text{ENC}(u, v, z) \land \neg x = u \rightarrow \bot \quad (4)$$

$$\text{ENC}(x, y, z) \land \text{ENC}(u, v, z) \land \neg y = v \rightarrow \bot \quad (5)$$

$$\text{ENC}(x, y, z) \rightarrow \text{OK}_c(x, y, z) \quad (6)$$

$$\neg\text{OK}_c(x, y, z) \rightarrow \text{OK}_c(x, y, z) \quad (7)$$

$$\text{ENC}(x, y, z) \land \text{ENC}(x, y, u) \land \neg u = v \rightarrow \bot \quad (8)$$

2. Let $\Pi_2$ be the program consisting of the following rules:

$$y = c \rightarrow \text{MRG}(x, y, x) \quad (9)$$

$$\text{MRG}(x, u, v) \land \text{ENC}(u, w, v) \land \text{ENC}(v, w, z) \rightarrow \text{MRG}(x, y, z) \quad (10)$$

$$x = c \rightarrow \text{EXT}(x, y, x) \quad (11)$$

$$\text{EXT}(u, y, v) \land \text{ENC}(u, w, v) \land w = y \rightarrow \text{EXT}(x, y, v) \quad (12)$$

$$\text{EXT}(u, y, v) \land \text{ENC}(u, w, x) \land \neg w = y \land \text{ENC}(v, w, z) \rightarrow \text{EXT}(x, y, z) \quad (13)$$

$$\text{ENC}(x, y, u) \land \text{IN}(u, y) \quad (14)$$

$$\text{ENC}(x, y, v) \land \text{IN}(u, x) \land \text{IN}(u, y) \quad (15)$$

$$x = c \rightarrow \text{SUBC}(x, y) \quad (16)$$

$$\text{SUBC}(u, y) \land \text{ENC}(u, v, x) \land \text{ENC}(v, y, u) \rightarrow \text{SUBC}(x, y) \quad (17)$$

$$\text{SUBC}(x, y) \land \text{SUBC}(v, x) \rightarrow \text{EQU}(x, y) \quad (18)$$

3. Let $\Pi_3$ be the logic program consisting of the rule

$$\text{TRUE}(u) \land \text{EQU}(u, v) \rightarrow \text{TRUE}(v) \quad (19)$$

and the rule

$$\text{TRUE}(x_1) \land z_1 = [\theta_1] \land \text{IN}(z_1, x_1) \land \cdots$$

$$\land [\theta_k] \land \text{IN}(z_2, x_k) \land \text{EXT}(x_1, z_1, y_1) \land \cdots \land \text{EXT}(x_k, z_k, y_k)$$

$$\land \text{MRG}(y_1, \ldots, y_k, [\gamma]_n, v) \land v \rightarrow \text{TRUE}(v) \quad (20)$$

for each rule $\gamma$ in $\Pi$, where $\theta_1, \ldots, \theta_k$ list all the atoms that have strictly positive occurrences in the body of $\gamma$ for some integer $k \geq 0$; $\gamma_n$ is the head of $\gamma$; $\gamma_n$ is the conjunction of literals occurring in the body of $\gamma$ but not in $\theta_1, \ldots, \theta_k$; for each integer $1 \leq i \leq k$, $z_i = [\theta_i]$ denotes the formula

$$\text{ENC}(eP, t_1, u_1) \land \text{ENC}(u_1, t_2, u_2) \land \cdots$$

$$\land \text{ENC}(u_{m-1}, t_m, u_m) \land z_i = u_i$$

if $\theta_i = P(t_1, \ldots, t_m)$; $\text{MRG}(y_1, \ldots, y_k, [\gamma]_n, v)$ denotes

$$\text{ENC}(c_1, [\gamma_1], v_1) \land \text{ENC}(v_1, [\gamma_2], v_2) \land \cdots$$

$$\land \text{ENC}(v_{n-1}, [\gamma_n], v_n) \land \text{MRG}(y_1, y_2, w_2)$$

$$\land \cdots \land \text{MRG}(w_{k-1}, y_k, w_k) \land \text{ENC}(w_k, v, v)$$

if $\gamma_n = \gamma_1 \lor \cdots \lor \gamma_n$, for some atoms $\gamma_1, \ldots, \gamma_n$ and $n \geq 0$.

4. Let $\Pi_4$ be the program consisting of the rule

$$x = c \rightarrow \text{FALSE}(x) \quad (21)$$

and the rule

$$\text{FALSE}(x) \land \text{ENC}(x, [\theta], y) \land \neg \theta \rightarrow \text{FALSE}(y) \quad (22)$$

for every intensional predicate $P$ of $\Pi$ and every atom $\theta$ of the form $P(\theta p)$, where $z_1, z_2, \ldots$ are individual variables, $k$ is the arity of $P$, and $\theta p$ denotes the tuple $z_1 \cdots z_k$.

5. Let $\Pi_5$ be the logic program consisting of the rule

$$\text{TRUE}(c_1) \rightarrow \bot \quad (23)$$

and the following rule

$$\text{TRUE}(x) \land \text{EXT}(x, [\theta], y) \land \text{FALSE}(y) \rightarrow \theta \quad (24)$$

for each atom $\theta$ of the form same as that in $\Pi_4$.

In the end, define $\Pi^0$ as the union of programs $\Pi_1, \ldots, \Pi_5$. 

\[ \]
Next, we explain the intuition of this translation. Program \( \Pi_1 \) assures that \( \text{ENC} \) will be interpreted as an encoding function on the domain and constants in \( C_1 \) will be interpreted as ending flags for the encoding of atoms and positive clauses. Program \( \Pi_2 \) defines the merging and extracting functions, and all the encoding predicates mentioned before. Based on these assumptions, program \( \Pi_3 \) then simulates the progression of \( \Pi \) on encodings of positive clauses. As all the needed clauses will be derived in \( \omega \) stages, this simulation is realizable. Lastly, we use program \( \Pi_1 \) and \( \Pi_2 \) to decode the encodings of positive clauses. The only difficulty in this decoding is that we need represent the resulting positive clauses by a program without disjunction. As the set of positive clauses are clearly head-cycle-free, we can get over it by applying the shift operation presented in [Ben-Eliyahu and Dechter, 1994].

**Theorem 1.** Let \( \Pi \) be a disjunctive logic program. Then over infinite structures, \( \text{SM}(\Pi) \) is equivalent to \( \exists \text{SM}(\Pi^\circ) \), where \( \circ \) denotes the set of constants occurring in \( \Pi^\circ \) but not in \( \Pi \).

**Proof.** Let \( v_1, \ldots, v_5 \) and \( \tau \) be the set of all intensional predicates of \( \Pi_1, \Pi_2, \) and \( \Pi_3 \) respectively. Let \( \sigma \) be the union of \( v_1, v_2 \), and \( v(\Pi) \). Then by Splitting Lemma in [Ferraris et al., 2009] and the second-order transformation, we can conclude

\[
\text{SM}(\Pi^\circ) \equiv \text{SM}(\Pi_1) \land \cdots \land \text{SM}(\Pi_5).
\]

(25)

Let \( A \) be any infinite structure of \( v(\Pi) \), and let \( B \) be any \( \sigma \)-expansion of \( A \) that satisfies the following conditions:

1. ENC is interpreted as the graph of an encoding function \( \text{enc}^A \) on \( A \) such that no element among \( c_1^A \) and \( c_2^A \) (for all \( P \in \pi \)) belongs to the range of \( \text{enc} \). ENC is interpreted as the complement of the graph of \( \text{enc} \), \( \Omega_1^A = A \times A \).
2. MRG and EXT are interpreted as graphs of the merging and extracting functions related to \( \text{enc} \) and \( c_1^A \) respectively, \( \text{IN} \), \( \text{SUBC} \), \( \text{EQU} \) are interpreted as encoding predicates \( \text{in} \), \( \text{subc} \), \( \text{equ} \) related to \( \text{enc} \) and \( c_2^A \) respectively.

For convenience, we need define some notations. Given a grounded atom \( \vartheta \) of the form \( P(a_1, \ldots, a_k) \) for any \( P \in \pi \), let \( \langle \vartheta \rangle \) be short for \( \text{enc}(a_1, \ldots, a_k; c_2^A) \). Given a grounded clause \( C \in \text{GPC}(\tau, A) \) of the form \( \vartheta \lor \cdots \lor \vartheta_n \) where each \( \vartheta_i \) is an atom, let \( \langle C \rangle \) denote \( \text{enc}(\langle \vartheta_1 \rangle, \ldots, \langle \vartheta_n \rangle; c_2^A) \). Given a set \( \Sigma \subseteq \text{GPC}(\tau, A) \), let \( \langle \Sigma \rangle \) denote the set \( \{ \langle C \rangle : C \in \Sigma \} \). Moreover, let \( \Delta^\tau(B) = \{ a \in B : \text{true}(a) \in \Gamma_1^A \uparrow \omega \} \). By an induction on \( n \), we can show the following claim:

**Claim 1.** For all integers \( n \geq 0 \), \( (\Gamma_1^A \uparrow n) = \Delta^\tau(B) \).

Now, let \( B^+ \) be the \( (\Pi^\circ) \)-expansion of \( B \) that interprets \( \text{TRUE} \) as the set \( \bigcup_{n \geq 0} \Delta^\tau(B) \), and interprets \( \text{FALSE} \) as the set of all \( C \in \text{GPC}(\tau, A) \) such that \( \text{INS}(\langle C \rangle, \tau) \models \lnot C \).

**Claim 2.** Let \( C = B^+ \). Then \( \text{INS}(A, \tau) \) is a minimal model of \( \Gamma_1^A \uparrow \omega \) iff \( \text{INS}(C, \tau) \) is a minimal model of \( \Pi^\circ \).

To prove this claim, we need check the validity of decoding, and then show that the shift operation preserves the semantics, which is similar to that of Theorem 4.17 in [Ben-Eliyahu and Dechter, 1994]. Due to the limit of space, we leave the detailed proof to a full version of this paper.

With these two claims, we can prove the theorem now:

**“=”** Suppose \( A \) is an infinite model of \( \text{SM}(\Pi) \). Let \( B \) be a \( \sigma \)-expansion of \( A \) defined by conditions 1 and 2. The existence of expansion \( B \) is clearly assured by the infiniteness of \( A \). It is easy to check that \( B \) is a stable model of both \( \Pi_1 \) and \( \Pi_2 \). Let \( C \) be the structure \( B^+ \). Then it is also clear that \( C \) is a stable model of both \( \Pi_3 \) and \( \Pi_4 \). On the other hand, by Proposition 2 and the assumption, \( \text{INS}(A, \tau) \) should be a minimal model of \( \Gamma_1^A \uparrow \omega \). So, by Claim 2, \( \text{INS}(C, \tau) \) is a minimal model of \( \Pi^\circ \), which means that \( C \) is a stable model of \( \Pi_5 \) by Proposition 1. By equation (25), \( C \) is then a stable model of \( \Pi^\circ \), which means that \( A \) satisfies \( \exists \text{SM}(\Pi^\circ) \).

**“⇐”** Suppose \( A \) is an infinite model of \( \exists \text{SM}(\Pi^\circ) \). Then there is an \( (\Pi^\circ) \)-expansion \( C \) of \( A \) such that \( C \) is a stable model of \( \Pi^\circ \). Let \( B \) be the restrictions of \( C \) to \( \sigma \). By equation (25), \( B \) is a model of the formulae \( \text{SM}(\Pi_1) \) and \( \text{SM}(\Pi_2) \) which implies that ENC is interpreted as an encoding function for \( \pi \) on \( A \), \( c_1 \) and \( c_2 \) for all predicates \( P \in \pi \) are interpreted as elements not in the range of \( \text{enc} \), predicates MRG, EXT, IN, SUBC, EQU are interpreted as the corresponding functions or predicates mg, ext, in, subc, equ related to \( \text{enc} \) and \( c_2^A \). By equation (25) again, \( C \) is a stable model of \( \Pi_3 \) and \( \Pi_4 \), and by Proposition 2, \( \text{INS}(C, \varnothing) \) is then a minimal model of \( \Gamma_1^A \uparrow \omega \). So, we have \( C = B^+ \). According to Proposition 1, \( \text{INS}(C, \tau) \) should be a minimal model of \( \Pi_5 \) as \( C \) is clearly a stable model of \( \Pi_5 \). Applying Claim 2, we then have that \( \text{INS}(A, \tau) \) is a minimal model of \( \Gamma_1^A \uparrow \omega \). Thus, by Proposition 2, \( A \) should be a stable model of \( \Pi \).

**Remark 2.** Note that, given any finite domain \( A \), there is no injective function from \( A \times A \) into \( A \). Therefore, we can not expect that the above translation works on finite structures.

From the above theorem, we then get the following result:

**Corollary 1.** DLP \( \simeq_{\text{INF}} \text{NLP} \).

### 4 Finite Structures

In this section, we will focus on the relationship between disjunctive and normal logic programs over finite structures. In the general case, the separation of their expressive power is turned out to be very difficult due to the following result:

**Proposition 3.** DLP \( \simeq_{\text{FIN}} \text{NLP} \) iff NP = coNP.

**Proof.** Let \( \Sigma^1_2 \) denote the set of sentences of the form \( \exists \tau \forall \sigma \varphi \), where \( \tau \) and \( \sigma \) are finite sets of predicate variables, \( \varphi \) is a first-formula order. Let ESO be the set of sentences of the above form such that \( \sigma \) is empty. By Fagin Theorem [1974] and Stockmeyer’s characterization of the polynomial hierarchy [1977], we have that \( \Sigma^1_2 \simeq_{\text{FIN}} \text{ESO} \) iff \( \Sigma^2_2 \subseteq \text{NP} \). As a routine complexity theoretical argument, we also have that \( \Sigma^2_2 \subseteq \text{NP} \) iff \( \text{NP} \subseteq \text{coNP} \). On the other hand, according to the proof of Theorem 6.3 in [Eiter et al., 1997], or by Lemma 2 in this section, Leivant’s Normal Form [1989] and the definition of SM, we can conclude that DLP \( \simeq_{\text{FIN}} \Sigma^1_2 \); by Lemma 1 in this section\(^2\), it holds that NLP \( \simeq_{\text{FIN}} \text{ESO} \). Combining these results, we then have the desired proposition. 

---

1 A similar result over function-free Herbrand structures follows from the expressive power of traditional answer set programs. Here is a reformulation of it under the general stable model semantics.

2 Note that functions in both existential second-order logic and normal logic programs can be easily simulated by predicates.
Due to the significant difficulty of general separation, the rest of this section is devoted to a weaker separation between disjunctive and normal programs. To do this, we first study some relationship between logic programs and classical logic.

In the following, let $\text{ESO}^k_\mathbb{F}[\mathbb{V}^*]$ denote the set of all sentences of the form $\exists \forall \exists_\mathbb{V} \varphi$, where $\tau$ is a finite set of predicate and function variables of arity $\leq k$, and $\varphi$ is quantifier-free.

**Lemma 1.** $\text{NLP}^k_\mathbb{F} \simeq_{\text{FIN}} \text{ESO}^k_\mathbb{F}[\mathbb{V}^*]$ for all $k > 1$.

**Proof.** $\geq_{\text{FIN}}$ Let $\varphi$ be a sentence in $\text{ESO}^k_\mathbb{F}[\mathbb{V}^*]$. It is clear that $\varphi$ can be rewritten as an equivalent sentence of the form $\exists \forall \exists_\mathbb{V} \varphi$.

Let $\Pi$ be a logic program consisting of the rule $\gamma_i \rightarrow \lambda$ for each $1 \leq i \leq n$, where $\gamma_i$ is obtained from $\gamma_i$ by substituting $\varphi$ for each negated atom $\neg \varphi_i$ and substituting $\land$ for each connective $\lor$. Obviously, $\exists \forall \exists_\mathbb{V} \varphi$ is in $\text{NLP}^k_\mathbb{F}$ and equivalent to $\varphi$.

$\leq_{\text{FIN}}$ Let $\exists \forall \exists_\mathbb{V} \varphi$ be a formula in $\text{NLP}^k_\mathbb{F}$ such that $\Pi$ is a normal logic program. Without loss of generality, we assume the head of each rule in $\Pi$ is of the form $P(\bar{x})$ for some integer $i \geq 0$ and $l$-ary intensional predicate $P$ of $\Pi$. Let $\tau$ and $\Sigma$ be the sets of all intensional predicates and atoms of $\Pi$ respectively. For all $\sigma \in \Sigma$, define $\gamma_i = \neg \psi_i \rightarrow \lambda$, where $\psi_i$ is the tuple of all individual variables occurring in $\gamma_i$ but not in $\lambda$. Now, we can define $\varphi$ as the conjunction of formulae

$$(\varphi_i \land \varphi_j \land \cdots \land \varphi_m) \lor \lambda \rightarrow \text{DRVBL}(\lambda)$$

for all $1 \leq i \leq n$, and define $\psi_i$ as the formula

$${\text{DRVBL}}(\lambda) \rightarrow \lambda \land \exists F_{\psi_i} \left[ \psi_i \land \bigwedge_{j=1}^{m} \text{DRVLESS}(\psi_j, \lambda) \right]$$

where $\psi_i$ is a new binary predicate and $\varphi$ is a new individual constant, which are intended to be interpreted as a strict total order and the maximal element in the order respectively; $\psi_i$ is the formula describing that $\psi_i$ is strictly less than $\lambda$. $\psi_i$ is the formula describing that $\psi_i$ is strictly less than $\lambda$. $\psi_i$ is a new predicate of the same arity as $\varphi_i$, and $\bar{y}_i$ is the tuple of all individual variables occurring in $\gamma_i$ but not in $\lambda$.

Define $\varphi_{\Pi}$ as the universal closure of conjunction of the formula $\varphi_{\Pi}$ and the formula $\varphi_{\Pi} \land \psi_i$ for all $\alpha \in \Sigma$, where $\varphi_{\Pi}$ is a sentence in $\text{ESO}^k_\mathbb{F}[\mathbb{V}^*]$ which describes that $\psi_i$ is a strict total order and $\varphi_{\Pi}$ is the largest element in that order. By a moment's thought, we can construct such a sentence. Clearly, $\varphi_{\Pi}$ is equivalent to a sentence in $\text{ESO}^k_\mathbb{F}[\mathbb{V}^*]$ by introducing Skolem functions. Now, we observe that $\text{SM}(\Pi) \equiv_{\text{FIN}} \exists \forall \varphi_{\Pi}$, where $\sigma$ is the set of constants occurring in $\varphi_{\Pi}$ but not in $\Pi$.

Due to the limit of space, we only show $\leq_{\text{FIN}}$. Assume $\mathbb{F}$ is a finite model of $\varphi_{\Pi}$. By the formula $\varphi$, the predicate $\psi_i$ is interpreted as a strict total order on $B$ and $\text{max}^B$ is the restriction of $\mathbb{B}$ to the vocabulary $\psi_{\Pi}$. To show that $\psi_{\Pi}$ is a stable model of $\Pi$, by Proposition 2 it suffices to show that $\text{Ins}(\mathbb{A}, \psi_{\Pi}) = \Gamma^\mathbb{A}_\Pi \uparrow\omega$. Through a routine induction on $n$, we can show that $\Gamma^\mathbb{A}_\Pi \uparrow\omega \subseteq \text{Ins}(\mathbb{A}, \psi_{\Pi})$ for all integers $n \geq 0$.

By notation $\Sigma_{1,2}^k[\mathbb{V}^*, \mathbb{E}]$ we denote the set of all sentences of the form $\exists \forall \exists_\mathbb{V} \exists_\mathbb{E} \varphi$, where $\tau$ and $\sigma$ are two finite sets of predicate variables of arity $\leq k$, $\bar{x}$ is a $k$-tuple of individual variables, $\psi$ is quantifier-free. Now we then have:

**Lemma 2.** $\Sigma_{1,2}^k[\mathbb{V}^*, \mathbb{E}] \subseteq_{\text{FIN}} \text{DLP}^k_\mathbb{F}$ for all $k > 1$.

**Proof.** (Sketch) Let $\exists \forall \exists_\mathbb{V} \psi \in \Sigma_{1,2}^k[\mathbb{V}^*, \mathbb{E}]$ where $\tau, \sigma$ are finite sets of predicate variables of arities $\leq k$. Without loss of generality, suppose $\psi = \exists \forall \exists_\mathbb{E} \varphi(\bar{x}, \bar{y})$, where $\bar{x}$ is of length $k$; $\forall \varphi$ is a formula built from literals and connectives $\land$ and $\lor$. By a modification of the proof of Theorem 6.3 in [Eiter et al., 1997], we can obtain a disjunctive logic program with auxiliary predicates of arities $\leq k$ which defines the property expressed by $\exists \forall \exists_\mathbb{V} \psi$.

Let $\Pi_1$ be a logic program same as $\Pi_2$ in the proof of Theorem 6.3 in [Eiter et al., 1997] where the binary predicate $S$ defines a successor relation on the domain; the unary predicates $P$ and $L$ define the singleton sets that contain the least and the maximal elements in the order defined by $S$ respectively. Let $\Pi_2$ be the program consisting of (i) the rules $X(\bar{u}) \lor X'(\bar{u})$ for all predicate variables $X$ in $\tau$ or $\sigma$, (ii) the rules

$$L(\bar{x}) \land D(\bar{x}) \rightarrow Y(\bar{u}) \land X'(\bar{u})$$

for all predicate variables $Y$ in $\sigma$, (iii) the rules

$$F(\bar{x}) \land \varphi(\bar{x}, y) \rightarrow D(\bar{x})$$

and (iv) the rule $L(\bar{z}) \land \neg D(\bar{z}) \rightarrow D(\bar{z})$, where, for every predicate $X$ in $\tau$ or $\sigma$, $X'$ is a new predicate of the same arity; $\varphi$ is the formula obtained from $\psi$ by substituting $Y'(\bar{t})$ for all occurrences of $Y(\bar{t})$ if $Y < \sigma$ and $\bar{t}$ is a tuple of items; $D$ is a $k$-ary new predicate; $P(x_1, \ldots, x_k)$ denotes the formula $P(x_1) \land \cdots \land P(x_k)$ if $P$ is $F$ or $L$; and $\overline{S}(\bar{x}, \bar{z})$ denotes a quantifier-free formula defining the lexicographic order generated by $S$ on $k$-tuples. Note that, in general, formulae (27) and (28) are not rules defined previously. However, by applying some distributivity-like laws [Cabalar et al., 2005], they can be replaced by an (strongly) equivalent set of rules.

Let $\Pi_1$ be the union of $\Pi_1$ and $\Pi_2$. By a similar but slightly more complex argument than that in Theorem 6.3 of [Eiter et al., 1997], we can show that $\forall \exists \varphi \equiv_{\text{FIN}} \exists \forall \varphi_{\Pi}$, where $\gamma$ is the set of all predicates in $\psi_{\Pi}$ but not in $\psi_{\Pi}$.
Proof. (Sketch) Let $n$ be an integer $\geq 1$ and $v$ the vocabulary consisting of only an $n$-ary predicate $P$. Define \textup{PARITY} to be the class of $v$-structures in each of which $P$ is interpreted as a set consisting of an even number of $n$-tuples. We first show that the property \textup{PARITY} is definable in DLP$^k$.

Let $\varphi_1$ be a formula asserting “$S$ is interpreted as a successor relation on the domain; 0 and $m$ are interpreted as the least and the maximal elements in the order defined by $S$ respectively”. Let $\varphi_2$ be the following formula:

$$(Y(0) \leftrightarrow P(x,0)) \land \forall \bar{u} \bar{v}[S(u, \bar{v}) \rightarrow P(x, \bar{v})]$$

where $\varphi_\psi \land \chi$ denotes the formula $\varphi \leftrightarrow \chi$; $\bar{c}$ denotes the tuple $(c, \ldots, c)$ of length $k$ if $c$ is 0 or $m$. It is easy to see that $\varphi_2$ describes the property “$X(\bar{a})$ is true iff the cardinality of the set $\{b : P(\bar{a}, b)\}$ is odd”. Let $\varphi_3$ be the following formula:

$$(X(0) \leftrightarrow Y(0)) \land \forall \bar{u} \bar{v}[S(\bar{u}, \bar{v}) \rightarrow (X(\bar{v}) \leftrightarrow Y(\bar{v}) + Y(\bar{u}))]$$

This formula asserts “$X$ consists of an even number of $k$-tuples on the domain”. Now let $\varphi$ be the following sentence:

$$\exists m \exists \exists \bar{S}_1 \land \exists X \forall Y \forall \bar{\varphi}_1 \land \varphi_3$$

It is not difficult to check that $\varphi$ defines \textup{PARITY} over finite structures. By Lemma 2, there is a logic program $\Pi_0$ such that $\exists X \forall Y \forall \bar{\varphi}_1 \land \varphi_3 \equiv \text{FIN} \exists \varphi \exists \text{SM}(\Pi_0)$ $\in$ DLP$^k$, where $\tau$ is a finite set of predicates of arity $\leq k$. On the other hand, $\varphi_1$ can be easily encoded by a disjunctive logic program $\Pi_3$ involving only predicates of arity $\leq 2$. Hence, we have that $\varphi \equiv \text{FIN} \exists m \exists \exists \bar{S}_1 \land \exists \varphi \exists \text{SM}(\Pi_0)$.

Let $\Pi$ be $\Pi_0 \cup \Pi_0$. By Splitting Lemma in [Ferraris et al., 2009], $\varphi \equiv \text{FIN} \exists m \exists \exists \bar{S}_1 \land \exists \varphi \exists \text{SM}(\Pi_0)$. Let $\Pi'$ be the program obtained from $\Pi_0$ by simulating individual constants $0$ and $m$ by unary predicates. This is then the desired program.

Next, we show that \textup{PARITY} is undefinable in NLP$^{2k-1}$ over finite structures. By Lemma 1, it suffices to show that \textup{PARITY} is undefinable in ESO$^{2k-1}$. Towards a contradiction, assume that there is a sentence $\psi$ in this class such that finite models of $\psi$ are exactly the structures in \textup{PARITY}. By employing an idea similar to that in Theorem 3.1 of [Durand et al., 1998], we can then construct a formula $\psi_0$ in ESO$^{2k-2}$ to define \textup{PARITY}$^k$. However, according to Theorem 2.1 of [Ajtai, 1983], this is impossible since $\psi_0$ has an equivalent in ESO$^{2k-1}$; i.e., the set of formulae in ESO$^{2k-1}$ without variable functions. This then completes the proof.

5 Arbitrary Structures

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

Theorem 3. DLP $\simeq$ NLP if and only if DLP $\simeq$ FIN NLP.

Proof. (Sketch) It is trivial from left to right. Now we show the converse direction. Assume DLP $\simeq$ FIN NLP. Then, for each disjunctive logic program $\Pi$, there should be a normal logic program $\Pi^*$ such that SM$(\Pi) \equiv_{\text{FIN}} \exists \varphi \exists \text{SM}(\Pi^*)$, where $\varphi$ is the set of all predicates and functions occurring in $\Pi^*$ but not in $\Pi$. To show DLP $\simeq$ NLP, our idea is to design a logic program testing whether or not the model currently considered is finite. If that is true, we then let $\Pi^*$ work; otherwise, let $\Pi^*$ which is developed for infinite structures work. To do this, we introduce two proposition constants, INF and FIN, as flags. Let $\Pi_{\text{inf}}$ be the program consisting of following rules:

1. $\neg \text{ARC}(x, y) \rightarrow \text{ARC}(y, x)$, 4. $\neg \text{OK}_a(x) \rightarrow \text{OK}_a(x)$,
2. $\neg \text{ARC}(x, y) \rightarrow \text{ARC}(y, x)$, 5. $\text{ARC}(x, y) \rightarrow \text{INF}$,
3. $\text{ARC}(x, y) \rightarrow \text{OK}_a(x)$, 6. $\neg \text{INF} \rightarrow \text{INF}$,

and rule $\text{ARC}(x, y) \land \text{ARC}(y, z) \rightarrow \text{ARC}(x, z)$. This program sets flag INF to be true if the intended model is infinite. When the intended model is finite, we use program $\Pi_{\text{fin}}$ to set flag FIN, which is obtained from $\Pi_1$ in Lemma 6.4 of [Eiter et al., 1997] by substituting FIN for $\text{Order}$ and by applying the shift operation in Section 4.5 of [Ben-Eliyahu and Dechter, 1994]. Let $\Pi_{\text{inf}, \Pi_{\text{fin}}}$ be the program obtained from $\Pi^*$ (\Pi$^*$, respectively) by adding INF (FIN, respectively) to the body of each rule as a conjunct. Let $\Pi_{\text{inf}, \Pi_{\text{fin}}}$ be the union of $\Pi_{\text{inf}}$, $\Pi_{\text{fin}}$, $\Pi_{\text{inf}}$, and $\Pi_{\text{fin}}$. We can show $\text{SM}(\Pi) \equiv \exists \varphi \exists \text{SM}(\Pi^*)$, where $\varphi$ is the set of all constants occurring in $\Pi^*$ but not in $\Pi$. □

Remark 3. In classical logic, it is well-known that separating languages over arbitrary structures is usually easier than that over finite structures. This is not surprising as arbitrary structures enjoy a lot of properties, including the compactness and the interpolation theorem, that fail on 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; Zhang and Zhang, 2013]. Therefore, the above result sheds a new insight on the stronger separations of DLP from NLP over finite structures.

From Theorem 3 and Proposition 3, we immediately have:

Corollary 2. DLP $\simeq$ NLP if and only if $\text{FIN}$ NLP = $\text{coNP}$.

6 Related Works and Conclusion

Over Herbrand structures, [Eiter and Gottlob, 1997; Schlipf, 1995] showed that both disjunctive and normal logic programs define the same class of database queries if functions are allowed. Our result over infinite structures is more general and stronger than theirs as Herbrand structures are only a special class of countable infinite structures. It is not clear whether or not their approach, which employs the inductive definability from [Barwise, 1976; Moschovakis, 1974], can be generalized to arbitrary infinite structures.

To the best of our knowledge, the weaker separation over finite structures in this paper gives us the first lower bound for arities of auxiliary predicates in the translatability from disjunctive logic programs into normal logic programs. Improving the lower bound will shed light on deeply understanding the expressive power of disjunctive and normal logic programs, which will be a challenging task in the further study. The equivalence of the translatability over finite structures and over arbitrary structures provides us a new perspective to achieve this goal. We will pursue this in the near future.
References

[Ajtai and Gurevich, 1994] M. Ajtai and Y. Gurevich. Data-
log vs first-order logic. Journal of Computer and System

[Ajtai, 1983] M. Ajtai. $\Sigma_1^1$-formulae on finite structures. Ann-


[Ben-Eliyahu and Dechter, 1994] R. Ben-Eliyahu and
R. Dechter. Propositional semantics for disjunctive
logic programs. Annals of Mathematics and Artificial

Reducing propositional theories in equilibrium logic to

[Dantsin et al., 2001] E. Dantsin, T. Eiter, G. Gottlob, and
A. Voronkov. Complexity and expressive power of logic
programming. ACM Computing Surveys, 33(3):374–425,

[Durand et al., 1998] A. Durand, C. Lautemann, and
T. Schwentick. Subclasses of binary NP. Journal of Logic

[Ebbinghaus and Flum, 1999] H.-D. Ebbinghaus and
J. Flum. Finite Model Theory. Springer-Verlag, New

siveness of stable model semantics for disjunctive logic
programs with functions. The Journal of Logic Program-

[Eiter et al., 1997] T. Eiter, G. Gottlob, and H. Mannila. Dis-
junctive datalog. ACM Transactions on Database Systems,

[Fagin, 1974] R. Fagin. Generalized first-order spectra and
polynomial-time recognizable sets. In Complexity of Com-
putation, volume 7 of SIAM-AMS Proceedings, pages 43–
73, 1974.

[Ferraris et al., 2009] P. Ferraris, J. Lee, V. Lifschitz, and
R. Palla. Symmetric splitting in the general theory of sta-

Stable models and circumscription. Artificial Intelligence,

The stable model semantics for logic programming. In

[Leivant, 1989] D. Leivant. Descriptive characterizations of
computational complexity. Journal of Computer and Sys-

logic programming to circumscription via logic of GK. Ar-

Foundations of Disjunctive Logic Programming. The MIT

[Moschovakis, 1974] Y. Moschovakis. Elementary Induc-
tion on Abstract Structures. North Holland, Amsterdam,
1974.

[Schlipf, 1995] J. S. Schlipf. The expressive powers of the
logic programming semantics. Journal of Computer and System


[Zhang and Zhang, 2013] H. Zhang and Y. Zhang. First-
order expressibility and boundedness of disjunctive logic