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 over finite, infinite, or arbitrary structures. 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] 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 intensional predicates of arities \( \leq k \) that can not be equivalently translated to any normal program with auxiliary predicates and functions of arities \( \leq 2k \). 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 \( \phi \) and each set \( \Sigma \) of formulae, let \( \nu(\phi) \) and \( \nu(\Sigma) \) be the sets of all constants occurring in \( \phi \) and \( \Sigma \) respectively. Let \( Q_\tau \) and \( Q\bar{\tau} \) 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 \( 1 \leq i \leq n \), \( x = x_1 \cdots x_m \), \( Q \) is \( \forall \) or \( \exists \), \( X_j \) and \( x_i \) are predicate/function and individual variables respectively.

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

Every assignment in a structure $\mathcal{A}$ 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 $\varphi$ and an assignment $\alpha$ in $\mathcal{A}$, we write $(\mathcal{A}, \alpha) \models \varphi$ if $\alpha$ satisfies $\varphi$ in $\mathcal{A}$ in the standard way.

In particular, if $\varphi$ is a sentence, we simply write $\mathcal{A} \models \varphi$, and say $\mathcal{A}$ is a model of $\varphi$, or in other words, $\mathcal{A}$ satisfies $\varphi$. Given two (second-order) formulae $\psi$ and $\varphi$ and a class $C$ of structures, we say $\varphi$ is equivalent to $\psi$ over $C$, or write $\varphi \equiv_C \psi$ for short, if for every structure $\mathcal{A}$ in $C$ and every assignment $\alpha$ in $\mathcal{A}$, $\alpha$ satisfies $\varphi$ in $\mathcal{A}$ if and only if $\alpha$ satisfies $\psi$ in $\mathcal{A}$.

Suppose $\tau$ is a set of predicates and $A$ is a domain, i.e., a nonempty set. Let $G\mathcal{A}(\tau, A)$ denote the set of $P(\bar{a})$ for all predicates $P \in \tau$ and all $n$-tuples $\bar{a}$ on $A$ where $n$ is the arity of $P$. Let $G\mathcal{P}(P, \tau, A)$ be the set of finite disjunctions built from atoms in $G\mathcal{A}(\tau, A)$. Each element in $G\mathcal{A}(\tau, A)$ ($G\mathcal{P}(P, \tau, A)$) is called a grounded atom (grounded positive clause) of $\tau$ over $A$.

A structure $\mathcal{A}$, let $\text{INS}(\mathcal{A}, \tau)$ be the set of grounded atoms $P(\bar{a})$ such that $P \in \tau$ and $P(\bar{a})$ is true in $\mathcal{A}$.

Let $\text{FIN}$ denote the class of all finite structures, and let $\text{INF}$ denote the class of all infinite structures. Suppose $\Sigma$ and $\Pi$ are two sets of second-order formulae and let $C$ be a class of structures. We write $\Sigma \subseteq_C \Pi$ if for each formula $\varphi$ in $\Sigma$, there is a formula $\psi$ in $C$ such that $\varphi \equiv_C \psi$. We write $\Sigma \subsetneq_C \Pi$ if both $\Sigma \subseteq_C \Pi$ and $\Pi \subsetneq \Sigma$ hold. In particular, if $C$ is the class of all arbitrary structures, the subscript $C$ may 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 $1 \leq m \leq n$, 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 $\Pi$ be a disjunctive logic program. Then each intensional predicate of $\Pi$ is a predicate constant that occurs in the head of some rule in $\Pi$. Atoms built from intensional predicates of $\Pi$ are called intensional atoms of $\Pi$.

Let $\Pi$ be a disjunctive logic program. Then $\Pi$ is normal if the head of each rule contains at most one atom. $\Pi$ is plain if there is no negation of any intensional atom of $\Pi$ occurring in any of its rules. $\Pi$ is propositional if no predicate of positive arity occurs in any of its rules, and $\Pi$ 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 $\Pi$, let $\text{SM}(\Pi)$ denote the formula $\forall \tau \forall \tau^* (\star \tau \rightarrow \neg \varphi^*)$, $\varphi$ is the set of all intensional predicate constants of $\Pi$; $\tau^*$ is the set of predicate variables $P^*$ for all $P \in \tau$; $\star \tau$ is the formula $\land_{P \in \tau} \forall \bar{x} (P^*(\bar{x}) \rightarrow P(\bar{x}))$; $\neg \land_{P \in \tau} \forall \bar{x} (P^*(\bar{x}) \rightarrow P(\bar{x}))$; $\varphi$ is the conjunction of all sentences $\forall(\gamma)$ such that $\gamma$ is a rule in $\Pi$ and $\forall(\gamma)$ is the universal closure of $\gamma$; $\varphi^*$ is the conjunction of $\forall(\gamma^*)$ such that $\gamma^*$ is a rule in $\Pi$ and $\gamma^*$ is the rule obtained from $\gamma$ by substituting $P^*(\bar{t})$ for all positive occurrences of $P(\bar{t})$ in its head or its body if $P$ is in $\tau$. A structure $\mathcal{A}$ is a stable model of $\Pi$ if it is a model of $\text{SM}(\Pi)$.

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 $\exists \Pi \text{SM}(\Pi)$ are exactly the structures in $C$, where $\tau$ is a set of predicate and function constants occurring in $\Pi$. Constants in $\tau$ are called auxiliary constants.

Now, given a rule $\gamma$, a structure $\mathcal{A}$ and an assignment $\alpha$ in $\mathcal{A}$, let $\gamma[\alpha]$ be the rule obtained from $\gamma$ by substituting $P(\bar{a})$ for all atoms $P(\bar{t})$ where $\bar{t} = \alpha(\bar{a})$, let $\gamma_B$ be the set of all conjuncts in the body of $\gamma$ in which no intensional predicate positively occurs, and let $\gamma^+$ be the rule obtained from $\gamma$ by removing all literals in $\gamma_B^-$. Given a disjunctive logic program $\Pi$, let $\Pi^\alpha$ be the set of rules $\gamma^+[\alpha]$ for all assignments $\alpha$ in $\mathcal{A}$ and all rules $\gamma$ in $\Pi$ such that $\alpha$ satisfies $\gamma^+$. 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 $\Pi$ be a disjunctive logic program with a $\tau$ set of intensional predicates. Then an $v(\Pi)$-structure $\mathcal{A}$ is a stable model of $\Pi$ iff $\text{INS}(\mathcal{A}, \tau)$ is a minimal (via set inclusion) model of $\Pi^\alpha$.

#### 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 $\Pi$ is a propositional, possibly infinite and plain disjunctive logic program, and $\Sigma$ is a set of finite disjunctions of atoms in $v(\Pi)$. We define $\Gamma_\Pi(\Sigma)$ as the set of all positive clauses $H \lor C_1 \lor \cdots \lor C_k$ such that $k \geq 0$ and there are a rule $p_1 \land \cdots \land p_k \rightarrow H$ in $\Pi$ and a sequence of positive clauses $C_1 \lor p_1, \ldots, C_k \lor p_k$ in $\Sigma$. It is easy to verify that $\Gamma_\Pi$ is a monotonic function on the sets of positive clauses of $v(\Pi)$.

Now, by the first-order GL-reduction defined above, a progression operator for first-order logic programs is then defined. Let $\Pi$ be a disjunctive logic program and let $\mathcal{A}$ be a structure of $v(\Pi)$. We define $\Gamma_\Pi^\mathcal{A} = \text{operator } \Gamma_\Pi$. Furthermore, define $\Gamma_\Pi^\mathcal{A} \uparrow n$ as the empty set, and define $\Gamma_\Pi^\mathcal{A} \uparrow n$ as the union of $\Gamma_\Pi^\mathcal{A} \uparrow n - 1$ and $\Gamma_\Pi^\mathcal{A} \uparrow (\Gamma_\Pi^\mathcal{A} \uparrow n - 1)$ for all integers $n > 0$. Finally, let $\Gamma_\Pi^\mathcal{A} \uparrow \omega$ be the union of $\Gamma_\Pi^\mathcal{A} \uparrow n$ 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 $\Pi$ be a disjunctive logic program, $\tau$ the set of all intensional predicates of $\Pi$, and $\mathcal{A}$ a structure of $v(\Pi)$. Then $\mathcal{A}$ is a stable model of $\Pi$ iff $\text{INS}(\mathcal{A}, \tau)$ is a minimal model of $\Gamma_\Pi^\mathcal{A} \uparrow \omega$. 


Remark 1. In above proposition, it is easy to see that, if $\Pi$ is normal, $A$ is a stable model of $\Pi$ if and only if $\text{INS}(A, \tau) = \Gamma_\Pi \uparrow \omega$.

3 Infinite Structures

In this section, we propose a translation that turns each disjunctive logic program into an equivalent normal logic 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 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 on $A$ is an injective function from $A \times A$ into $A$. Let $\text{enc}$ be an encoding function on $A$ and $c$ an element in $A$ such that $\text{enc}(a, b) \neq c$ for all elements $a, b \in A$. For the sake of convenience, we let $\text{enc}(a_1, \ldots, a_k; c)$ denote the following expression

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

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

$$\text{merg}(\text{enc}(a; c), \text{enc}(b; c)) = \text{enc}(\bar{a}, \bar{b}; c)$$

for all tuples $\bar{a}, \bar{b} \in A^*$. The extracting function $\text{ext}$ on $A$ related to $\text{enc}$ and $c$ is the function from $\text{enc}(A, c) \times A$ into $\text{enc}(A, c)$ that satisfies $\text{ext}(\text{enc}(a; c), b) = \text{enc}(\bar{a}; c)$, where $\bar{a}$ 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 $\text{enc}$ 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 $\text{enc}$ and $c$. The predicate $\text{in}$ is a subset of $\text{enc}(A, c) \times A$ such that $\text{enc}(\bar{a}, c), b) \in \text{in}$ iff $b$ occurs in $\bar{a}$; the predicate $\text{subc}$ is a binary relation on $\text{enc}(A, c)$ such that $(\text{enc}(\bar{a}, c), \text{enc}(b, c)) \in \text{subc}$ iff all the elements in $\bar{a}$ occur in $\bar{b}$. The predicate $\text{equiv}$ is a binary relation on $\text{enc}(A, c)$ such that $(\text{enc}(a, b; c), \text{enc}(b, c; c))$ in $\text{equiv}$ iff $(a, b) \in \text{subc}$ and $(b, c) \in \text{subc}$.

Example 1. Let $\mathbb{Z}^+$ be the set of all positive integers, and define $e(m, n) = 2^m + 3^n$ for all integers $m, n \in \mathbb{Z}^+$. Then $e$ is clearly an encoding function on $\mathbb{Z}^+$, 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 $2^{150} + 3^5$; the positive clause $P_2(1, 3, 5) \lor P_3(2) \lor P_1(4)$ can be encoded as $e(1, 3, 5; 2), e(2, 3; 4), 1), 4)$, 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 $C_{\Pi}$ be the set consisting of an individual constant $c_P$ for each predicate constant $P$ that occurs in $\Pi$, and of individual constant $c_e$. Let $\Pi_1$ consist of the rule

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

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

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

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

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

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

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

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

$$\text{ENC}(x, y, z) \land \text{ENC}(x, y, u) \land \neg z = u \rightarrow \perp$$

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

$$y = c_e \rightarrow \text{MRG}(x, y, x)$$

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

$$x = c_e \rightarrow \text{EXT}(x, y, x)$$

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

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

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

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

$$x = c_e \rightarrow \text{SUBC}(x, y)$$

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

$$\text{SUBC}(x, y) \land \text{SUBC}(x, y) \rightarrow \text{SUBC}(x, y)$$

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

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

and the rule

$$\text{TRUE}(x_1) \land z_1 \equiv [\theta_1] \land \text{IN}(z_1, 1) \land \cdots$$

$$\text{TRUE}(x_k) \land z_k \equiv [\theta_k] \land \text{IN}(z_k, x_k) \land \cdots$$

$$\text{EXT}(x_1, z_1, y_1) \land \cdots \land \text{EXT}(x_k, z_k, y_k) \rightarrow \text{TRUE}(v)$$

$$\text{MRG}(y_1, y_2, v) \land \text{EQU}(y_1, v) \rightarrow \text{TRUE}(v)$$

$$\text{MRG}(y_1, \ldots, y_k, \gamma_1, 

where $\gamma_1$ is the head of $\gamma_1$, $\gamma_n$ is the conjunction of literals occurring in the body of $\gamma$ but not in $\theta_1, \ldots, \theta_k$; for each integer $1 \leq i \leq k$, $z_i \equiv [\theta_i]$ denotes the formula

$$\text{ENC}(c_P, t_{1,1}, u_1^1) \land \text{ENC}(t_{1,2}, u_1^2) \land \cdots$$

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

if $\theta_1 = P(t_1, \ldots, t_m); \text{MRG}(y_1, \ldots, y_k, \gamma_1, v)$ denotes

$$\text{MRG}(c_P, [\zeta_1], v_1) \land \text{ENC}(c_P, [\zeta_2], v_2) \land \cdots$$

$$\land \text{ENC}(c_P, [\zeta_n], v_n) \land \text{MRG}(y_1, y_2, w_2) \land \cdots$$

$$\land \text{MRG}(w_{m-1}, [\zeta_m], v_n) \land \text{ENC}(w_m, v_n, v)$$

if $\gamma_n = \zeta_1 \lor \cdots \lor \zeta_n$ for some atoms $\zeta_1, \ldots, \zeta_n$ and $n \geq 0$.

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

$$x = c_e \rightarrow \text{FALSE}(x)$$

and the rule

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

for every intensional predicate $P$ of $\Pi$ and every atom $\theta$ of the form $P(z_P)$, where $z_1, z_2, \ldots$ are individual variables, $k_P$ is the arity of $P$, and $z_P$ denotes the tuple $z_1 \cdots z_{k_P}$.

5. Let $\Pi_5$ be the logic program consisting of the rule
TRUE(φ) → ⊥ (23)

and the following rule

TRUE(x) ∧ EXT(x, [φ]), y) ∧ FALSE(y) → ϑ (24)

for each atom ϑ of the form same as that in Π1.

In the end, define Π′ as the union of programs Π1, . . . , Π3.

Next, we explain the intuition of this translation. Program Π1 assures that ENC will be interpreted as an encoding function on the domain and constants in C1 will be interpreted as ending flags for the encoding of atoms and positive clauses. Program Π2 defines the merging and extracting functions, and all the encoding predicates mentioned before. Based on these assumptions, program Π3 then simulates the progression of Π on encodings of positive clauses. As all the needed clauses will be derived in ω stages, this simulation is realizable. Lastly, we use program Π3 and Π4 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 Π be a disjunctive logic program. Then over infinite structures, SM(Π) is equivalent to 3∃SM(Π′), where ≺ denotes the set of constants occurring in Π but not in Π.

**Proof.** Let υ1, . . . , υ5 and τ be the set of all intensional predicates of Π1, Π2, and Π respectively. Let σ be the union of υ1, υ2 and υ(Π). Then by Splitting Lemma in [Ferraris et al., 2009] and the second-order transformation, we can conclude

SM(Π′) ≡ SM(Π1) ∧ ... ∧ SM(Π5). (25)

Let A be an infinite structure of υ(Π), and let B be any σ-expansion of A that satisfies the following conditions:

1. ENC is interpreted as the graph of an encoding function enc on A such that no element among c′ B and c B (for all P ∈ τ) belongs to the range of enc. ENC is interpreted as the complement of the graph of enc, OR B = A × A;

2. MRG and EXT are interpreted as graphs of the merging and extracting functions related to enc and c B respectively. IN, SUBC, EQU are interpreted as encoding predicates in, subc, equ related to enc and c B respectively.

For convenience, we need define some notations. Given a grounded atom φ of the form P(a1, . . . , ak) for any P ∈ τ, let (φ) be short for enc(a1, . . . , ak) c B. Given a grounded clause C in gpc(τ, A) of the form u1 ∨ . . . ∨ un where each ui is a literal, let (C) denote enc(⟨u1⟩, . . . , ⟨un⟩, c B). Given a set Σ ⊆ gpc(τ, A), let Σ denote the set {⟨C⟩ : C ∈ Σ}. Moreover, let ∆n(B) = {a ∈ B : TRUE(a) ∈ ΓΠ1 ∪ n}. By an induction on n, we can show the following claim:

**Claim 1.** For all integers n ≥ 0, (ΓΠ1 ↑ n) = ∆n(B).

Now, let B be the Π′-expansion of B that interprets TRUE as the set U∈n=0 ∆n(B), and interprets FALSE as the set of (C) for all C ∈ gpc(τ, A) such that INS(A, τ) |= ¬C.

**Claim 2.** Let C = B′. Then INS(A, τ) is a minimal model of ΓΠ1 ↑ ω iff INS(C, τ) is a minimal model of Π′.

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 SM(Π). Let B be a σ-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 Π1 and Π2. Let C be the structure B′. Then it is also clear that C is a stable model of both Π3 and Π4. On the other hand, by Proposition 2 and the assumption, INS(A, τ) should be a minimal model of ΓΠ1 ↑ ω. So, by Claim 2, INS(C, τ) is a minimal model of Π, which means that C is a stable model of Π by Proposition 1. By equation (25), C is then a stable model of Π, which means that A satisfies 3∃SM(Π′).

⇒•: Suppose A is an infinite model of 3∃SM(Π′). Then there is a Π′-expansion C of A such that C is a stable model of Π. Let B be the restrictions of C to σ. By equation (25), B is a model of the formulae SM(Π1) and SM(Π2), which implies that ENC is interpreted as an encoding function enc on A, c B and c P for all predicates P ∈ τ are interpreted as elements not in the range of enc, predicates MRG, EXT, IN, SUBC, EQU are interpreted as the corresponding functions or predicates MRG, EXT, IN, subc, equ related to enc and c B. By equation (25) again, C is a stable model of Π3 and Π4, and by Proposition 2 INS(C, υ5) is then a minimal model of ΓΠ3 ↑ ω. So, we have C = B′. According to Proposition 1 INS(C, τ) should be a minimal model of Π as C is clearly a stable model of Π. Applying Claim 2, we then have that INS(A, τ) is a minimal model of Π as A should be a stable model of Π.

**Remark 2.** Note that, given any finite domain A, there is no injective function from A × 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 ≅ INF 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 result1.

**Proposition 3.** DLP ≅ FIN NLP iff NP = coNP.

**Proof.** Let Σ2 denote the set of sentences of the form ∃∀φ, where τ and σ are finite sets of predicate variables, φ is a first-order formula. Let ESO be the set of sentences of the above form such that τ is empty. By Fagin Theorem [1974] and Stockmeyer’s characterization of the polynomial hierarchy [1977], we have that Σ2 ≅ FIN ESO iff Σ2 ≅ NP. By a routine complexity theoretical argument, we also have that

1A 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.
...Σ_F^p = NP iff NP = coNP. On the other hand, according to the proof of Theorem 6.3 in [Enter 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 ≃FIN Σ_F^1; by Lemma 1 in this section, it holds that NLP ≃FIN ESO.

Combining these results, we then have the desired proposition.

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 ESO_F^k[ψ^*] denote the set of all sentences of the form ∃τ∀x ϕ, where τ is a finite set of predicate and function variables of arity ≤ k, and ϕ is quantifier-free.

**Lemma 1.** NLP_F^k ≃FIN ESO_F^k[ψ^*] for all k > 1.

**Proof.** “≃FIN”: Let ϕ be a sentence in ESO_F^k[ψ^*]. It is clear that ϕ can be rewritten as an equivalent sentence of the form ∃τ∀x (γ_1 ∧ · · · ∧ γ_n) for some n ≥ 0, where each γ_i is a disjunction of atoms or negated atoms, and σ a finite set of functions or predicates of arity ≤ k. Let P be a logic program consisting of the rule γ_i → ⊥ for each 1 ≤ i ≤ n, where γ_i is obtained from γ_i by substituting ϑ for each negated atom ¬ϑ, substituting ϑ for each atom ϑ, and substituting ∧ for ∨. Obviously, ∃σSM(II) is in NLP_F^k and equivalent to ϕ.

“≃FIN”: Let ∃σSM(II) be a formula in NLP_F^k such that P is a normal logic program. Without loss of generality, we assume the head of each rule in P is of the form P(¯x) for some integer l ≥ 0 and l-ary intensional predicate P of P. Let τ and σ be the sets of all intensional predicates and atoms of P respectively. Let c = k·|τ|+1. For each λ ∈ S, suppose γ_1, ..., γ_n list all rules in P whose heads are λ. Suppose

\[ γ_i = ζ_i^1 ∧ ζ_i^2 ∧ · · · ∧ ζ_i^m_i → λ \]

where ζ_i^1, ..., ζ_i^m_i are intensional atoms, ζ_i^1 is a conjunction of literals that are not intensional atoms of P, m_i ≥ 0, and ỹ_i is the tuple of all individual variables occurring in γ_i but not in λ. Now, we then define ϕ as the conjunction of formulae

\[ (ζ_i^1 ∧ ζ_i^2 ∧ · · · ∧ ζ_i^m_i) \lor λ → DRVBL(λ) \]

for all 1 ≤ i ≤ n, and define ϕ as the formula

\[ DRVBL(λ) → λ \land \land_{i=1}^{n} \exists ỹ_i (ζ_i^1 \land \land_{j=1}^{m_i} DRVBL(ζ_i^j, λ) \]
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 $\Pi$ 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 $\forall \sigma \varphi \equiv_{\text{FIN}} \exists \sigma \text{SM}(\Pi)$, where $\varsigma$ is the set of all predicates in $\nu(\Pi)$ but not in $\nu(\forall \sigma \varphi)$. \hfill $\square$

With these two lemmas, we then have the following result:

**Theorem 2.** $\text{DLP}^k \not\leq_{\text{FIN}} \text{NLP}^{2k-1}$ for all $k > 1$.

*Proof. (Sketch)* Let $n$ be an integer $\geq 1$ and $\nu$ the vocabulary consisting of only an $n$-ary predicate $P$. Define $\text{PARITY}^n$ to be the class of $\nu$-structures in which $P$ is interpreted as the least and the maximal elements in the order defined by $S$ respectively.

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:

$(Y(\bar{0}) \leftrightarrow P(\bar{x}, \bar{0})) \land \forall \bar{u} \forall \bar{v}(\bar{u} \rightarrow P(\bar{x}, \bar{v})) \rightarrow \neg Y(\bar{m})$

where $\psi \oplus \chi$ denotes the formula $\psi \rightarrow \neg \chi$, $\oplus$ 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 “$X(\bar{u})$ is true iff the cardinality of the set $\{b : P(\bar{b}, \bar{u})\}$ is odd”. Let $\varphi_3$ be the following formula:

$(X(\bar{0}) \leftrightarrow Y(\bar{0})) \land \forall \bar{u} \forall \bar{v}(\bar{u} \rightarrow (X(\bar{v}) \leftrightarrow Y(\bar{u})) \rightarrow \neg Y(\bar{m}))$

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

$\exists m \exists \exists S[\varphi_1 \land \exists X \forall Y \forall \bar{x}(\varphi_2 \land \varphi_3)]$

It is not difficult to check that $\varphi$ defines $\text{PARITY}^{2k}$ over finite structures. By Lemma 2 there is a logic program $\Pi_{\varphi}$ such that $\exists X \forall Y \forall \bar{x}(\varphi_2 \land \varphi_3) \equiv_{\text{FIN}} \exists \sigma \text{SM}(\Pi_{\varphi}) \in \text{DLP}^k$, 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 $\Pi_{\phi}$ involving only predicates of arity $\leq 2$. Hence, we have that $\varphi \equiv_{\text{FIN}} \exists m \exists \exists S[\exists \sigma \text{SM}(\Pi_{\phi}) \land \exists \exists \text{SM}(\Pi_{\varphi})]$.

Let $\Pi$ be $\Pi_{\phi} \sqcup \Pi_{\varphi}$. By Splitting Lemma [Eiter et al., 2009], $\varphi \equiv_{\text{FIN}} \exists m \exists \exists S[\exists \sigma \text{SM}(\Pi_{\phi}) \land \exists \exists \text{SM}(\Pi_{\varphi})]$. Let $\Pi'$ be the program obtained from $\Pi$ by simulating individual constants 0 and $m$ by unary predicates. This is then the desired program.

Next, we show that $\text{PARITY}^{2k}$ is undefinable in $\text{NLP}^{2k-1}$. By Lemma 1, it suffices to show that $\text{PARITY}^{2k}$ is undefinable in $\text{ESO}^{2k-1}$. Towards a contradiction, assume that there is a sentence $\psi$ in this class such that finite models of $\psi$ are exactly the structures in $\text{PARITY}^{2k}$. 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 $\text{ESO}^{4k-2}$ to define $\text{PARITY}^{4k}$. However, according to Theorem 2.1 of [Ajtai, 1983], this is impossible since $\psi_0$ has an equivalent in $\text{ESO}^{4k-1}$, i.e., the set of formulae in $\text{ESO}^{4k-1}$ without function variables. This then completes the proof. \hfill $\square$

5. **Arbitrary Structures**

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

**Theorem 3.** $\text{DLP} \simeq_{\text{FIN}} \text{NLP}$ iff $\text{DLP} \simeq_{\text{FIN}} \text{NLP}$.

*Proof. (Sketch)* It is trivial from left to right. Now we show the converse direction. Assume $\text{DLP} \simeq_{\text{FIN}} \text{NLP}$. Then, for each disjunctive logic program $\Pi$, there should be a normal logic program $\Pi'$ such that $\text{SM}(\Pi) \equiv_{\text{FIN}} \exists \sigma \text{SM}(\Pi')$, where $\sigma$ is the set of all predicates and functions occurring in $\Pi'$ but not in $\Pi$. To show $\text{DLP} \simeq_{\text{NL}}$, 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. $\neg \text{ARC}(x, y) \rightarrow \text{ARC}(x, y)$, 4. $\neg \text{OK}_a(x) \rightarrow \text{OK}_a(x)$, 2. $\neg \text{ARC}(x, y) \rightarrow \text{ARC}(x, y)$, 5. $\text{ARC}(x, y) \rightarrow \text{inf}$, 3. $\text{ARC}(x, y) \rightarrow \text{OK}_a(x)$, 6. $\inf \rightarrow \fin$.

and rule $\text{ARC}(x, y) \land \text{ARC}(y, z) \rightarrow \text{ARC}(x, z)$. This program sets flag $\text{inf}$ to be true if the intended model is infinite. When the intended model is finite, we use program $\Pi_{\text{fin}}$ to set flag $\text{fin}$, which is obtained from $\pi_1$ in Lemma 6.4 of [Eiter et al., 1997] by substituting $\text{FIN}$ for $\text{ORDER}$ and by applying the shift operation in Section 4.5 of [Ben-Eliyahu and Dechter, 1994]. Let $\Pi_{\varphi}^\circ (\Pi')$ be the program obtained from $\Pi'$ ( $\Pi'$, respectively) by adding $\text{FIN}$ ($\inf$, respectively) to the body of each rule as a conjunct. Let $\Pi'$ be the union of $\Pi_{\varphi}^\circ, \Pi', \Pi_{\text{inf}}$ and $\Pi_{\text{fin}}$. We can show $\text{SM}(\Pi') \equiv_{\exists \sigma \text{SM}(\Pi')}$, where $\sigma$ is the set of all constants occurring in $\Pi'$ but not in $\Pi$. \hfill $\square$

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 surprise 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 NL over finite structures.

From Theorem 3 and Proposition 3, we immediately have:

**Corollary 2.** $\text{DLP} \simeq_{\text{FIN}} \text{NLP}$ if $\text{FIN} = \text{coNP}$. 

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. Datalog vs first-order logic. *Journal of Computer and System Sciences*, 49:562–588, 1994.

[Ajtai, 1983] M. Ajtai. Σ₁¹-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[Barwise, 1976] J. Barwise. *Admissible Sets and Structures*. Springer, Berlin, 1976.

[Ben-Eliyahu and Dechter, 1994] R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12(1–2):53–87, 1994.

[Cabalar et al., 2005] P. Cabalar, D. Pearce, and A. Valverde. Reducing propositional theories in equilibrium logic to logic programs. In *Proc. EPIA*, pages 4–17, 2005.

[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, 2001.

[Durand et al., 1998] A. Durand, C. Lautemann, and T. Schwentick. Subclasses of binary NP. *Journal of Logic and Computation*, 8(2):189–207, 1998.

[Ebbinghaus and Flum, 1999] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, New York, 2 edition, 1999.

[Eiter and Gottlob, 1997] T. Eiter and G. Gottlob. Expressiveness of stable model semantics for disjunctive logic programs with functions. *The Journal of Logic Programming*, 33:167–178, 1997.

[Eiter et al., 1997] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, 22:364–418, 1997.

[Fagin, 1974] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation*, 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 stable models. In *Proc. IJCAI*, pages 797–803, 2009.

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

[Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. ICLP/SLP*, pages 1070–1080, 1988.

[Leivant, 1989] D. Leivant. Descriptive characterizations of computational complexity. *Journal of Computer and System Sciences*, 39:51–83, 1989.

[Lin and Zhou, 2011] F. Lin and Y. Zhou. From answer set logic programming to circumscription via logic of GK. *Artificial Intelligence*, 175(1):264–277, 2011.

[Lobo et al., 1992] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. The MIT Press, Cambridge, 1992.

[Moschovakis, 1974] Y. Moschovakis. *Elementary Induction 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 Sciences*, 51(1):64–86, 1995.

[Stockmeyer, 1977] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Zhang and Zhang, 2013] H. Zhang and Y. Zhang. First-order expressibility and boundedness of disjunctive logic programs. *To appear in Proc. IJCAI*, 2013.