Ordered Completion for Logic Programs with Aggregates

Vernon Asuncion^a, Yin Chen^b, Yan Zhang^a, Yi Zhou^a

 ^a Artificial Intelligence Research Group (AIRG), School of Computing, Engineering and Mathematics, University of Western Sydney, Australia
 ^b Department of Computer Science, South China Normal University, Guangzhou, China

Abstract

We consider the problem of translating first-order answer set programs with aggregates into first-order sentences with the same type of aggregates. In particular, we show that, on finite structures, normal logic programs with convex aggregates, which cover both monotone and antimonotone aggregates as well as the aggregates appearing in most benchmark programs, can always be captured in firstorder logic with the same type of aggregates by introducing auxiliary predicates. More precisely, we prove that every finite stable model of a normal program with convex aggregates is corresponding to a classical model of its enhanced ordered completion. This translation then suggests an alternative way for computing the stable models of such kind of programs. We report some experimental results, which demonstrate that our solver GROCv2 is comparable to the state-of-the-art answer set solvers. We futher show that convex aggregates form a maximal class for this purpose. That is, we can always construct a normal logic program under any given non-convex aggregate context and prove that it can never be translated into first-order sentences with the same type of aggregates unless NP = coNP.

1. Introduction

In this paper, we consider to translate first-order Answer Set Programming (ASP), a predominant declarative programming paradigm in the area of knowledge representation and logic programming [3, 20, 24, 25], into first-order logic. Work in this direction is not only of theoretical interests but also of practical relevances as it suggests an alternative way to implement ASP.

Recently, Asuncion et al. [2] proposed a notion of ordered completion (a firstorder sentence with some extra predicates) for first-order normal logic programs, and showed that the stable models of a normal program are exactly corresponding to the classical models of its ordered completion on finite structures. Interestingly, there is no such translation on arbitrary structures nor prohibiting extra predicates. Based on this translation, they developed a new ASP solver, which first translates a program to its ordered completion, then grounds this first-order sentence, and finally calls an SMT solver. This is significantly different from previous ASP solvers, which ground the first-order programs directly. A first implementation shows that this new solver is promising as it performs relatively good for the Hamiltonian Circuit program, particularly on big instances [2].

However, their work cannot handle aggregates, a very important building block for modern Answer Set Programming. The reason why aggregates are crucial in answer set solving is of twofold. Firstly, they enhance the expressive power of ASP, and often they can simplify the representation task. For many applications, one can write a simpler and more elegant logic program by using aggregates, for instance, the job scheduling program [28]. Secondly and more importantly, aggregates can improve the efficiency of ASP solving [19]. Normally, the program using aggregates can be solved much faster [12].

In this paper, we consider the problem of extending ordered completion for programs with aggregates. This is a challenging task as some programs with aggregates are expressive enough to capture disjunctive logic programming (see in [16]), thus can never be captured in first-order logic with the same type of aggregates providing some general assumptions in the computational complexity theory (see Proposition 6 in [2]).

Hence, an important task is to draw a boundary between the normal programs with aggregates that can be captured in first-order logic with the same type of aggregates and those programs that cannot. For this purpose, we extend the notion of convex constraints proposed by Liu and Truszczyński [23] into first-order convex aggregates. We show that the class of convex aggregates is exactly the boundary we need in the sense that

- First-order normal logic programs with convex aggregates can always be captured in first-order logic with the same type of aggregates on finite structures. More precisely, we extend the notion of ordered completion for first-order normal logic programs with convex aggregates, and show that every stable model of such a program is corresponding to a classical model of its enhanced ordered completion.
- Given any non-convex aggregate context, there exists a normal program under this context such that it can never be translated into first-order sentences

with the same type of aggregates unless NP = coNP.

In fact, the class of convex aggregates is expressive enough to capture both monotone and antimonotone aggregates [23] as well as the aggregates appearing in most benchmark programs [5]. Therefore, based on our theoretical results, we are able to develop an alternative ASP solver for first-order normal programs with convex aggregates. Following this idea, we implement a new ASP solver GROCv2. Our experimental results demonstrate that GROCv2 is comparable to the state-of-the-art ASP solvers.

The paper is organized as follows. Section 2 reviews basic concepts and notations that we will need through out the paper. Section 3 presents the ordered completion for logic programs with aggregates, and proves the main theorems. Section 4 introduces the implementation of the ASP solver GROCv2, and reports some experimental results. Finally, Sections 5 and 6 discuss some related work and draw our conclusions respectively. We leave the very long proofs of some theorems to Appendix A for a more fluent reading.

2. Preliminaries

We consider a second-order language without functions but with equality =. A signature contains a finite set of constants and a finite set of predicates. A term is either a variable or a constant. A standard atom is an expression P(t), where P is a predicate and t is a tuple of terms which matches the arity of P. An equality atom is an expression $t_1 = t_2$, where t_1 and t_2 are terms.

A multiset (also called a bag) is a pair $M = (M_s, M_f)$, where M_s is a set and M_f is a function, called the multiplicity function, from M_s to \mathbb{N} , i.e., the set of positive integers $\{1, 2, 3, ...\}$. A multiset (M_s, M_f) is finite if M_s is finite. Let M and M' be two multisets. We denote by $M \subseteq M'$ if $M_s \subseteq M'_s$ and for all elements $a \in M_s$, $M_f(a) \leq M'_f(a)$. We write M = M' if $M \subseteq M'$ and $M' \subseteq M$. For convenience, a multiset M, where $M_s = \{a_1, \ldots, a_n\}$ and $M_f(a_i) = c_i(1 \leq i \leq n)$, is also denoted as $\{\{a_1, \ldots, a_1, \ldots, a_i, \ldots, a_i, \ldots, a_n, \ldots, a_n\}\}$. The order of the elements is irrelevant. For example, $\{\{a, a, b, c\}\}$ is the multiset M, where $M_s = \{a, b, c\}$ and $M_f(a) = 2$, $M_f(b) = M_f(c) = 1$.

2.1. The Syntax of Aggregates

Aggregate is a crucial auxiliary building block for answer set programming [12, 13, 16, 19, 22, 23, 28]. We first define the syntax of aggregates in the first-

order case. We assume a set of aggregate symbols \mathcal{AG} and a (fixed) set of comparison operators on numbers $\mathcal{CO} = \{<, \leq, =, \neq, \geq, >\}$.

Definition 1. An aggregate atom δ is an expression of the form

$$OP\langle \mathbf{v}: \exists \mathbf{w} Q_1(\mathbf{y}_1) \land \dots \land Q_s(\mathbf{y}_s) \land \neg R_1(\mathbf{z}_1) \land \dots \land \neg R_t(\mathbf{z}_t) \rangle \preceq t, ^1 \quad (1)$$

where

- OP $\in AG$ is an aggregate symbol,
- $Q_i(\mathbf{y}_i)$ $(1 \le i \le s)$ and $R_j(\mathbf{z}_j)$ $(1 \le j \le t)$ are standard atoms or equality atoms. In addition,

$$Q_1(\mathbf{y}_1) \wedge \dots \wedge Q_s(\mathbf{y}_s) \wedge \neg R_1(\mathbf{z}_1) \wedge \dots \wedge \neg R_t(\mathbf{z}_t)$$
(2)

is called the body of δ , denoted by $Bd(\delta)$,

- **v** and **w** are tuples of variables mentioned in (2), and $\mathbf{v} \cap \mathbf{w} = \emptyset$,
- $\leq \in CO$ is a comparison operator on numbers,
- t is a term, and we assume that variables occurring in t are not in $\mathbf{v} \cup \mathbf{w}$.

For convenience, we use $Ps(\delta)$ and $Ng(\delta)$ to denote the sets $\{Q_1(\mathbf{y}_1), \ldots, Q_s(\mathbf{y}_s)\}$ and $\{R_1(\mathbf{z}_1), \ldots, R_t(\mathbf{z}_t)\}$ respectively. Given an aggregate atom δ of the form (1), a variable in δ is a *free* variable if it is not a variable in $\mathbf{v} \cup \mathbf{w}$.

Example 1. Let SUM and CARD be aggregate symbols in \mathcal{AG} . The following are two aggregate atoms:

$$\operatorname{CARD}\langle x: P(x) \rangle = 2, \quad \operatorname{SUM}\langle x: P(x) \rangle \le 5.$$

Intuitively, they are equivalent to the weight constraints

$$2\{p(X)\}2, \quad \{p(X) = X\}5$$

in smodels [29], and the aggregate atoms

$$\#count\{X: p(X)\} = 2, \quad \#sum\{X: p(X)\} \le 5$$

in DLV [14] and ASP-Core-2. 2 \Box

¹Here, w could be empty. In this case, (1) is simply written as $OP(\mathbf{v} : Bd(\delta)) \leq t$.

²ASP-Core-2 is currently the standard ASP input language, available at https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.03b.pdf.

An *atom* is either an equality atom, or a standard atom, or an aggregate atom. A *first-order formula with aggregates* (or *formula* for short) is built from atoms and logical connectives as usual. A formula without aggregate atom is called a *classical* formula in this paper. The *free* variable of a formula is defined as usual. We use $free(\phi)$ to denote the set of free variables of a formula ϕ .

2.2. The Semantics for First-Order Logic with Aggregates

As aggregate is an extra building block, we need to extend the standard semantics for classical first-order logic for incorporating aggregates, in which aggregates are considered as predefined function symbols.

Definition 2. An aggregate context AC is a tuple of the form:

$$(\mathcal{AC}_{ag}, \mathcal{AC}_{co}, \mathcal{AC}_{num}, \operatorname{OP}_{1}^{\mathcal{AC}}, \dots, \operatorname{OP}_{n}^{\mathcal{AC}})$$

where

- $\mathcal{AC}_{ag} = \{ OP_1, \dots, OP_n \}$ is a subset of \mathcal{AG} ;
- $\mathcal{AC}_{co} \subseteq \mathcal{CO}$ is a set of comparison operators;
- $\mathcal{AC}_{num} \subseteq \mathbb{Z}$ is a set of numbers;
- for every aggregate symbol $OP_i \in \mathcal{AG}$ $(1 \leq i \leq n)$, there is a partial function $OP_i^{\mathcal{AC}}$ from the set of multisets over tuples on \mathcal{AC}_{num} to \mathcal{AC}_{num} .

Example 2. Consider an aggregate context \mathcal{AC}_1 , where

- $\mathcal{AC}_{1ag} = \{CARD, SUM, MIN, MAX\}, \text{ for cardinality, sum, minimum, and maximum respectively;}$
- $\mathcal{AC}_{1co} = \{<, \leq, =, \geq, >\};$
- $\mathcal{AC}_{1num} = \mathbb{N};$
- given a multiset $M = \{\{\mathbf{a}_1, \dots, \mathbf{a}_k\}\}$, where each $\mathbf{a}_i, (1 \le i \le k)$ is a tuple of numbers,
 - CARD(M) is defined as the cardinality of M;
 - $\operatorname{SUM}(M)$ is defined as $\sum_{i=1}^k \mathbf{a}_i[1];\,{}^3$

³Given a tuple **a**, $\mathbf{a}[i]$ denotes the *i*-th component of **a**, where $1 \le i \le |\mathbf{a}|$.

- MIN(M) is undefined if $M = \emptyset$, and the minimum of $\mathbf{a}_i[1]$, $(1 \le i \le k)$ if $M \ne \emptyset$;
- MAX(M) is undefined if $M = \emptyset$, and the maximum of $\mathbf{a}_i[1]$, $(1 \le i \le k)$ if $M \ne \emptyset$.

Furthermore, we may also define the following two aggregate contexts:

- Let \mathcal{AC}_2 be the aggregate context which is identical to \mathcal{AC}_1 except that $\mathcal{AC}_{2co} = \mathcal{CO} = \mathcal{AC}_{1co} \cup \{\neq\}.$
- Let \mathcal{AC}_3 be the aggregate context which is identical to \mathcal{AC}_2 except that $\mathcal{AC}_{3num} = \mathbb{Z}$.

The aggregate context \mathcal{AC}_1 in Example 2 presents the most common aggregate functions in the current ASP solvers. However, the following example shows that it is possible to define more general aggregates in our setting theoretically.

Example 3. Consider an aggregate context \mathcal{AC}_4 , where

- $\mathcal{AC}_{4ag} = \{\text{SUMALL}, \text{SAT}\}, \text{ where SUMALL and SAT are two aggregate symbols in } \mathcal{AG};$
- $\mathcal{AC}_{4co} = \{<, \leq, =, \geq, >\};$
- $\mathcal{AC}_{4num} = \mathbb{N};$
- given a multiset $M = \{\{\mathbf{a}_1, \dots, \mathbf{a}_k\}\}$, where each $\mathbf{a}_i, (1 \le i \le k)$ is a tuple of numbers,
 - SUMALL(M) is defined as

$$\sum_{i=1}^k \sum_{j=1}^{|\mathbf{a}_i|} \mathbf{a}_i[j];$$

- SAT(M) is 1 if φ_M is satisfiable in propositional logic and 0 otherwise, where φ_M is the formula

$$\bigwedge_{i=1}^k \left(\bigvee_{a\in\mathbf{a}_i} x_a\right),\,$$

and $\{x_1, x_2, \ldots\}$ is a set of propositional atoms.

Definition 3. Given a signature σ , an extended structure \mathcal{A} of σ is a tuple

$$(A, f_{\mathcal{A}}, c_1^{\mathcal{A}}, \dots, c_l^{\mathcal{A}}, P_1^{\mathcal{A}}, \dots, P_m^{\mathcal{A}}),$$
(3)

where

- A is the domain of \mathcal{A} , denoted by $Dom(\mathcal{A})$,
- f_A is a total function from A to \mathbb{Z} ,
- $c_i^{\mathcal{A}}$, $(1 \le i \le l)$ is the interpretation for constant c_i ;
- $P_j^{\mathcal{A}}$, $(1 \le j \le m)$ is the interpretation for predicate P_j .

An extended structure is *finite* if its domain is finite. Note that the only difference between the extended structure defined above and the structure in first-order logic is the partial function f_A which maps domain elements to numbers. This enables us to freely use variables and constants in aggregate functions. Since aggregate functions such as SUM are defined over numbers but not over arbitrary domain elements, we use a two step approach to interpret an aggregate atom. First, using the standard first-order semantics, we map the terms (i.e., constants and variables) into domain elements. Then, we use the function f_A to further map these domain elements into numbers so that the aggregate atoms are well defined. We extend the function f_A for a tuple of domain elements and for a multiset. By $f_A(\mathbf{c})$, we denote the tuple $(f_A(c_1), \ldots, f_A(c_n))$, where $\mathbf{c} = (c_1, \ldots, c_n)$ is a tuple. By $f_A(A)$, we denote the multiset {{ $f_A(a_1), \ldots, f_A(a_k)$ }}, where $A = {{a_1, \ldots, a_k}}$ is a multiset. In the following, an extended structure is simply called a structure when it is clear from the context.

Let \mathcal{A} be a structure. An *assignment* is an expression of the form \mathbf{x}/\mathbf{a} , where \mathbf{x} is a tuple of distinct variables and $\mathbf{a} \in Dom(\mathcal{A})^{|\mathbf{x}|}$ is a tuple of domain elements. Assignments can be extended for terms. Let \mathbf{t} be a tuple of terms, in which the variables are from \mathbf{x} . We use $\mathbf{t}[\mathbf{x}/\mathbf{a}]$ to denote the tuple of domain elements by simultaneously replacing the variables in \mathbf{x} via the assignment \mathbf{x}/\mathbf{a} and constant c by $c^{\mathcal{A}}$.

Under an aggregate context \mathcal{AC} , the satisfaction relation between a structure \mathcal{A} and a formula $\varphi(\mathbf{x})$ (with aggregate atoms) together with an assignment \mathbf{x}/\mathbf{a} is defined recursively as follows:

- If φ is $P(\mathbf{t})$, then $\mathcal{A} \models \varphi[\mathbf{x}/\mathbf{a}]$ iff $\mathbf{t}[\mathbf{x}/\mathbf{a}] \in P^{\mathcal{A}}$;
- If φ is $\mathbf{t}_1 = \mathbf{t}_2$, then $\mathcal{A} \models \varphi[\mathbf{x}/\mathbf{a}]$ iff $\mathbf{t}_1[\mathbf{x}/\mathbf{a}] = \mathbf{t}_2[\mathbf{x}/\mathbf{a}]$;
- If φ is $\neg \psi$, then $\mathcal{A} \models \varphi[\mathbf{x}/\mathbf{a}]$ iff $\mathcal{A} \not\models \psi[\mathbf{x}/\mathbf{a}]$;
- If φ is $\psi \wedge \xi$, then $\mathcal{A} \models \varphi[\mathbf{x}/\mathbf{a}]$ iff $\mathcal{A} \models \psi[\mathbf{x}/\mathbf{a}]$ and $\mathcal{A} \models \xi[\mathbf{x}/\mathbf{a}]$;
- If φ is $\forall y\psi$, then $\mathcal{A} \models \varphi[\mathbf{x}/\mathbf{a}]$ iff for all $b \in Dom(\mathcal{A}), \mathcal{A} \models \psi[\mathbf{x}y/\mathbf{a}b]$;
- Finally, if φ is an aggregate atom δ of the form (1) in the aggregate context *AC*, then *A* ⊨ φ[**x**/**a**] iff
 - 1. OP $\in \mathcal{AC}_{ag}$ and $\preceq \in \mathcal{AC}_{co}$;
 - 2. $t[\mathbf{x}/\mathbf{a}]$ is in the domain of $f_{\mathcal{A}}$;
 - 3. the multiset

$$M = \{ \{ f_{\mathcal{A}}(\mathbf{c}) \mid \mathbf{c} \in M', f_{\mathcal{A}}(\mathbf{c}) \text{ is defined} \} \}$$

is in the domain of $OP^{\mathcal{AC}}$, where M' is the set

$$\{\mathbf{c} \mid \mathcal{A} \models Bd(\delta)[\mathbf{xwv/abc}], \mathbf{b} \in Dom(\mathcal{A})^{|\mathbf{w}|}, \mathbf{c} \in Dom(\mathcal{A})^{|\mathbf{v}|}\};$$

4. $OP^{\mathcal{AC}}(\mathcal{A}) \leq f_{\mathcal{A}}(t[\mathbf{y/a}]);$

Example 4. Consider the aggregate context \mathcal{AC}_2 and a signature σ with only one predicate P. For any structure \mathcal{A} with the domain $\{a, b\}$ and $f_{\mathcal{A}}(a) = 2$ and $f_{\mathcal{A}}(b) = 3$, we have

$$\mathcal{A} \models \mathrm{SUM}\langle x : P(x) \rangle \le 5,$$

and

$$\mathcal{A} \models \text{SUM}\langle x : P(x) \rangle \neq 2 \equiv (P(a) \to P(b)).$$

Let \mathcal{AC} be an aggregate context. An aggregate $OP \in \mathcal{AC}_{ag}$ is *polynomial* if the problem of checking $OP^{\mathcal{AC}}(M) \preceq n$ can be done in polynomial time with respect to |M| for every multiset M and number n, where |M|, the length of M, is defined as the sum of the length of all tuples in M [16]. \mathcal{AC} is *polynomial* if all aggregates in it are polynomial. It can be verified that all three aggregate contexts in Example 2 are polynomial, while \mathcal{AC}_4 in Example 3 is not polynomial unless P = NP.

2.3. The Stable Model Semantics for Normal Logic Programs with Aggregates

We now propose the stable model semantics for first-order normal logic programs with aggregates. A *normal program with aggregates* (or *program* for short) is a finite set of *rules* of the form

$$\alpha \leftarrow \beta_1, \dots, \beta_l, \operatorname{not} \gamma_1, \dots, \operatorname{not} \gamma_m, \tag{4}$$

where α is either a standard atom or \perp , β_i , $(1 \le i \le l)$ and γ_j , $(1 \le j \le m)$ are atoms.

A rule of the form (4) is called a *constraint* if α is \perp . By Π^{\perp} , we denote the set of constraints in program Π . Let r be a rule of the form (4). We call α the head of r and $\{\beta_1, \ldots, \beta_l, \operatorname{not} \gamma_1, \ldots, \operatorname{not} \gamma_m\}$ the body of r. A variable is called a *local variable* of r if it occurs in a standard atom, equality atom, or occurs freely in an aggregate atom in the body of r but it does not occur in the head of r. For convenience, we use Head(r) and Body(r) to denote α and $\beta_1 \wedge \cdots \wedge \beta_l \wedge$ $\neg \gamma_1 \wedge \cdots \wedge \neg \gamma_m$, respectively. We also use Pos(r) and PosAgg(r) to denote the set of positive atoms and the set of positive aggregate atoms from the body of r, respectively. By \hat{r} , we denote the universal closure of the formula

$$\beta_1 \wedge \cdots \wedge \beta_l \wedge \neg \gamma_1 \wedge \cdots \wedge \neg \gamma_m \to \alpha.$$

By $\widehat{\Pi}$, we denote the formula $\bigwedge_{r \in \Pi} \widehat{r}$.

The signature of a program Π , denoted by $\tau(\Pi)$, consists of all constants and predicate symbols occurring in Π . A predicate in a program is said to be *intensional* if it occurs in the head of some rule in the program, and *extensional* otherwise. We use $\mathcal{P}_{int}(\Pi)$ to denote the set of all intensional predicates of Π .

Example 5. Consider the following program Π_1 with an aggregate atom

$$r_1: P(x) \leftarrow R_1(x), \tag{5}$$

$$r_2: P(x) \leftarrow P(y), R_2(y, x), \tag{6}$$

$$r_3: P(x) \leftarrow \mathsf{SUM}\langle y: P(y) \land R_3(x, y) \rangle > 3.$$
(7)

Here, SUM is an aggregate symbol, P is intensional and R_1, R_2, R_3 are extensional. \Box

The stable model semantics of a program is defined by a second-order sentence. We first introduce some notations. Let P and Q be two predicates or predicate variables of the same arity. We use P < Q to denote the formula

$$\forall \mathbf{x}(P(\mathbf{x}) \to Q(\mathbf{x})) \land \neg \forall \mathbf{x}(Q(\mathbf{x}) \to P(\mathbf{x})).$$

Let $\mathbf{P} = P_1 \cdots P_k$ and $\mathbf{P}' = P'_1 \cdots P'_k$ be two tuples of predicates or predicate variables such that for every $1 \le i \le k$, P_i and P'_i have the same arity. We use $\mathbf{P} < \mathbf{P}'$ to denote the formula

$$\bigwedge_{i=1}^{k} \forall \mathbf{x} (P_i(\mathbf{x}) \to P'_i(\mathbf{x})) \land \neg \bigwedge_{i=1}^{k} \forall \mathbf{x} (P'_i(\mathbf{x}) \to P_i(\mathbf{x})).$$

Let Π be a program such that $\mathcal{P}_{int}(\Pi) = \{P_1, \ldots, P_n\}$. Let $\mathbf{U} = U_1 \ldots U_n$ be a tuple of new predicates such that each U_i , $(1 \le i \le n)$ matches the arity of P_i . Given an atom ρ , ρ^* is defined as

- ρ itself, if ρ is an equality atom or an atom of the form $P(\mathbf{x})$, where P is an extensional predicate;
- $U_i(\mathbf{x})$, if ρ is an atom of the form $P_i(\mathbf{x})$, where P_i is an intensional predicate;
- (OP⟨v : ∃wBd(δ)*⟩ ≤ t) ∧ (OP⟨v : ∃wBd(δ)⟩ ≤ t), if ρ is an aggregate atom δ of the form (1), where

$$Bd(\delta)^* = Q_1^*(\mathbf{y}_1) \wedge \dots \wedge Q_s^*(\mathbf{y}_s) \wedge \neg R_1(\mathbf{z}_1) \wedge \dots \wedge \neg R_t(\mathbf{z}_t).$$

Let r be a rule of the form (4) which is not a constraint. We use $Body(r)^*$ to denote the formula

$$\beta_1^* \wedge \cdots \wedge \beta_l^* \wedge \neg \gamma_1 \wedge \cdots \wedge \neg \gamma_m,$$

and r^* the universal closure of the formula

$$Body(r)^* \to Head(r)^*.$$

Finally, by $SM(\Pi)$, we denote the following second-order sentence

$$\bigwedge_{r \in \Pi} \widehat{r} \wedge \neg \exists \mathbf{U} (\mathbf{U} < \mathbf{P} \wedge \bigwedge_{r \in \Pi \setminus \Pi^{\perp}} r^*).$$
(8)

Definition 4 (stable model). Let Π be a program and AC an aggregate context. A structure A on $\tau(\Pi)$ is said to be a stable model of Π if A is a model of $SM(\Pi)$. For normal programs without aggregates, the definition of $SM(\Pi)$ is exactly the same as those presented in [2] and [17]. For the stable model semantics of propositional programs with aggregates, there are several alternative definitions [4, 8, 10, 13, 16, 18, 27, 30]. The Ferraris' semantics and the FLP semantics have recently been extended into the first-order case [4, 18, 22]. Our definition of $SM(\Pi)$ can be considered as another first-order extension of the Ferraris' semantics [16, 18]. Nevertheless, if the aggregate atoms only occur in the positive bodies of rules and the bodies of these aggregates contain no negative atoms (this is actually the case in many benchmark programs), these two semantics coincide.

Example 6. (Example 5 continued) Consider the aggregate context \mathcal{AC}_1 in Example 2 and three structures $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ on $\tau(\Pi_1)$ such that for i = 1, 2, 3,

$$Dom(\mathcal{M}_i) = \{a, b, c, d\},\$$

$$f_{\mathcal{M}_i}(a) = 1, f_{\mathcal{M}_i}(b) = 2, f_{\mathcal{M}_i}(c) = 3, f_{\mathcal{M}_i}(d) = 4,\$$

$$R_1^{\mathcal{M}_i} = \{b\}, R_2^{\mathcal{M}_i} = \{aa, bc\}, R_3^{\mathcal{M}_i} = \{aa, ac, db, dc\},\$$

and

$$P^{\mathcal{M}_1} = \{b, c\}, P^{\mathcal{M}_2} = \{b, c, d\}, P^{\mathcal{M}_3} = \{a, b, c, d\}.$$

- M₁ is not a stable model of Π₁ since it is not a model of r̂₃. To see this, note that M₁ ⊨ SUM⟨y : P(y) ∧ R₃(x, y)[x/d] and M₁ ⊭ P(x)[x/d].
- Both M₂ and M₃ are models of Λ_{r∈Π1} r̂, but M₃ is not a stable model of Π₁, as M₃ is not a model of the second-order sentence (8). Indeed, let U' = {b, c, d} ⊂ P^{M₃}. Then, we have M₃ ⊨ U < P[U/U'] and M₃ ⊨ Λ_{r∈Π1} r^{*}[U/U'], so that M₃ ⋡ ¬∃U(U < P ∧ Λ_{r∈Π1} r^{*}).
- It can be verified that M₂ is the only stable model of Π₁. Roughly speaking, M₂ is obtained as follows:
 - P(b) is derived by the fact $R_1(b)$ and the rule r_1 ;
 - P(c) is derived by the facts P(b), $R_2(b, c)$ and the rule r_3 ;
 - P(d) is derived by the rule r_3 by noticing that $\mathcal{M}_3 \models \text{SUM}\langle y : P(y) \land R_3(x,y) \rangle > 3[x/d];$
 - P(a) cannot be derived by either r_2 or r_3 without assuming P(a) itself.

г		
L		
L		
L	 	

3. Ordered Completion for Normal Logic Programs with Aggregates

Ordered completion, introduced by Asuncion et al. [2] for normal logic program without aggregates, is a modification of Clark's completion [7] by adding some auxiliary predicates to capture the derivation order during the program's evaluation. It is showed that, on finite structures, the stable models of a normal logic program are exactly corresponding to the classical models of its ordered completion.

From a theoretical point of view, ordered completion makes an important contribution on understanding first-order answer set programming. Firstly, it shows that the stable model semantics can be captured by Clark's completion plus derivation order. Secondly, it clarifies the relationship between first-order normal ASP and first-order logic. More precisely, every normal answer set program can be captured by a first-order sentence with some new predicates on finite structures. Interestingly, this result does not hold on arbitrary structures nor if no new predicate symbol is used [2].

In addition, ordered completion is relevant from a practical point of view. It initiates a new direction of developing an alternative ASP solver by first translating a normal logic program to its ordered completion, then working on finding a model of this first-order sentence. A first implementation shows that this new approach is promising as it performs good on the Hamiltonian Circuit program [25], especially on very large instances. However, ordered completion can hardly be used beyond the Hamiltonian Circuit program because it cannot handle aggregates - a crucial building block widely used in many applications in ASP.

Extending ordered completion for dealing with aggregates is not an easy task. First of all, the aggregate atoms in a logic program are highly interacted with the rest parts. Hence, a naive extension of ordered completion by simply treating aggregate atoms as extensional atoms would not work.

Another observation is from a computational complexity point of view. In the propositional case, it is shown that checking the existence of stable models of normal program with aggregates is Σ_2^P complete for both the Ferraris' semantics and the FLP semantics [11, 16], which lies on a higher complexity level than the same task for normal programs without aggregates. Together with some well known results in finite model theory, this probably suggests that, in general, firstorder normal programs with arbitrary aggregates cannot be captured in first-order logic with the same type of aggregates at all. In this paper, we shall show that this is indeed the case.

Nevertheless, it is also observed that normal programs with some type of ag-

gregate atoms still have the same complexity as normal programs without aggregates [16]. This means that, simply from a complexity point of view, it is possible to capture this class of programs in first-order logic with the same type of aggregate atoms. In fact, two important classes of such aggregate atoms are well discussed in the literature [4, 13, 16, 27, 31], namely monotone and anti-monotone aggregate atoms.

Definition 5. Let \mathcal{AC} be an aggregate context, $OP \in \mathcal{AC}_{ag}$ be an aggregate symbol, and $\leq \in \mathcal{AC}_{co}$ a comparison operator. We say that OP is monotone with respect to \leq if for any two multisets M_1, M_2 such that $M_1 \subseteq M_2$,

- if M_1 is in the domain of OP^{AC} , then M_2 is also in the domain of OP^{AC} , and
- for any $n \in \mathcal{AC}_{num}$, if $OP^{\mathcal{AC}}(M_1) \preceq n$ then $OP^{\mathcal{AC}}(M_2) \preceq n$.

The definition of anti-monotone is similar, with $M_1 \subseteq M_2$ replaced by $M_2 \subseteq M_1$.

Consider aggregate symbols CARD and SUM in the aggregate context \mathcal{AC}_1 in Example 2. They are either monotone or anti-monotone with respect to the comparison symbols in $\{<, \leq, \geq, >\}$, but neither monotone nor anti-monotone with respect to =.

As we shall show in the paper, normal logic programs with these two types of aggregates can indeed be captured in first-order logic with the same type of aggregates. However, they are not powerful enough to capture all. For instance, a commonly used aggregate is of the form SUM(M) = n, which is neither monotone nor anti-monotone, but can be expressed as the conjunction of the two monotone and anti-monotone aggregates: SUM(M) > n - 1 and SUM(M) < n + 1. Therefore, normal logic programs with the aggregate type SUM(M) = n can be expressed in first-order logic with this aggregate as well.

Hence, an important task is to draw a boundary. That is, does there exist a class of aggregates such that first-order normal programs with this type of aggregates can always be captured in first-order logic with the same type of aggregates, while this cannot be done for normal programs with any other aggregates not in this class?

For this purpose, we extend Liu and Truszczyński's notion of convex constraints [23] to first-order convex aggregates and show that this is exactly the boundary we need. That is, first-order normal logic programs with convex aggregates can always be captured in first-order logic with the same type of aggregates. On the contrary, for any non-convex aggregates, we can always construct a normal program with this aggregate such that it can never be captured in first-order logic with the same type of aggregates providing some general assumptions in the computational complexity theory.

Definition 6. Let \mathcal{AC} be an aggregate context, $OP \in \mathcal{AC}_{ag}$ an aggregate symbol, and $\preceq \in \mathcal{AC}_{co}$ a comparison operator. Then, OP is convex with respect to \preceq if there does not exist finite multisets of tuples of the same arities⁴ $M_1 \subseteq M_2 \subseteq M_3$ and $n \in \mathbb{Z}$ such that $OP^{\mathcal{AC}}(M_1) \preceq n$ and $OP^{\mathcal{AC}}(M_3) \preceq n$ holds, while $OP^{\mathcal{AC}}(M_2) \preceq n$ does not hold.

An aggregate context \mathcal{AC} is convex, if for every aggregate symbol $OP \in \mathcal{AC}_{ag}$ and comparison operator $\leq \leq \mathcal{AC}_{co}$, OP is convex with respect to \leq .

Example 7. Consider the aggregate contexts in Example 2.

- \mathcal{AC}_3 is non-convex since SUM is non-convex with respect to \geq . As an example, both $SUM^{\mathcal{AC}_3}(\{\{-1,1\}\}) \geq 0$ and $SUM^{\mathcal{AC}_3}(\{\{-1\}\}\}) \geq 0$ hold, but $SUM^{\mathcal{AC}_3}(\{\{-1\}\}\}) \geq 0$ does not hold.
- If we restrict to non-negative numbers, SUM is convex with respect to ≥, but AC₂ is still non-convex, since SUM is non-convex with respect to ≠. As an example, we can see that SUM^{AC₂}({{1,2}}) ≠ 1 and SUM^{AC₂}(Ø) ≠ 1 hold, while SUM^{AC₂}({{1}}) ≠ 1 does not hold.
- It can be verified that \mathcal{AC}_1 is convex. Note that aggregates CARD and SUM are convex with respect to =, though they are neither monotone nor anti-monotone.

In fact, both monotone and antimonotone aggregates are subclasses of convex aggregates.

Proposition 1. Let \mathcal{AC} be an aggregate context, $OP \in \mathcal{AC}_{ag}$ be an aggregate symbol and $\leq \in \mathcal{AC}_{co}$ a comparison operator. If OP is (anti)monotone with respect to \leq , then OP is convex with respect to \leq .

⁴That is, for any $\mathbf{t}_1, \mathbf{t}_2 \in (M_1 \cup M_2 \cup M_3)$, we have $|\mathbf{t}_1| = |\mathbf{t}_2|$.

Proof: It suffices to show that if OP is non-convex with respect to \leq , then it is neither monotone nor anti-monotone with respect to \leq .

If OP is non-convex with respect to \preceq , then there exist multisets $M_1 \subseteq M_2 \subseteq M_3$ and $n \in \mathbb{Z}$ such that $OP^{\mathcal{AC}}(M_1) \preceq n$ and $OP^{\mathcal{AC}}(M_3) \preceq n$ hold, while $OP^{\mathcal{AC}}(M_2) \preceq n$ does not hold. Since $M_1 \subseteq M_2$, $OP^{\mathcal{AC}}(M_1) \preceq n$ holds but $OP^{\mathcal{AC}}(M_2) \preceq n$ does not hold, OP is not anti-monotone. Similarly, since $M_2 \subseteq M_3$, $OP^{\mathcal{AC}}(M_3) \preceq n$ holds but $OP^{\mathcal{AC}}(M_1) \preceq n$ does not hold, then OP is not monotone. \Box

3.1. Ordered Completion

Now we define ordered completion for normal logic programs with aggregates. Let σ be a signature. By σ^{\leq} , we denote the signature σ together with the set of new predicates

 $\{\leq_{PQ} \mid P, Q \in \sigma \text{ are two intensional predicates}\},\$

where the arity of \leq_{PQ} is the sum of the arities of P and Q.⁵ The ordered completion of a program Π is defined as a formula over the signature $\tau(\Pi)^{\leq}$.

Let Π be a program. Then by $Trans(\Pi)$, we denote the formula

$$\bigwedge_{P,Q,R\in\mathcal{P}_{int}(\Pi)}\forall\mathbf{xyz}(\leq_{PQ}(\mathbf{xy})\wedge\leq_{QR}(\mathbf{yz})\rightarrow\leq_{PR}(\mathbf{xz})).$$

Also, given two predicates P and Q, we use $P(\mathbf{x}) < Q(\mathbf{y})$ to denote the formula

$$\leq_{PQ} (\mathbf{x}\mathbf{y}) \land \neg \leq_{QP} (\mathbf{y}\mathbf{x}).$$

Definition 7 (ordered completion with aggregates). Let Π be a program. Then the modified completion of Π , denoted by $MComp(\Pi)$, is the formula

$$\widehat{\Pi} \wedge \bigwedge_{\substack{P \in \mathcal{P}_{int}(\Pi)}} \forall \mathbf{x} \Big[P(\mathbf{x}) \to \bigvee_{\substack{r \in \Pi \setminus \Pi^{\perp} \\ Head(r) = P(\mathbf{x})}} \exists \mathbf{y} \big(Body(r) \wedge Pos(\widehat{r}) < P(\mathbf{x}) \wedge PosAgg(\widehat{r}) < P(\mathbf{x}) \big) \Big],$$
(9)

where

⁵Note that P and Q might be the same predicate.

- $\widehat{\Pi}$ is $\bigwedge_{r\in\Pi} \widehat{r}$,
- $Pos(r) < P(\mathbf{x})$ is the formula

$$\bigwedge_{\substack{Q(\mathbf{y})\in Pos(r)\setminus PosAgg(r)\\Q\in\mathcal{P}_{int}(\Pi)}} (\widehat{Q(\mathbf{y}) < P}(\mathbf{x})),$$

• $PosAgg(r) < P(\mathbf{x})$ is the formula

$$\bigwedge_{\delta \in PosAgg(r)} (\operatorname{OP}\langle \mathbf{v} : \exists \mathbf{w}(Bd(\delta) \land Ps(\delta) < P(\mathbf{x})) \rangle \preceq t),$$

in which $Ps(\delta) < P(\mathbf{x})$ is a shorthand of

$$\bigwedge_{\substack{1 \le i \le s \\ Q_i \in \mathcal{P}_{int}(\Pi)}} (Q_i(\mathbf{y}_i) < P(\mathbf{x})).$$

Finally, the ordered completion of Π , denoted by $OC(\Pi)$, is the formula

 $MComp(\Pi) \wedge Trans(\Pi).$

Let us take a closer look at Definition 7. First of all, for non-aggregate atoms in Body(r), we treat them the same way as in the original definition of ordered completion [2]. That is, for each positive non-aggregate atom in Pos(r), we introduce the comparison atoms via the formula $Pos(r) < P(\mathbf{x})$. However, this is not done for negative non-aggregate atoms.

For aggregate atoms occurring in Body(r), similar to non-aggregate atoms, we also distinguish between the negative and positive occurrences. For negative occurrences, we also do not introduce the comparison atoms into these aggregates. However, for positive occurrences, we need to introduce the comparison assertions via $PosAgg(r) < P(\mathbf{x})$, where $PosAgg(r) < P(\mathbf{x})$ denotes the formula:

$$\bigwedge_{\delta \in PosAgg(r)} (\operatorname{OP}\langle \mathbf{v} : \exists \mathbf{w}(Bd(\delta) \land Ps(\delta) < \widetilde{P}(\mathbf{x})) \rangle \preceq t),$$

which simply introduces the comparison atoms into the positive body $Ps(\delta)$ of each aggregate atom in PosAgg(r), i.e., via the formulas of the form $Ps(\delta) < P(\mathbf{x})$. The reason that we introduce the comparison atoms for these positive aggregates is that we need to keep track of the derivation order as implied by the stable model semantics.

Example 8. (Example 5 continued) The ordered completion of Π_1 , denoted by $OC(\Pi_1)$, is the conjunction of $Trans(\Pi_1)$ and the following sentences:

$$\forall x (R_1(x) \to P(x)), \tag{10}$$

$$\forall x \forall y (P(y) \land R_2(y, x) \to P(x)), \tag{11}$$

$$\forall x(\operatorname{SUM}\langle y: P(y) \land R_3(x, y) \rangle > 3 \to P(x)), \tag{12}$$

$$\forall x (P(x) \to (R_1(x) \lor \exists y (P(y) \land R_2(y, x) \land P(y) < P(x)) \lor (SUM\langle y : P(y) \land R_3(x, y) \rangle > 3 \land$$
$$SUM\langle y : P(y) \land R_3(x, y) \land P(y) < P(x) \rangle > 3))).$$
(13)

Now consider again the aggregate context \mathcal{AC}_1 in Example 2 and the structures $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ in Example 6.

- \mathcal{M}_1 is not a model of (12). So it cannot be expanded to a model of $OC(\Pi_1)$.
- Both \mathcal{M}_2 and \mathcal{M}_3 are models of (10)-(12).
- Let M₂' be a structure expanded from M₂ by the additional interpretation for ≤_{PP}:

$$\leq_{PP}^{\mathcal{M}_2'} = \{bc, bd, cd\}$$

It can be verified that

$$\mathcal{M}'_{2} \models Trans(\Pi_{1}),$$

$$\mathcal{M}'_{2} \models R_{1}(x)[x/b],$$

$$\mathcal{M}'_{2} \models P(y) \land R_{2}(y, x) \land P(y) < P(x)[xy/bc],$$

$$\mathcal{M}'_{2} \models \text{SUM}\langle y : P(y) \land R_{3}(x, y) \rangle > 3 \land$$

$$\text{SUM}\langle y : P(y) \land R_{3}(x, y) \land P(y) < P(x) \rangle > 3[x/d]$$

So, \mathcal{M}'_2 is a model of $OC(\Pi_1)$.

 Let M'₃ be a model of Trans(Π₁) expanded from M₃. Now we show that M'₃ is not a model of OC(Π₁). Indeed, since we have

$$\mathcal{M}_3' \not\models P(y) < \tilde{P}(x)[xy/aa],$$

we also have

$$\mathcal{M}_{3}' \not\models P(y) \land R_{2}(y, x) \land P(y) < P(x)[xy/aa],$$
$$\mathcal{M}_{3}' \not\models \operatorname{SUM}\langle y : P(y) \land R_{3}(x, y) \land P(y) < P(x) \rangle > 3[x/a].$$

Therefore, \mathcal{M}'_3 is not a model of (13).

In general, we have the following theorem.

Theorem 1 (main theorem). Let \mathcal{AC} be a convex aggregate context, Π a program and \mathcal{A} a finite structure of $\tau(\Pi)$. Then, \mathcal{A} is a stable model of Π on \mathcal{AC} if and only if \mathcal{A} can be expanded to a model of $OC(\Pi)$.

3.2. The Proof of Theorem 1

To prove Theorem 1, we need to use the notion of externally supported set [6]. Roughly speaking, a set of ground atoms is externally supported if there exists a ground atom in the set and an associated rule that supports the atom (i.e., the atom is the head of the ground rule) and whose positive body could be satisfied by external ground atoms (i.e., ground atoms not in this set). Then, we provide a lemma showing that a structure is a stable model of a program if and only if it is a model of the program and every subset of ground atoms included in this structure is externally supported. Then based on this lemma, we finally give the proof of the main theorem.

Let Π be a program and \mathcal{A} a structure of signature σ such that $\tau(\Pi) \subseteq \sigma$. Then a ground atom is an expression of the form $P(\mathbf{a})$, where P is a predicate and \mathbf{a} is a tuple of domain elements matching the arity of P. By $[\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$, we denote the set of ground atoms $\{P(\mathbf{a}) \mid \mathbf{a} \in P^{\mathcal{A}}, P \in \mathcal{P}_{int}(\Pi)\}$.

Definition 8 (Externally supported set). Let \mathcal{AC} be an aggregate context, Π a program, and \mathcal{A} a structure of σ such that $\tau(\Pi) \subseteq \sigma$. A set of ground atoms $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ is externally supported (under \mathcal{AC} , Π , and \mathcal{A}) if there exists a ground atom $P(\mathbf{a}) \in S$ and a rule r of the form $P(\mathbf{x}) \leftarrow Body(r)$ with local variables \mathbf{y}_r , such that for some assignment of the form $\mathbf{xy}_r/\mathbf{ab}_r$,

- $\mathcal{A} \models Body(r)[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r];$
- $(Pos(r) \setminus PosAgg(r))[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r] \cap S = \emptyset;$
- For all aggregate atoms $\delta \in PosAgg(r)$ of the form (1),

$$OP^{\mathcal{AC}}(\{\!\!\{f_{\mathcal{A}}(\mathbf{c}_v) \mid \mathcal{A} \models Bd(\delta)[\alpha], Ps(\delta)[\alpha] \cap S = \emptyset \}\!\!\}) \preceq f_{\mathcal{A}}(t[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r]),$$

where α is the assignment of the form $\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v$, and \mathbf{c}_w and \mathbf{c}_v are tuples of domain elements.

Lemma 1. ⁶ Let \mathcal{AC} be an aggregate context, Π a program, and \mathcal{A} a structure of $\tau(\Pi)$. Then, \mathcal{A} is a stable model of Π if and only if $\mathcal{A} \models \widehat{\Pi}$ and every $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ is externally supported.

Proof: Suppose that $\mathcal{P}_{int}(\Pi) = \{P_1, \ldots, P_n\}$. Let $\mathbf{U} = \{U_1, \ldots, U_n\}$ be a set of new predicates such that for each $1 \leq i \leq n, U_i$ has the same arity of P_i .

(⇒) Since $\mathcal{A} \models SM(\Pi)$, we have $\mathcal{A} \models \Pi$. It suffices to show that every set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ is externally supported. We prove this by contradiction. Assume that there exists a set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ that is not externally supported. We construct a structure \mathcal{U} of $\tau(\Pi) \cup \mathbf{U}$ as follows:

- $Dom(\mathcal{U}) = Dom(\mathcal{A})$ and $f_{\mathcal{U}}$ and $f_{\mathcal{A}}$ are identical;
- $c^{\mathcal{U}} = c^{\mathcal{A}}$ for each constant $c \in \tau(\Pi)$;
- $P^{\mathcal{U}} = P^{\mathcal{A}}$ for each predicate $P \in \tau(\Pi)$;
- $U_i^{\mathcal{U}} = P_i^{\mathcal{A}} \setminus \{\mathbf{a} \mid P_i(\mathbf{a}) \in S\} \text{ for } 1 \leq i \leq n.$

Clearly, $\mathcal{U} \models \mathbf{U} < \mathcal{P}_{int}(\Pi)$. Since $\mathcal{A} \models SM(\Pi)$, there exists a rule r of the form (4) and an assignment $\mathbf{xy}_r/\mathbf{ab}_r$ such that $\mathcal{U} \not\models r^*[\mathbf{xy}_r/\mathbf{ab}_r]$. Then, we have $\mathcal{U} \models Body(r)^*[\mathbf{xy}_r/\mathbf{ab}_r]$ and $\mathcal{U} \not\models Head(r)^*[\mathbf{xy}_r/\mathbf{ab}_r]$. It follows $\mathcal{A} \models Body(r)[\mathbf{xy}_r/\mathbf{ab}_r]$ and $\mathcal{A} \models Head(r)[\mathbf{xy}_r/\mathbf{ab}_r]$. So, $Head(r)[\mathbf{xy}_r/\mathbf{ab}_r] \in S$.

As we assume that S is not externally supported, by Definition 8, we have

- (1) either there exists an atom $\beta \in Pos(r) \setminus PosAgg(r)$ such that $\beta[\mathbf{xy}_r/\mathbf{ab}_r] \in S$,
- (2) or there exists an aggregate atom δ of the form (1) such that

$$OP^{\mathcal{AC}}(\{\!\{f_{\mathcal{A}}(\mathbf{c}_{v}) \mid \mathcal{A} \models Bd(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}], \\ Ps(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}] \cap S = \emptyset, \\ c_{w} \in Dom(\mathcal{A})^{|w|}, c_{v} \in Dom(\mathcal{A})^{|v|}\}\}) \preceq f_{\mathcal{A}}(t[\mathbf{x}\mathbf{y}_{r}/\mathbf{a}\mathbf{b}_{r}])$$
(14)

does not hold.

⁶Note that this lemma holds without the assumption of convex aggregate contexts.

In both cases, we will show that $\mathcal{U} \not\models Body(r)^*[\mathbf{xy}_r/\mathbf{ab}_r]$, which is a contradiction.

(1) We already have $\beta[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r] \in S$, so $\mathcal{U} \not\models \beta^*[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r]$. It follows $\mathcal{U} \not\models Body(r)^*[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r]$.

(2) By the construction of \mathcal{U} , for every $\mathbf{c}_w \in Dom(\mathcal{A})^{|w|}$ and $\mathbf{c}_v \in Dom(\mathcal{A})^{|v|}$,

$$\mathcal{U} \models Bd(\delta)^* [\mathbf{x}\mathbf{y}_r \mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r \mathbf{c}_w \mathbf{c}_v]$$

if and only if

$$\mathcal{A} \models Bd(\delta)[\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v] \text{ and } Ps(\delta)[\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v] \cap S = \emptyset.$$

Therefore,

$$\{\!\{f_{\mathcal{U}}(\mathbf{c}_{v}) \mid \mathcal{U} \models Bd(\delta)^{*}[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}], \mathbf{c}_{w} \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_{v} \in Dom(\mathcal{A})^{|v|}\}\!\}$$

= $\{\!\{f_{\mathcal{A}}(\mathbf{c}_{v}) \mid \mathcal{A} \models Bd(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}], Ps(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}] \cap S = \emptyset, \mathbf{c}_{w} \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_{v} \in Dom(\mathcal{A})^{|v|}\}\!\}.$

By (14), we have that

$$OP^{\mathcal{AC}}(\{\!\!\{f_{\mathcal{U}}(\mathbf{c}_v) \mid \mathcal{U} \models Bd(\delta)^* [\mathbf{x}\mathbf{y}_r \mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r \mathbf{c}_w \mathbf{c}_v], \\ \mathbf{c}_w \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_v \in Dom(\mathcal{A})^{|v|}\}\!\}) \leq f_{\mathcal{U}}(t[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r]) \quad (15)$$

does not hold. Therefore,

$$\mathcal{U} \not\models (\operatorname{OP}\langle \mathbf{v} : \exists \mathbf{w} Bd(\delta)^* \rangle \preceq t) [\mathbf{x} \mathbf{y}_r / \mathbf{a} \mathbf{b}_r],$$

and $\mathcal{U} \not\models Body(r)^*[\mathbf{xy}_r/\mathbf{ab}_r].$

(\Leftarrow) Since $\mathcal{A} \models \widehat{\Pi}$, it suffices to show that

$$\mathcal{A} \models \neg \exists \mathbf{U}(\mathbf{U} < \mathcal{P}_{int}(\Pi) \land \bigwedge_{r \in \Pi \setminus \Pi^{\perp}} r^*).$$

Again, we prove this by contradiction. Otherwise, let \mathcal{U} be a structure of $\tau(\Pi) \cup \mathbf{U}$ such that \mathcal{U} is an expansion of \mathcal{A} and

$$\mathcal{U} \models \mathbf{U} < \mathcal{P}_{int}(\Pi) \land \bigwedge_{r \in \Pi \setminus \Pi^{\perp}} r^*.$$
(16)

Let $S = \{P(\mathbf{a}) \mid \mathbf{a} \in P_i^{\mathcal{A}} \setminus U_i^{\mathcal{U}}, P_i \in \mathcal{P}_{int}(\Pi)\}$. By (16), S is not empty. Since S is externally supported, there exist a rule r of the form (4) and an assignment $\mathbf{xy}_r/\mathbf{ab}_r$ such that

- $Head(r)[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r] \in S \text{ and } \mathcal{A} \models Body(r)[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r];$
- $(Pos(r) \setminus PosAgg(r))[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r] \cap S = \emptyset;$
- For all aggregate atoms $\delta \in PosAgg(r)$ of the form (1),

$$OP^{\mathcal{AC}}(\{\!\{f_{\mathcal{A}}(\mathbf{c}_{v}) \mid \mathcal{A} \models Bd(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}], Ps(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}] \cap S = \emptyset, \\ \mathbf{c}_{w} \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_{v} \in Dom(\mathcal{A})^{|v|}\}\!\}) \leq f_{\mathcal{A}}(t[\mathbf{x}\mathbf{y}_{r}/\mathbf{a}\mathbf{b}_{r}]).$$

$$(17)$$

By the construction of \mathcal{U} , we have $\mathcal{U} \models Body(r)^*[\mathbf{xy}_r/\mathbf{ab}_r]$ since

- for negative atoms, $\mathcal{U} \models \neg \gamma_j [\mathbf{x} \mathbf{y}_r / \mathbf{a} \mathbf{b}_r], (1 \le j \le m);$
- for positive non-aggregate atoms $\beta \in Pos(r) \setminus PosAgg(r), \mathcal{U} \models \beta^*[\mathbf{xy}_r/\mathbf{ab}_r];$
- for positive aggregate atoms $\delta \in PosAgg(r)$,

$$OP^{\mathcal{AC}}(\{\!\!\{f_{\mathcal{U}}(\mathbf{c}_v) \mid \mathcal{U} \models Bd(\delta)[\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v], \\ \mathbf{c}_w \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_v \in Dom(\mathcal{A})^{|v|}\}\!\}) \leq f_{\mathcal{U}}(t[\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v])$$

and

$$OP^{\mathcal{AC}}(\{\!\!\{f_{\mathcal{U}}(\mathbf{c}_v) \mid \mathcal{U} \models Bd(\delta)^*[\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v], \\ \mathbf{c}_w \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_v \in Dom(\mathcal{A})^{|v|}\}\!\}) \leq f_{\mathcal{U}}(t[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r]).$$

We also have that $\mathcal{U} \not\models Head(r)^*[\mathbf{xy}_r/\mathbf{ab}_r]$. This is a contradiction since $\mathcal{U} \models r^*[\mathbf{xy}_r/\mathbf{ab}_r]$ by (16). This completes our proof. \Box

Now, we are ready to prove our main theorem.

Proof: (\Rightarrow) By Lemma 1, for every set $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$, S is externally supported. We first define a level mapping, denoted by lm, from the set of ground atoms $[\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ to the numbers. Consider the following procedure:

- (I) Let i = 0, and $T = [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$ be the set of atoms whose levels are still undefined.
- (II) If $T = \emptyset$, then quit. Otherwise, T is externally supported, so there exist a ground atom $P(\mathbf{a}) \in T$ and a rule r of the form $P(\mathbf{x}) \leftarrow Body(r)$ with local variables \mathbf{y}_r , such that for some assignment of the form $\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r$,

- $\mathcal{A} \models Body(r)[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r];$
- $(Pos(r) \setminus PosAgg(r))[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r] \cap S = \emptyset;$
- For all aggregate atoms $\delta \in PosAgg(r)$ of the form (1),

$$OP^{\mathcal{AC}}(\{\!\!\{f_{\mathcal{A}}(\mathbf{c}_v) | \mathcal{A} \models Bd(\delta)[\alpha], Ps(\delta)[\alpha] \cap T = \emptyset\}\!\!\}) \preceq f_{\mathcal{A}}(t[\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r]),$$

where α is the assignment of the form $\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v$, and \mathbf{c}_w and \mathbf{c}_v are tuples of domain elements.

- (III) Let $lm(P(\mathbf{a})) = i, i = i + 1$, and $T = T \setminus \{P(\mathbf{a})\}$.
- (IV) Go back to (II).

S is finite, so the procedure defined above always terminates when $T = \emptyset$. Based on this ranking, we expand \mathcal{A} to \mathcal{A}' of the signature $\tau(\Pi) \cup \sigma_{<}$ such that

$$\leq_{PQ}^{\mathcal{A}'} = \{ \mathbf{ab} \mid lm(P(\mathbf{a})) \leq lm(Q(\mathbf{b})) \},\$$

where $P, Q \in \mathcal{P}_{int}(\Pi)$. Now it remains to show that $\mathcal{A}' \models OC(\Pi)$. By the definition of \mathcal{A}' , we have $\mathcal{A}' \models \widehat{\Pi} \wedge Trans(\Pi)$. It suffices to show that for every $P(\mathbf{a}) \in [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}}$, there exist a rule r of the form $P(\mathbf{x}) \leftarrow Body(r)$ with local variables \mathbf{y}_r and an assignment of the form $\mathbf{xy}_r/\mathbf{ab}_r$ such that $P(\mathbf{a}) = Head(r)[\mathbf{xy}_r/\mathbf{ab}_r]$, and

$$\mathcal{A}' \models P(\mathbf{x}) \to \left(\widehat{Body(r)} \land \widehat{Pos(r) < P(\mathbf{x})} \land \widehat{PosAgg(r) < P(\mathbf{x})} \right) [\mathbf{x}\mathbf{y}_r/\mathbf{a}\mathbf{b}_r].$$

Actually, the rule r and the assignment xy_r/ab_r in (II) in the procedure are exactly what we need.

(\Leftarrow) Since $\mathcal{A}' \models OC(\Pi)$, then $\mathcal{A}' \models \widehat{\Pi}$. Therefore, the reduct of \mathcal{A}' on $\tau(\Pi)$ is a model of $\widehat{\Pi}$ since $\widehat{\Pi}$ mentions no comparison predicates. It remains to show that for all $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}'}$, S is externally supported. Otherwise, there exists $S \subseteq [\mathcal{P}_{int}(\Pi)]^{\mathcal{A}'}$ that is not externally supported. In the following, we will construct an infinite sequence of ground atoms:

$$P_1(\mathbf{a}_1),\ldots,P_i(\mathbf{a}_i),\ldots$$

such that

(i) $P_i(\mathbf{a}_i) \in S, i \ge 1;$

- (ii) $\mathcal{A}' \models \leq_{P_{i+1}P_i} (\mathbf{a}_{i+1}\mathbf{a}_i) \land \neg \leq_{P_iP_{i+1}} (\mathbf{a}_i\mathbf{a}_{i+1}), i \geq 1;$
- (iii) $P_i(\mathbf{a}_i) \neq P_j(\mathbf{a}_j), i \neq j.$

This is a contradiction since S is finite. By $\mathcal{A}' \models Trans(\Pi)$, we only need to consider (i) and (ii), since (iii) is a consequence of (ii). We construct the sequence by induction. Assume that $P_1(\mathbf{a}_1)$ is a ground atom in S. If we already have $P_1(\mathbf{a}_1), \ldots, P_i(\mathbf{a}_i)$, we will find $P_{i+1}(\mathbf{a}_{i+1})$ as follows. Since $\mathcal{A}' \models P_i(\mathbf{a}_i)$ and \mathcal{A}' is a model of $OC(\Pi)$, there exists a rule $P_i(\mathbf{x}) \leftarrow Body(r)$ with local variables \mathbf{y}_r and an assignment $\mathbf{x}\mathbf{y}_r/\mathbf{a}_i\mathbf{b}_r$ such that

$$\mathcal{A}' \models Body(r) \land Pos(r) < \widehat{P}(\mathbf{x}) \land PosAgg(r) < \widehat{P}(\mathbf{x})[\mathbf{x}\mathbf{y}_r/\mathbf{a}_i\mathbf{b}_r].$$
(18)

By assumption, S is not externally supported. Consider the rule r and the assignment xy_r/a_ib_r ,

- either there is an atom $\beta \in Pos(r) \setminus PosAgg(r)$ such that $\beta[\mathbf{x}\mathbf{y}_r/\mathbf{a}_i\mathbf{b}_r] \in S$,
- or there is an aggregate atom δ of the form (1) such that

$$OP^{\mathcal{AC}}(M_1) \preceq f_{\mathcal{A}}(t[\mathbf{x}\mathbf{y}_r/\mathbf{a}_i\mathbf{b}_r]) \text{ does not holds},$$
 (19)

where

$$M_{1} = \{\{f_{\mathcal{A}}(\mathbf{c}_{v}) \mid \mathcal{A} \models Bd(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}_{i}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}], \\ Ps(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}_{i}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}] \cap S = \emptyset, \\ \mathbf{c}_{w} \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_{v} \in Dom(\mathcal{A})^{|v|}\}\}.$$
(20)

For the first case, let $P_{i+1}(\mathbf{a}_{i+1}) = \beta[\mathbf{x}\mathbf{y}_r/\mathbf{a}_i\mathbf{b}_r]$. Then, (i) holds trivially, and (ii) holds by (18). For the second case, let M_2 and M_3 be the multisets such that

$$M_{2} = \{\{f_{\mathcal{A}}(\mathbf{c}_{v}) \mid \mathcal{A} \models Bd(\delta)[\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}_{i}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}] \\ \mathbf{c}_{w} \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_{v} \in Dom(\mathcal{A})^{|v|}\}\},$$
(21)

$$M_{3} = \{ \{ f_{\mathcal{A}}(\mathbf{c}_{v}) \mid \mathcal{A} \models Bd(\delta) \land Ps(\delta) < \widehat{P}(\mathbf{x}) [\mathbf{x}\mathbf{y}_{r}\mathbf{w}\mathbf{v}/\mathbf{a}_{i}\mathbf{b}_{r}\mathbf{c}_{w}\mathbf{c}_{v}], \\ \mathbf{c}_{w} \in Dom(\mathcal{A})^{|w|}, \mathbf{c}_{v} \in Dom(\mathcal{A})^{|v|} \} \}.$$
(22)

By (18), we have

$$OP^{\mathcal{AC}}(M_2) \leq f_{\mathcal{A}}(t[\mathbf{x}\mathbf{y}_r/\mathbf{a}_i\mathbf{b}_r])$$
(23)

$$OP^{\mathcal{AC}}(M_3) \preceq f_{\mathcal{A}}(t[\mathbf{x}\mathbf{y}_r/\mathbf{a}_i\mathbf{b}_r])$$
(24)

By (20),(21) and (22), we have $M_1 \subseteq M_2$ and $M_3 \subseteq M_2$. Since \mathcal{AC} is a convex aggregate context, we have $M_3 \not\subseteq M_1$ by (19), (23) and (24). So, there exists an atom $Q(\mathbf{x}') \in Ps(\delta)$ such that $Q(\mathbf{x}')[\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}_i\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v] \in S$ and $\mathcal{A}' \models \leq_{QP} (\mathbf{x}'\mathbf{x}) \land \neg \leq_{PQ} (\mathbf{x}\mathbf{x}')[\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}_i\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v]$. Let $P_{i+1}(x_{i+1}) = Q(\mathbf{x}')[\mathbf{x}\mathbf{y}_r\mathbf{w}\mathbf{v}/\mathbf{a}_i\mathbf{b}_r\mathbf{c}_w\mathbf{c}_v]$. It can be verified that both (i) and (ii) hold.

This completes the proof. \Box

3.3. A Negative Result on Non-Convex Aggregates

Theorem 1 shows that the stable models of a normal program with convex aggregates can be captured by its ordered completion. However, this result does not hold for non-convex aggregates. Consider the following example.

Example 9. Let Π_2 be the following program with a single rule:

$$r_1: P(x) \leftarrow \text{SUM}\langle y: P(y) \land R_1(x, y, z) \rangle \neq z, R_2(x, z)$$

Here, P is the only intensional predicate and R_1, R_2 are extensional predicates. Now let \mathcal{M} be a structure on $\tau(\Pi_2)$ such that

$$Dom(\mathcal{M}) = \{a, b, c\},$$

$$f_{\mathcal{M}}(a) = 2, f_{\mathcal{M}}(b) = 3, f_{\mathcal{M}}(c) = 4,$$

$$R_1^{\mathcal{M}} = \{aab, abb, aac, abc, baa, bba, caa, cca\},$$

$$R_2^{\mathcal{M}} = \{ab, ac, ba, ca\},$$

$$P^{\mathcal{M}} = \{a, b\}.$$

In addition, let \mathcal{M}' be a $\tau(\Pi_2)^{\leq}$ -structure expanded from \mathcal{M} such that

$$\leq_{PP} = \{ba\}.$$

We now show that, under the non-convex aggregate context \mathcal{AC}_2 in Example 2, \mathcal{M}' is a model of $OC(\Pi_2)$ but \mathcal{M} is not a stable model of Π_2 .

On the one side, $OC(\Pi_2)$ is the conjunction of $Trans(\Pi_2)$ and the following sentences:

$$\forall x \forall z (R_2(x, z) \land \text{SUM} \langle y : P(y) \land R_1(x, y, z) \rangle \neq z \to P(x)),$$

$$\forall x (P(x) \to \exists z (R_2(x, z) \land \text{SUM} \langle y : P(y) \land R_1(x, y, z) \rangle \neq z$$

$$\land \text{SUM} \langle y : P(y) \land R_1(x, y, z) \land P(y) < P(x) \rangle \neq z))$$

$$(25)$$

It is obvious that $\mathcal{M}' \models Trans(\Pi_2)$. To show \mathcal{M}' is a model of (25), it suffices to verify that

$$\mathcal{M}' \models \text{SUM}\langle y : P(y) \land R_1(x, y, z) \rangle \neq z[xz/ab],$$

$$\mathcal{M}' \models \text{SUM}\langle y : P(y) \land R_1(x, y, z) \rangle \neq z[xz/ac],$$

$$\mathcal{M}' \models \text{SUM}\langle y : P(y) \land R_1(x, y, z) \rangle \neq z[xz/ba],$$

$$\mathcal{M}' \models \text{SUM}\langle y : P(y) \land R_1(x, y, z) \rangle \neq z[xz/ca].$$

To show \mathcal{M}' is a model of (26), it suffices to verify that

$$\mathcal{M}' \models \operatorname{SUM}\langle y : P(y) \land R_1(x, y, z) \rangle \neq z$$

$$\land \operatorname{SUM}\langle y : P(y) \land R_1(x, y, z) \land P(y) < P(x) \rangle \neq z[xz/ba],$$

$$\mathcal{M}' \models \operatorname{SUM}\langle y : P(y) \land R_1(x, y, z) \rangle \neq z$$

$$\land \operatorname{SUM}\langle y : P(y) \land R_1(x, y, z) \land P(y) < P(x) \rangle \neq z[xz/ac].$$

On the other side, to show that \mathcal{M} is not a stable of Π_2 , it suffices to show that \mathcal{M} is not a model of (8). Indeed, let $U' = \{a\} \subset P^{\mathcal{M}}$. Then we have $\mathcal{M} \models U < P[U/U']$ and $\mathcal{M} \models r_1^*[U/U']$. Hence, $\mathcal{M} \not\models \neg \exists U(U < P \land r_1^*)$. \Box

In fact, convex aggregate is a maximal subclass for this task. That is, for any given non-convex aggregate context, we can always construct a normal program with these aggregates such that it can never be captured in first-order logic with the same type of aggregates, providing some general assumptions in the complexity theory. Actually, we can prove a stronger result that normal programs under any given non-convex aggregate context are able to capture the full expressive power of disjunctive programs (without aggregates).

First of all, we introduce some background. A *disjunctive program with ag*gregates (or *disjunctive program*) is a finite set of *disjunctive rules* of the form

$$\alpha_1; \ldots; \alpha_k \leftarrow \beta_1, \ldots, \beta_l, \operatorname{not} \gamma_1, \ldots, \operatorname{not} \gamma_m, \tag{27}$$

where α_i , $(1 \le i \le k)$ are standard atoms, β_i , $(1 \le i \le l)$, and γ_j , $(1 \le j \le m)$ are atoms. The stable models of a disjunctive program are defined as the models of $SM(\Pi)$, where $SM(\Pi)$ is the same as (8) except that for a disjunctive rule r of the form (27), r^* is the universal closure of the formula

$$Body(r)^* \to \bigvee_{1 \le i \le k} \alpha_i^*.$$
 (28)

Our negative result is inspired by Ferraris' work [16]. In order to obtain the complexity results for propositional normal programs with arbitrary aggregate atom, Ferraris proved the following two facts:

(Fact 1) Every propositional disjunctive rule of the form

$$a_1; \ldots; a_k \leftarrow b_1, \ldots, b_l, \operatorname{not} c_1, \ldots, \operatorname{not} c_m$$

can be equivalently transformed into a set of rules with implications in the bodies as follows

(Fact 2) An implication $p \rightarrow q$ is strongly equivalent to the following aggregate atom

$$sum\langle p = -1, q = 1 \rangle \ge 0, \tag{30}$$

where two expressions are said to be strongly equivalent in Answer Set Programming if replacing one by another in any logic program does not change the answer sets.

Therefore, any propositional disjunctive program without aggregate atom can be equivalently transformed into a propositional normal program with aggregates similar to (30). As a consequence, the complexity of checking the existence of answer sets of propositional normal program with arbitrary aggregate atom is Σ_2^P -complete, which is higher than that of normal program with monotone and antimonotone aggregate atom (NP-complete).

We extend Ferraris' result in an essential way in the sense that the above procedure can be applied for every non-convex aggregate context. That is, given any non-convex aggregate context, we can use it to simulate implications, thus to simulate disjunctive rules. The full proof of this result in the first-order case is rather technical and tedious. We provide a sketch of the proof here, and leave the very long detailed full proof in Appendix A.

Lemma 2. Let \mathcal{AC} be a polynomial and non-convex aggregate context. Then, on finite structures, every disjunctive program Γ_D without aggregates can be polynomially translated into a normal program Γ_N with some aggregates from \mathcal{AC} .

Proof (*Sketch*) First, since \mathcal{AC} is a non-convex aggregate context, there exist three multisets M_1, M_2, M_3 such that $OP(M_1) \preceq N$ and $OP(M_3) \preceq N$ hold while $OP(M_1) \preceq N$ does not hold, where $OP \in \mathcal{AC}_{ag}$ is an aggregate symbol, $\preceq \in \mathcal{AC}_{op}$ is a comparison operator, and $N \in \mathcal{AC}_{num}$ is a number. Let

- $M_1 = \{\!\!\{\mathbf{m}_1, \ldots, \mathbf{m}_{N_1}\}\!\!\};$
- $M_2 \setminus M_1 = \{\!\!\{\mathbf{m}_{N_1+1}, \ldots, \mathbf{m}_{N_2}\}\!\!\};$
- $M_3 \setminus M_2 = \{\{\mathbf{m}_{N_2+1}, \dots, \mathbf{m}_{N_3}\}\},\$

where N_1, N_2, N_3 are the sizes of M_1, M_2, M_3 respectively, and $0 \le N_1 < N_2 < N_3$.

Then, we introduce some new predicates and constants to build the program Γ_N . Let $QD_1, \ldots QD_{N_3}$ be N_3 new predicates. The program Γ_N contains some rules and constraints such that if \mathcal{M} is a stable model of Γ_N , then $\mathcal{M} \models QD_i(\mathbf{d}_i)$, if and only if $f_{\mathcal{M}}(\mathbf{d}_i^{\mathcal{M}}) = \mathbf{m}_i$, where \mathbf{d}_i , $(1 \le i \le N_3)$ is a tuple of new constants. In addition, let QM_1, QM_2, QM_3 be three new predicates. With the help of $QD_1, \ldots QD_{N_3}$, the program Γ_N is built such that if \mathcal{M} is a stable model of Γ_N , then

$$\mathcal{M} \models \operatorname{OP} \langle \mathbf{x} : QM_1(\mathbf{x}) \rangle \preceq cn$$
$$\mathcal{M} \not\models \operatorname{OP} \langle \mathbf{x} : QM_2(\mathbf{x}) \rangle \preceq cn$$
$$\mathcal{M} \models \operatorname{OP} \langle \mathbf{x} : QM_3(\mathbf{x}) \rangle \preceq cn,$$

where cn is a new constant for N. Note that such \mathcal{M} does not exist if the aggregate context is convex.

Furthermore, for every rule r of the form (27) and two atoms $P_i(\mathbf{v}_i), P_j(\mathbf{v}_j), (1 \leq i, j \leq k)$ in Head(r), a new predicate $Q_{r,i,j}$ is introduced. With some carefully defined rules and constraints in Γ_N and the new predicates introduced above, the aggregate $OP\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \leq cn$ behaves exactly the same as the implication $P_j(\mathbf{v}_i) \rightarrow P_i(\mathbf{v}_i)$. This is similar to the (Fact 2) in Ferraris' work [16].

Now, we present the main idea of the translation. The normal program with aggregates Γ_N has three parts of the rules:

$$\Gamma_N = \mathsf{AGG} \cup \mathsf{DEF} \cup \mathsf{CST},$$

where AGG is a set of normal rules with aggregates which could simulate the disjunctive rules, and DEF and CST are sets of rules to define the new predicates.

Let $r \in \Gamma_D \setminus {\Gamma_D}^{\perp}$ be a rule of the form (27). Then, r is translated into k rules in Γ_N of the form:

$$\alpha_{i} \leftarrow \beta_{1}, \dots, \beta_{l}, \operatorname{not} \gamma_{1}, \dots, \operatorname{not} \gamma_{m},$$

$$OP\langle \mathbf{x} : Q_{r,i,1}(\mathbf{x}, \mathbf{v}_{i}, \mathbf{v}_{1}) \rangle \preceq cn,$$

$$\dots,$$

$$OP\langle \mathbf{x} : Q_{r,i,k}(\mathbf{x}, \mathbf{v}_{i}, \mathbf{v}_{k}) \rangle \preceq cn,$$

$$NotLitsNew(Var(r)),$$
(31)

where:

- $1 \le i \le k$, and k is the number of atoms in the head of r,
- $P_1(\mathbf{v}_1), \ldots, P_k(\mathbf{v}_k)$ are atoms in the head of r,
- x is a tuple of distinct new variables,
- NotLitsNew(Var(r)) denotes the set of negative atoms of the form not x = c, where x is a variable in r and c is a new constant not in $\tau(\Gamma_D)$.

Intuitively, the rules defined above play the same roles in first-order case as those rules (29) do in the propositional case. This is similar to (*Fact 1*) in Ferraris' work [16].

Finally, we show that a structure \mathcal{M}_D on $\tau(\Gamma_D)$ is a stable model of Γ_D if and only if there is a structure \mathcal{M}_N on $\tau(\Gamma_N)$ such that \mathcal{M}_N is a stable model of Γ_N , where \mathcal{M}_D and \mathcal{M}_N agree on all interpretations of predicates and constants in $\tau(\Gamma_D)$. \Box

Our negative result follows from Lemma 2 since disjunctive programs without aggregates can capture the complexity class Σ_2^P [9] but first-order logic with new predicates can only capture the complexity class NP [15].

Theorem 2. Let \mathcal{AC} be a polynomial and non-convex aggregate context. Then, there exists a normal program with aggregates Π such that it cannot be translated into any first-order sentences (with extra predicates) on finite structures with the aggregate context \mathcal{AC} unless NP = coNP.

Proof: By Lemma 2, the following program $\Pi_D^{3-\text{UNCOLOR}}$ (a disjunctive program without aggregates) for 3-uncolorability can be translated into a normal program

with aggregates $\Pi_N^{3-\text{UNCOLOR}}$:

$$\begin{split} R(x); G(x); B(x) \leftarrow, \\ NC \leftarrow E(x,y), R(x), R(y), \\ NC \leftarrow E(x,y), G(x), G(y), \\ NC \leftarrow E(x,y), B(x), B(y), \\ R(x) \leftarrow NC, \\ G(x) \leftarrow NC, \\ B(x) \leftarrow NC, \\ NC \leftarrow \operatorname{not} NC. \end{split}$$

Assume that $NP \neq coNP$. As 3-uncolorability is a coNP-complete problem, by Fagin's theorem [15], it cannot be captured in existential second-order logic on finite structures. Note that adding polynomial aggregate atoms into first-order logic (existential second-order logic) does not increase the expressive power in finite model theory. So 3-uncolorability cannot be captured in existential secondorder logic with polynomial aggregates either. Hence, it cannot be translated into any first-order sentences (with extra predicates) on finite structures with the polynomial aggregate context AC. \Box

4. Implementation and Experimental Results

Based on Theorem 1, we have implemented a new system, called GROCv2 (GRounder on Ordered Completion Version 2), for computing answer sets of firstorder normal logic programs with convex aggregates. This work is an extension of the prototype implementation GROC [2], which can only handle normal programs without aggregates.

Following the basic ideas of GROC, GROCv2 computes the answer sets of a normal program with convex aggregates together with an extensional database as follows.

- 1. First, GROCv2 translates the program into its enhanced ordered completion (see Definition 7).
- 2. Then, GROCv2 grounds the ordered completion into propositional theories (with modular theories for dealing with comparison predicates) based on the extensional database.

3. Finally, GROCv2 calls an SMT solver to compute a classical model of the propositional theories, which is corresponding to an answer set of the original program with the extensional database according to Theorem 1.

Note that this is significantly different from traditional ASP solvers that ground the logic programs directly into propositional programs. There are several potential benefits. First of all, the classical first-order semantics is much simpler than the stable model semantics. Therefore, more solving techniques, e.g. heuristic methods and simplification techniques, can be applied. Secondly, the results of grounding in GROCv2 are propositional theories, which can normally be solved more easily than propositional programs. Thirdly, Step 1 is a polynomial translation and it can be done offline. Moreover, some traditional techniques might be used here to simplify the first-order formula. Finally, Step 3 calls an SMT solver, which is used as a black box. Hence, it can be improved by new advances in the SAT/SMT community.

The main disadvantage of GROCv2 is that a number of new predicates (namely the comparison predicates) are introduced, which would potentially result in a bigger grounding theory. In principle, this is certainly the case. That is, ordered completion needs to introduce n^2 number of new predicates, where n is the number of intensional predicates in the program. However, in practice, the actual cost is not that much for many benchmark problems. One reason is that we only need to introduce the comparison predicates for those predicates in the same strongly connected component in the predicate dependency graph. This will significantly reduce the number of new predicates introduced. For instance, for the traveling salesman problem, we only need to introduce 1 (instead of 9) comparison predicate.

Also, note that we use SMT solvers in Step 3 instead of SAT solvers in GROCv2. The only reason is that SMT solvers are more efficient for dealing with comparison atoms and aggregates. In principle, one can use a SAT solver instead.

The key part of GROCv2 is Step 2 - the grounder that transforms the ordered completion together with an extensional database into propositional theories (with modular theories and aggregates). For this purpose, we first introduce some background about propositional SMT (Satisfiability Modulo Theories) with aggregates. In general, propositional SMT formulas are classical propositional formulas enhanced with a modular theory to express some components that cannot be easily handled in propositional logic, for instance $3x + y \le 10$. However, for dealing with comparison atoms mentioned in this paper, we only need a simple modular theory to compare the values of two numbers mapped from propositional atoms.

Definition 9. Let \mathcal{D} be a set of propositional atoms, A propositional SMT formula with aggregates (or SMT formula for short) on \mathcal{D} is defined as

$$\phi ::= \top \mid \alpha \mid (\alpha_1, \alpha_2)_{<} \mid \operatorname{OP}\langle c_1 : \phi_1, \cdots, c_k : \phi_k \rangle \preceq c \mid$$

$$\phi_1 \land \phi_2 \mid \neg \phi_1, \tag{32}$$

where $\alpha, \alpha_1, \alpha_2 \in \mathcal{D}$, $OP \in \mathcal{AG}$ is an aggregate symbol, $\leq \mathcal{CO}$ is a comparison symbol, $c_1, \ldots, c_k \in \mathbb{Z}$ are integer constants, and ϕ_1, \ldots, ϕ_k are SMT formulas.

The semantics of SMT formulas is similar to that of propositional formulas. Let $I \subseteq \mathcal{D}$ be a set of propositional atoms, and F a function from \mathcal{D} to \mathbb{Z} . Given an aggregate context \mathcal{AC} , the satisfaction relation between a pair (I, F) and an SMT formula ϕ , denoted by $(I, F) \models_{\mathcal{AC}} \phi$ (or simply $(I, F) \models \phi$ if \mathcal{AC} is clear from the context), is defined as:

- $(I, F) \models_{\mathcal{AC}} \top;$
- $(I, F) \models_{\mathcal{AC}} \alpha$, if α is an atom and $\alpha \in I$;
- $(I, F) \models_{\mathcal{AC}} (\alpha_1, \alpha_2)_<$, if $F(\alpha_1) < F(\alpha_2)$;
- $(I, F) \models_{\mathcal{AC}} OP\langle c_1 : \phi_1, \dots, c_k : \phi_k \rangle \leq c$, if $OP \in \mathcal{AC}_{ag}, \leq \mathcal{AC}_{co}, M$ is in the domain of $OP^{\mathcal{AC}}$ and $OP^{\mathcal{AC}}(M) \leq c$, where

$$M = \{\!\!\{c_i \mid (I, F) \models_{\mathcal{AC}} \phi_i, (1 \le i \le k)\!\!\}\};$$

- $(I, F) \models_{\mathcal{AC}} \phi_1 \land \phi_2$, if $(I, F) \models_{\mathcal{AC}} \phi_1$ and $(I, F) \models_{\mathcal{AC}} \phi_2$;
- $(I, F) \models_{\mathcal{AC}} \neg \phi_1$, if it is not the case that $(I, F) \models_{\mathcal{AC}} \phi_1$.

A set of propositional atoms $I \subseteq \mathcal{D}$ is a *model* of ϕ if there exists a function F from \mathcal{D} to \mathbb{Z} such that $(I, F) \models \phi$.

Now, we are able to ground a first-order formula with aggregates into SMT formulas defined above under a given domain. Let σ and σ' be two signatures such that they contain the same constants and the set of predicates in σ' is a subset of that in σ . Let \mathcal{A} be a structure on σ' . By $\mathcal{D}_{\mathcal{A}}$, we denote the set

$$\mathcal{D}_{\mathcal{A}} = \{ P(\mathbf{a}) \mid P \in \mathcal{P}_{int}, \mathbf{a} \in Dom(\mathcal{A})^n, \text{ where } n \text{ is the arity of } P \}.$$

Let Φ be a first-order formula with aggregates on σ^{\leq} , and \mathbf{y}/\mathbf{a} an assignment such that $free(\Phi) \subseteq \mathbf{y}$. The grounding of Φ on \mathbf{x}/\mathbf{a} with respect to \mathcal{A} , denoted by $\mathcal{GR}_{\mathcal{A}}(\Phi, \mathbf{y}/\mathbf{a})$, is an SMT formula defined recursively as follows:

- \top , if $\Phi = \alpha$ is either an equality atom or a standard atom of the form $P(\mathbf{t})$ such that P is a predicate in σ' , and $\mathcal{A} \models \alpha[\mathbf{y}/\mathbf{a}]$;
- $\neg \top$, if $\Phi = \alpha$ is either an equality atom or a standard atom of the form $P(\mathbf{t})$ such that P is a predicate in σ' , and $\mathcal{A} \not\models \alpha[\mathbf{y}/\mathbf{a}]$;
- P(t)[y/a], if Φ = P(t) is a standard atom such that P is a predicate in σ but not in σ';
- (P(t₁)[y/a], Q(t₂)[y/a])<, if Φ =≤_{PQ} (t₁t₂) such that ≤_{PQ} is a comparison predicate in σ[≤];
- OP $\langle M^* \rangle \leq f_{\mathcal{A}}(t[\mathbf{y}/\mathbf{a}])$, if $\Phi = \delta$ is an aggregate atom of the form (1), where

$$M^* = \{ \{ (f_{\mathcal{A}}(\mathbf{c}) : \mathcal{GR}_{\mathcal{A}}(\exists \mathbf{w} Bd(\delta), \mathbf{yv}/\mathbf{ac})) \mid \mathbf{c} \in M', f_{\mathcal{A}}(\mathbf{c}) \text{ is defined} \} \},\$$

and

$$M' = \{ \mathbf{c} \mid \mathcal{A} \models Bd(\delta)[\mathbf{ywv}/\mathbf{abc}], \mathbf{b} \in Dom(\mathcal{A})^{|\mathbf{w}|}, \mathbf{c} \in Dom(\mathcal{A})^{|\mathbf{v}|} \}$$
(33)

- $\neg \mathcal{GR}_{\mathcal{A}}(\Phi_1, \mathbf{y}/\mathbf{a})$, if Φ is of the form $\neg \Phi_1$;
- GR_A(Φ₁, y/a) ∨ GR_A(Φ₂, y/a), if Φ is of the form Φ₁ ∨ Φ₂;
- $\mathcal{GR}_{\mathcal{A}}(\bigwedge_{a \in Dom(\mathcal{A})} \Phi_1[y/a], \mathbf{y/a})$, if Φ is of the form $\forall y \Phi_1$.

If Φ is a formula without free variables, then the grounding of Φ with respect to \mathcal{A} is simply written as $\mathcal{GR}_{\mathcal{A}}(\Phi)$.

The following theorem shows that we can compute the models of ordered completions by grounding.

Theorem 3. Let Π be a program, and \mathcal{A}_e a finite structure on $\tau_{ext}(\Pi)$. Let \mathcal{A} be a structure on $\tau(\Pi)^{\leq}$, which is an expansion of \mathcal{A}_e and $\mathcal{A} \models Trans(\Pi)$. Then, $\mathcal{A} \models OC(\Pi)$ iff $I_{\mathcal{A}} \models \mathcal{GR}_{\mathcal{A}_e}(OC(\Pi))$, where

$$I_{\mathcal{A}} = \{ P(\mathbf{a}) \mid P \in \mathcal{P}_{int}(\Pi), \mathbf{a} \in P^{\mathcal{A}} \}.$$

Proof: Let $P(\mathbf{a}), Q(\mathbf{b}), \text{ and } R(\mathbf{c})$ be three elements in $I_{\mathcal{A}}$. Since $\mathcal{A} \models Trans(\Pi)$, we have $\mathcal{A} \models \leq_{PQ} (\mathbf{xy}) \land \leq_{QR} (\mathbf{yz}) \rightarrow \leq_{PR} (\mathbf{xz})[\mathbf{xyz/abc}]$.

Therefore, there exists a function F_A from I_A to \mathbb{Z} such that:

- $F_{\mathcal{A}}(P(\mathbf{a})) \neq F_{\mathcal{A}}(Q(\mathbf{b}))$ if $P(\mathbf{a})$ and $Q(\mathbf{b})$ are different;
- $F_{\mathcal{A}}(P(\mathbf{a})) < F_{\mathcal{A}}(Q(\mathbf{b}))$ if and only if $\mathcal{A} \models \leq_{PQ} (\mathbf{xy})[\mathbf{xy}/\mathbf{ab}].$

It suffices to prove the following result.

Let Φ be a formula on $\tau(\Pi)^{\leq}$, and \mathbf{y}/\mathbf{a} an assignment such that $free(\Phi) \subseteq \mathbf{y}$. Then, $\mathcal{A} \models \Phi[\mathbf{y}/\mathbf{a}]$ if and only if $(I_{\mathcal{A}}, F_{\mathcal{A}}) \models \mathcal{GR}_{\mathcal{A}_e}(\Phi, \mathbf{y}/\mathbf{a})$.

We prove this by induction on the structure of Φ .

- It is straightforward that A ⊨ ⊤ if and only if (I_A, F_A) ⊨ GR_{A_e}(⊤), and A ⊨ P(t)[y/a] if and only if (I_A, F_A) ⊨ GR_{A_e}(P(t), y/a), where P ∈ τ_{ext}(Π).
- For intensional predicate P ∈ τ_{int}(Π), we also have A ⊨ P(t)[y/a] if and only if I_A ⊨ GR_{A_e}(P(t), y/a) by noticing that P(a) ∈ I_A if and only if a ∈ P^A for every tuple a that matches the arity of P.
- For a comparison atom ≤_{PQ} (t₁t₂), A ⊨≤_{PQ} (t₁t₂)[y/a], where free(t₁t₂) ⊆ y, if and only if

$$F_{\mathcal{A}}(P(\mathbf{t_1})[\mathbf{y}/\mathbf{a}]) < F_{\mathcal{A}}(Q(\mathbf{t_2})[\mathbf{y}/\mathbf{a}])$$

if and only if

$$(I_{\mathcal{A}}, F_{\mathcal{A}}) \models (P(\mathbf{t_1})[\mathbf{y/a}], Q(\mathbf{t_2})[\mathbf{y/a}])_{<}$$

if and only if

$$(I_{\mathcal{A}}, F_{\mathcal{A}}) \models \mathcal{GR}_{\mathcal{A}_e}(\Phi).$$

• If $\Phi = \delta$ is an aggregate atom of the form (1) such that $free(\Phi) \subseteq \mathbf{y}$, we will show that $\mathcal{A} \models \delta[\mathbf{y}/\mathbf{a}]$ if and only if $(I_{\mathcal{A}}, F_{\mathcal{A}}) \models \mathcal{GR}_{\mathcal{A}_e}(\delta, \mathbf{y}/\mathbf{a})$. Consider the following set and multiset:

$$M' = \{ \mathbf{c} \mid \mathcal{A} \models Bd(\delta)[\mathbf{ywv}/\mathbf{abc}], \mathbf{b} \in Dom(\mathcal{A})^{|\mathbf{w}|}, \mathbf{c} \in Dom(\mathcal{A})^{|\mathbf{v}|} \},$$
(34)

$$M^* = \{ \{ (f_{\mathcal{A}}(\mathbf{c}) : \mathcal{GR}_{\mathcal{A}}(\exists \mathbf{w} Bd(\delta), \mathbf{yv}/\mathbf{ac})) \mid \mathbf{c} \in M', f_{\mathcal{A}}(\mathbf{c}) \text{ is defined} \} \},$$
(35)

$$M_1 = \{ \{ f_{\mathcal{A}}(\mathbf{c}) \mid \mathbf{c} \in M', f_{\mathcal{A}}(\mathbf{c}) \text{ is defined} \} \}$$
(36)

$$M_2 = \{\!\!\{c \mid (c,\phi) \in M^*, (I,F) \models_{\mathcal{AC}} \phi\}\!\!\}$$

$$(37)$$

We have $M_1 = M_2$ by noticing that

$$\mathcal{A} \models Bd(\delta)[\mathbf{ywv/abc}]$$

if and only if

$$\mathcal{A} \models \exists \mathbf{w} Bd(\delta)[\mathbf{y}\mathbf{v}/\mathbf{a}\mathbf{c}],$$

if and only if

$$(I_{\mathcal{A}}, F_{\mathcal{A}}) \models \mathcal{GR}_{\mathcal{A}}(\exists \mathbf{w}Bd(\delta), \mathbf{yv/ac}).$$

Hence, $\mathcal{A} \models \delta[\mathbf{y}/\mathbf{a}]$ if and only if $(I_{\mathcal{A}}, F_{\mathcal{A}}) \models \mathcal{GR}_{\mathcal{A}_e}(\delta, \mathbf{y}/\mathbf{a})$.

If Φ is of the form ¬Φ₁, Φ₁ ∧ Φ₂ and ∀xΦ₁, it is also straightforward to see that A ⊨ Φ[y/a] if and only if (I_A, F_A) ⊨ GR_{A_e}(Φ, y/a).

Based on Theorem 3, we have implemented a new solver GROCv2. Now we report some experimental results. We use Z3 (version 4.3.2)⁷ in GROCv2 as the underlying SMT solver. We compare our approach GROCv2 + Z3 with three state-of-the-art ASP solvers, namely DLV (version 4.2.1),⁸ CLASP (version 3.0.0),⁹ and CMODELS (version 3.85),¹⁰. We use GRINGO (version 3.0.5)¹¹ as the grounder for the solvers CLASP and CMODELS. The reason why we choose CLASP, DLV, and CMODELS here is because the first two are representative ASP solvers based on conflict analysis, while the last is a representative solver based on loop formulas.

We are mainly interested in non-tight programs as for tight programs, ordered completion is just Clark's completion. In this paper, we consider three benchmark programs, the *bounded-traveling salesman* problem, the *Nurikabe* puzzle, and the *weight-bounded dominating set* problem.

Tables 1 and 2 report our results for the bounded traveling salesman program. Here, Table 1 contains the results for our randomly generated instances while Table 2 contains the results taken from the ASPARAGUS benchmark suite.¹² For

⁷ http://z3.codeplex.com/

⁸ http://www.dlvsystem.com/

⁹ http://potassco.sourceforge.net/

¹⁰ http://www.cs.utexas.edu/users/tag/cmodels.html

¹¹ http://potassco.sourceforge.net/

¹² http://asparagus.cs.uni-potsdam.de/instanceclass/show/id/34

randomly generated instances, "rand_ x_y_i " represents a random graph with x number of nodes and y edges and of instance i. We set the timeouts to be 1000.00 sec, denoted by "——" in the tables. Note that we also include the grounding time for CLASP as well as CMODELS. For clarity, we pick up the best solver for each instance, highlighted by bold fonts. From Tables 1 and 2, it can be observed that GROCv2 + Z3 and GRINGO + CLASP outperform DLV and GRINGO + CMODELS in most cases on the bounded traveling salesman program while the previous two solvers are comparable. Interestingly, for randomly generated instances, GROCv2 + Z3 seems to have an advantage over GRINGO + CLASP on big problem instances.

Table 3 reports the experimental results on the Nurikabe puzzle. Here, we omit the results of DLV as it seems not to return the correct answers. One can obtain a similar conclusion on this program. That is, GROCv2 + Z3 and GRINGO + CLASP are slightly better than GRINGO + CMODELS and the previous two are comparable with each other. Again, for randomly generated instances, the bigger the problem is, the better GROCv2 + Z3 performs in comparison with GRINGO + CLASP. Table 4 reports our experiments on the weight-bounded dominating set program with the instances from the ASPARAGUS benchmark suites.¹³ On this program, GRINGO + CLASP is the clear winner. In addition, GROCv2 + Z3 performs slightly worse than DLV as well, although it is comparable to GRINGO + CMODELS.

To sum up, our solver GROCv2 + Z3 is comparable to other state-of-the-art ASP solvers in the literature. Since this is a first implementation so that many simplified techniques, e.g., first-order simplification of ordered completion, are not employed, we believe that ordered completion provides a new promising way to implement answer set programming. We have done some experimental analysis on other benchmark programs as well, including *N*-queens, bounded-spanning tree and projected hamiltonian cycle. For more details about our solver and the benchmark program instances, please see the following link http://staff.scm.uws.edu.au/~y: Unfortunately, the current version of GROCv2 does not support functions so that it is not able to handle some benchmark programs. We leave it to our future investigations.

¹³http://asparagus.cs.uni-potsdam.de/instanceclass/show/id/33

Instances	GROCv2 + Z3	CLASP	DLV	CMODELS
rand_50_300_1	109.23 (SAT)	9.95 (SAT)	—— (?)	—— (?)
rand_50_300_2	280.17 (SAT)	—— (?)	—— (?)	—– (?)
rand_50_300_3	0.03 (UNSAT)	0.01 (UNSAT)	0.04 (UNSAT)	0.01 (UNSAT)
rand_50_300_4	12.81 (SAT)	125.42 (SAT)	—— (?)	—— (?)
rand_50_300_5	38.71 (SAT)	132.88 (SAT)	—— (?)	—— (?)
rand_50_300_6	96.23 (SAT)	—— (?)	—— (?)	—— (?)
rand_50_300_7	0.03 (UNSAT)	0.01 (UNSAT)	0.05 (UNSAT)	0.01 (UNSAT)
rand_50_300_8	0.48 (SAT)	4.13 (SAT)	632.11 (SAT)	35.89 (SAT)
rand_50_300_9	89.49 (SAT)	315.93 (SAT)	—— (?)	—— (?)
rand_50_300_10	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.04 (UNSAT)
rand_60_350_1	0.79 (SAT)	18.85 (SAT)	—— (?)	—— (?)
rand_60_350_2	32.35 (SAT)	—— (?)	—— (?)	—— (?)
rand_60_350_3	109.61 (SAT)	140.03 (SAT)	—— (?)	—— (?)
rand_60_350_4	34.39 (SAT)	236.68 (SAT)	—— (?)	—— (?)
rand_60_350_5	0.88 (SAT)	76.91 (SAT)	—— (?)	—— (?)
rand_60_350_6	483.07 (SAT)	79.70 (SAT)	—— (?)	—— (?)
rand_60_350_7	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.05 (UNSAT)
rand_60_350_8	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.05 (UNSAT)
rand_60_350_9	8.01 (SAT)	101.7 (SAT)	119.68 (SAT)	—— (?)
rand_60_350_10	—— (?)	105.70 (SAT)	—— (?)	—— (?)
rand_70_400_1	123.84 (SAT)	—— (?)	—– (?)	—— (?)
rand_70_400_2	658.51 (SAT)	340.13 (SAT)	—— (?)	—— (?)
rand_70_400_3	113.39 (SAT)	9.57 (SAT)	—— (?)	—— (?)
rand_70_400_4	281.80 (SAT)	—— (?)	—— (?)	—— (?)
rand_70_400_5	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.06 (UNSAT)
rand_70_400_6	43.30 (SAT)	—— (?)	—— (?)	—— (?)
rand_70_400_7	17.02 (SAT)	361.151 (SAT)	—— (?)	—— (?)
rand_70_400_8	(?)	(?)	(?)	(?)
rand_70_400_9	424.07 (SAT)	(?)	(?)	(?)
rand_70_400_10	87.85 (SAT)	(?)	(?)	(?)

Table 1: Bounded-Traveling Salesman (randomly generated).

Instances	GROCv2 + Z3	CLASP	DLV	CMODELS
dom_rand_70_300_x_3	533.23 (UNSAT)	690.24 (UNSAT)	—— (?)	—— (?)
rand_70_300_x_0	—— (?)	—— (?)	—— (?)	—— (?)
rand_70_300_x_3	438.49 (UNSAT)	635.40 (UNSAT)	—— (?)	—— (?)
rand_70_300_x_4	0.03 (UNSAT)	0.02 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)
rand_70_300_x_5	0.06 (SAT)	0.01 (SAT)	0.10 (SAT)	8.732 (SAT)
rand_70_300_x_7	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.04 (UNSAT)
rand_70_300_x_8	0.03 (UNSAT)	0.01 (UNSAT)	0.02 (UNSAT)	0.05 (UNSAT)
rand_70_300_x_9	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.05 (UNSAT)
rand_70_300_x_11	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.04 (UNSAT)
rand_70_300_x_12	0.02 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.04 (UNSAT)
rand_70_300_x_14	0.04 (SAT)	0.02 (SAT)	—— (?)	92.42 (SAT)
rand_80_340_y_0	0.03 (UNSAT)	0.02 (UNSAT)	0.02 (UNSAT)	0.05 (UNSAT)
rand_80_340_y_4	0.03 (UNSAT)	0.01 (UNSAT)	0.02 (UNSAT)	0.06 (UNSAT)
rand_80_340_y_10	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.06 (UNSAT)
rand_80_340_y_11	0.03 (UNSAT)	0.01 (UNSAT)	0.02 (UNSAT)	0.06 (UNSAT)
rand_80_340_y_13	(?)	—— (?)	—— (?)	—— (?)
rand_80_340_y_15	0.03 (UNSAT)	0.01 (UNSAT)	0.02 (UNSAT)	0.06 (UNSAT)
rand_80_340_y_16	0.03 (UNSAT)	0.02 (UNSAT)	0.02 (UNSAT)	0.05 (UNSAT)
rand_80_340_y_17	0.03 (UNSAT)	0.02 (UNSAT)	0.01 (UNSAT)	0.04 (UNSAT)
rand_80_340_y_18	0.03 (UNSAT)	0.01 (UNSAT)	0.01 (UNSAT)	0.04 (UNSAT)

Table 2: Bounded-Traveling Salesman (from ASPARAGUS website).

Instances	GROCv2 + Z3	CLASP	DLV	CMODELS
puzzle_10_10_1	20.84 (UNSAT)	4.09 (UNSAT)	N/A	60.62 (UNSAT)
puzzle_10_10_2	35.91 (SAT)	1.62 (SAT)	N/A	12.14 (SAT)
puzzle_10_10_3	20.84 (SAT)	0.85 (SAT)	N/A	4.62 (SAT)
puzzle_10_10_4	25.08 (SAT)	1.97 (SAT)	N/A	11.46 (SAT)
puzzle_10_10_5	15.72 (SAT)	0.83 (SAT)	N/A	8.40 (SAT)
puzzle_10_10_6	11.62 (SAT)	1.48 (SAT)	N/A	35.40 (SAT)
puzzle_10_10_7	15.45 (SAT)	0.73 (SAT)	N/A	4.33 (SAT)
puzzle_12_12_1	20.70 (SAT)	1.61 (SAT)	N/A	17.33 (SAT)
puzzle_12_12_2	25.29 (SAT)	1.49 (SAT)	N/A	24.81 (SAT)
puzzle_12_12_3	58.81 (SAT)	1.84 (SAT)	N/A	3.85 (SAT)
puzzle_12_12_4	0.09 (UNSAT)	0.72 (UNSAT)	N/A	0.28 (UNSAT)
rand_10_10_n5_1	90.22 (UNSAT)	287.25 (UNSAT)	N/A	518.52 (UNSAT)
rand_10_10_n5_2	131.27 (UNSAT)	14.11 (UNSAT)	N/A	72.91 (UNSAT)
rand_10_10_n5_3	54.74 (UNSAT)	104.72 (UNSAT)	N/A	106.97 (UNSAT)
rand_10_10_n5_4	153.76 (UNSAT)	289.89 (UNSAT)	N/A	881.90 (UNSAT)
rand_10_10_n5_5	18.83 (UNSAT)	52.21 (UNSAT)	N/A	70.55 (UNSAT)
rand_10_10_n5_6	0.05 (UNSAT)	0.31 (UNSAT)	N/A	0.11 (UNSAT)
rand_10_10_n5_7	135.01 (UNSAT)	277.87 (UNSAT)	N/A	818.98 (UNSAT)
rand_10_10_n5_8	93.76 (UNSAT)	157.11 (UNSAT)	N/A	517.34 (UNSAT)
rand_10_10_n5_9	255.49 (UNSAT)	385.82 (UNSAT)	N/A	—— (?)
rand_10_10_n5_10	124.18 (UNSAT)	226.53 (UNSAT)	N/A	361.99 (UNSAT)
rand_10_10_n6_1	12.18 (UNSAT)	24.18 (UNSAT)	N/A	69.37 (UNSAT)
rand_10_10_n6_2	154.63 (UNSAT)	99.38 (UNSAT)	N/A	275.44 (UNSAT)
rand_10_10_n6_3	174.78 (UNSAT)	198.42 (UNSAT)	N/A	—— (?)
rand_10_10_n6_4	190.39 (UNSAT)	168.76 (UNSAT)	N/A	—— (?)
rand_10_10_n6_5	158.91 (UNSAT)	104.35 (UNSAT)	N/A	457.56 (UNSAT)
rand_10_10_n6_6	314.21 (UNSAT)	739.65 (UNSAT)	N/A	—— (?)
rand_10_10_n6_7	607.40 (UNSAT)	—— (?)	N/A	—— (?)
rand_10_10_n6_8	0.06 (UNSAT)	0.23 (UNSAT)	N/A	0.12 (UNSAT)
rand_10_10_n6_9	395.40 (UNSAT)	541.80 (UNSAT)	N/A	—— (?)
rand_10_10_n6_10	511.52 (UNSAT)	807.27 (UNSAT)	N/A	—— (?)
rand_12_12_n7_1	—— (?)	—— (?)	N/A	—— (?)
rand_12_12_n7_2	270.16 (UNSAT)	230.20 (UNSAT)	N/A	910.31 (UNSAT)
rand_12_12_n7_3	—— (?)	—— (?)	N/A	—— (?)
rand_12_12_n7_4	0.10 (UNSAT)	0.69 (UNSAT)	N/A	0.30 (UNSAT)
rand_12_12_n7_5	—— (?)	—— (?)	N/A	—— (?)
rand_12_12_n7_6	—— (?)	—— (?)	N/A	—— (?)
rand_12_12_n7_7	554.98 (UNSAT)	771.46 (UNSAT)	N/A	—— (?)
rand_12_12_n7_8	0.10 (UNSAT)	0.71 (UNSAT)	N/A	0.29 (UNSAT)
rand_12_12_n7_9	—— (?)	38(?)	N/A	—— (?)
rand_12_12_n7_10	—— (?)	(?)	N/A	—— (?)

Table 3: Nurikabe (hand coded and randomly generated).

Instances	GROCv2 + Z3	CLASP	DLV	CMODELS
rand_100_400_1159666138_1	0.28 (UNSAT)	0.04 (UNSAT)	0.05 (UNSAT)	0.56 (UNSAT)
rand_100_400_1159666138_2	0.34 (UNSAT)	0.05 (UNSAT)	0.05 (UNSAT)	0.27 (UNSAT)
rand_100_400_1159666138_3	0.32 (UNSAT)	0.04 (UNSAT)	0.06 (UNSAT)	0.18 (UNSAT)
rand_100_400_1159666138_5	0.27 (UNSAT)	0.05 (UNSAT)	0.04 (UNSAT)	0.47 (UNSAT)
rand_100_400_1159666138_6	0.29 (UNSAT)	0.06 (UNSAT)	0.05 (UNSAT)	0.58 (UNSAT)
rand_100_400_1159666138_7	0.22 (UNSAT)	0.04 (UNSAT)	0.04 (UNSAT)	0.44 (UNSAT)
rand_100_400_1159666138_8	0.44 (UNSAT)	0.07 (UNSAT)	0.06 (UNSAT)	0.66 (UNSAT)
rand_100_400_1159666138_9	0.51 (UNSAT)	0.05 (UNSAT)	0.09 (UNSAT)	0.54 (UNSAT)
rand_100_400_1159666138_13	0.33 (UNSAT)	0.04 (UNSAT)	0.05 (UNSAT)	0.50 (UNSAT)
rand_100_400_1159666138_19	0.32 (UNSAT)	0.06 (UNSAT)	0.09 (UNSAT)	0.79 (UNSAT)
rand_150_600_1159731678_3	0.81 (UNSAT)	0.12 (UNSAT)	0.11 (UNSAT)	2.40 (UNSAT)
rand_100_400_1159666138_5	0.78 (UNSAT)	0.08 (UNSAT)	0.09 (UNSAT)	1.87 (UNSAT)
rand_100_400_1159666138_6	0.76 (UNSAT)	0.09 (UNSAT)	0.11 (UNSAT)	2.07 (UNSAT)
rand_100_400_1159666138_7	1.15 (UNSAT)	0.10 (UNSAT)	0.10 (UNSAT)	1.97 (UNSAT)
rand_100_400_1159666138_8	0.76 (UNSAT)	0.09 (UNSAT)	0.11 (UNSAT)	1.89 (UNSAT)
rand_100_400_1159666138_11	0.76 (UNSAT)	0.08 (UNSAT)	0.11 (UNSAT)	1.85 (UNSAT)
rand_100_400_1159666138_12	0.73 (UNSAT)	0.22 (UNSAT)	0.11 (UNSAT)	1.95 (UNSAT)
rand_100_400_1159666138_14	0.76 (UNSAT)	0.08 (UNSAT)	0.10 (UNSAT)	0.53 (UNSAT)
rand_100_400_1159666138_15	0.72 (UNSAT)	0.08 (UNSAT)	0.11 (UNSAT)	1.89 (UNSAT)
rand_100_400_1159666138_17	0.79 (UNSAT)	0.09 (UNSAT)	0.12 (UNSAT)	2.49 (UNSAT)

Table 4: Weight-Bounded Connected Dominating Set (from ASPARAGUS instances)

5. Related Work

As a crucial building block of answer set programs, aggregates are extensively studied in the literature [4, 10, 13, 16, 21, 22, 23]. Although the syntactic form of aggregates is usually presented in a first-order language, its semantics is normally defined propositionally via grounding. Recently, several approaches are proposed to define a genuine first-order semantics for aggregates. An early attempt is due to Lee and Meng [22], although a more restricted form only for the choice and counting aggregates already appeared in [21]. In fact, our semantics for first-order aggregates is essentially equivalent to Lee and Meng's definition, when restricted into the syntax with aggregates of the form (1) although there is a slight difference. Lee and Meng's stable model semantics considers the number constant symbols from \mathbb{Z} as part of the signature so that the domains of the underlying structure are assumed to contain the numbers from \mathbb{Z} . On the contrary, our notion does not consider numbers in the domain of a structure \mathcal{A} . Instead, we use a partial function $f_{\mathcal{A}}$ that maps the domain elements of $Dom(\mathcal{A})$ to integers. This is addressed in Definition 3 by the notion of an *extended structure*. The only difference between extended structures and standard first-order structures is the function $f_{\mathcal{A}}$ that maps elements of $Dom(\mathcal{A})$ to \mathbb{Z} . It should be noted that our notion of an extended structure generalizes the notion by that of Lee and Meng [22] since we can always map domain elements with number symbols into the numbers they represent, e.g., if $Dom(\mathcal{A})$ contains the number symbols -1 and 3, then we can simply set $f_{\mathcal{A}}(-1) = -1$ and $f_{\mathcal{A}}(3) = 3$, where $f_{\mathcal{A}}$ will be partially fixed for the number symbols from \mathbb{Z} for all structures \mathcal{A} . The reason for our notion is that it is more abstract in the sense that it does not consider number symbols in the meta-level. Furthermore, it also allows us to only consider finite domains since Lee and Meng's notion always assumes all numbers from \mathbb{Z} to be presented in the domain of all underlying structures.

An alternative definition of aggregates is the FLP semantics [13], which is extended into the first-order case recently [4]. The FLP semantics is inherently different from the stable model semantics for aggregates. For instance, the former satisfies the anti-chain property [13] while the latter does not. However, these two semantics coincide if we only allow positive atoms in aggregates, that is, $Ng(\delta) = \emptyset$ for any aggregate atoms of the form (1). In fact, this is the case for many ASP benchmark programs, including the benchmark programs we tested in Section 4.

Another aggregate framework that is related to our work is the SP semantics of [30]. The SP semantics of normal programs with aggregates is defined via a

fixed point operator that behaves identically to the three-valued immediate consequence operator " Φ_P^{agg} " independently proposed in [26]. Although the semantics itself is based on a fixed point type characterization, which is different from our translational approach, this framework is related to our work in the sense that our aggregate syntax of the form (1) is similar to their notion of an *intensional multiset* of the form

$$\{\!\!\{x \mid \exists z_1, \ldots, z_r P(y_1, \ldots, y_m)\}\!\!\},\$$

where $\{x, z_1, \ldots, z_r\} \subseteq y_1, \ldots, y_m$ (see Definition 1 of [30]).

Aggregates are also defined for HEX programs [10], which is an extension of logic programs by introducing higher order external atoms. The semantics of aggregates in HEX programs is also defined in a similar manner as the FLP semantics. Interestingly, it is shown that external atoms in HEX programs generalizes the notion of aggregate atoms [10].

Our definition of convex aggregates is a lift of Liu and Truszczyński's notion [23] to the first-order case, although the syntax is presented quite differently. In fact, Liu and Truszczyński observed that normal programs with polynomial, convex aggregates does not increase the computational complexity. For instance, checking the existence of answer sets for normal programs and normal programs with polynomial, convex aggregates are both in NP. However, the negative result that convex aggregate is the maximal subclass for the above property is not considered. In other words, we discover that normal programs with non-convex aggregates will inevitably jump into another complexity level. From Lemma 2, normal programs with non-convex aggregates can capture disjunctive programs, thus checking the existence of answer sets for such programs is Σ_2^P complete.

Our negative result (see Lemma 2 and Theorem 2) is inspired by Ferraris' work [16]. Ferraris' work considered monotone and antimonotone aggregates but not the generalized concept of convex aggregates. He showed that, in the propositional case, there exists an aggregate context such that disjunctive programs can be converted into normal logic programs with aggregate atoms under this context. We further proved that this is the case for all non-convex aggregate context, both in the propositional case and in the first-order case. As a consequence, convex aggregates exactly draws a boundary of the expressive power as well as computational complexity of aggregates in normal logic programs. That is, in the propositional case, the complexity of checking the answer set existence for normal logic programs with any convex aggregate atoms is *NP*-complete, which is the same as that for normal logic program without aggregates. In contrast, the

complexity of checking the answer set existence for normal logic programs under any non-convex aggregate context is Σ_2^P -complete, which is on a higher complexity level. In the first-order case, our result shows that normal logic programs with any convex aggregate atoms can be converted into first-order logic with the same type of aggregates (see Theorem 1), while this can never be done for normal logic programs under any non-convex aggregate context providing some general assumptions in the complexity theory (see Theorem 2).

Interestingly, Alviano and Faber [1] recently have obtained a similar result for the FLP semantics in the propositional case. That is, under the FLP semantics, propositional programs with convex aggregates will not increase the complexity. However, adding a simple non-convex aggregates will result in a complexity jump from the first level of polynomial hierarchy to the second level. In this paper, we are mainly focused on the Ferraris' semantics but not the FLP semantics. Nevertheless, following the proofs, our results hold for the FLP semantics as well. The main difference is that we consider the first-order case instead of the propositional case. We show that convex aggregates draws the boundary not only on complexity but also on expressiveness (i.e., translatability to classic first-order logic with aggregates) as well.

Finally, we would address the relation of our work with the *first-order loop formulas* approach of Lee and Meng [22]. A main difference is that first-order loop formulas work for arbitrary aggregates while our enhanced ordered completion only works for convex aggregates. This is partially because first-order loop formulas are infinite theories in general while ordered completions always produce finite theories. From a semantic viewpoint, loop formulas is about the encoding of the external support of a set of atoms (which is infinite at the first-order level), while the ordered completion is about the encoding of a derivation order, although they both imply the stability of a model for the class of normal program with convex aggregates.

6. Conclusion

The main contributions of this paper are as follows. Firstly, we extended the notion of ordered completion for first-order programs with aggregates, and showed that the stable models of a program with convex aggregates are corresponding to the models of its ordered completion on finite structures (Theorem 1). This is an important extension as most ASP benchmark programs for real world problems need to use aggregates and these aggregates are indeed convex. Secondly, we showed that convex aggregate is a maximal subclass for the above task. More precisely, given a non-convex aggregate context, we can always construct a normal program under this aggregate context that can never be translated into a first-order sentence with the same type of aggregates providing some assumptions in the complexity theory (Theorem 2). To the best of our knowledge, our negative result is one of the first two results [1] to show that convex aggregates exactly draws a boundary of the expressive power as well as the computational complexity of aggregate atoms in answer set programming. Finally, we showed that we can ground a program with an extensional database into a propositional SMT theory in order to compute the answer sets (Theorem 3). Based on this, we implement an ASP solver and compare it with some modern ASP solvers. The experiments show that this new direction of answer set solving is promising, particularly for non-tight programs on large problem instances.

Acknowledgement

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is partially supported by the grant NSFC 61173010.

References

- [1] Mario Alviano and Wolfgang Faber. The complexity boundary of answer set programming with generalized atoms under the flp semantics. In *LPNMR*, pages 67–72, 2013.
- [2] Vernon Asuncion, Fangzhen Lin, Yan Zhang, and Yi Zhou. Ordered completion for first-order logic programs on finite structures. *Artif. Intell.*, 177-179:1–24, 2012.
- [3] Chitta Baral. *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press, 2003.
- [4] Michael Bartholomew, Joohyung Lee, and Yunsong Meng. First-order extension of the FLP stable model semantics via modified circumscription. In *IJCAI-2011*, pages 724–730, 2011.
- [5] Francesco Calimeri, Giovambattista Ianni, Francesco Ricca, Mario Alviano, Annamaria Bria, Gelsomina Catalano, Susanna Cozza, Wolfgang Faber, Onofrio Febbraro, Nicola Leone, Marco Manna, Alessandra Martello, Claudio Panetta, Simona Perri, Kristian Reale, Maria Carmela Santoro, Marco

Sirianni, Giorgio Terracina, and Pierfrancesco Veltri. The third answer set programming competition: Preliminary report of the system competition track. In *LPNMR*, pages 388–403, 2011.

- [6] Yin Chen, Fangzhen Lin, Yisong Wang, and Mingyi Zhang. First-order loop formulas for normal logic programs. In *KR*, pages 298–307, 2006.
- [7] Keith L. Clark. Negation as failure. In *Logics and Databases*, pages 293– 322, 1978.
- [8] Minh Dao-Tran, Thomas Eiter, Michael Fink, and Thomas Krennwallner. Modular nonmonotonic logic programming revisited. In *ICLP*, pages 145– 159, 2009.
- [9] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Expressive power and complexity of disjunctive datalog under the stable model semantics. In *IS/KI*, pages 83–103, 1994.
- [10] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *IJCAI*, pages 90–96, 2005.
- [11] Wolfgang Faber and Nicola Leone. On the complexity of answer set programming with aggregates. In *LPNMR*, pages 97–109, 2007.
- [12] Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Aggregate functions in DLV. In Answer Set Programming: Advances in Theory and Implementation, pages 274–288, 2003.
- [13] Wolfgang Faber, Gerald Pfeifer, and Nicola Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artif. Intell.*, 175(1):278–298, 2011.
- [14] Wolfgang Faber, Gerald Pfeifer, Nicola Leone, Tina Dell'Armi, and Giuseppe Ielpa. Design and implementation of aggregate functions in the DLV system. *TPLP*, 8(5-6):545–580, 2008.
- [15] Ronal Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Proceedings of SIAM-AMS* 7, pages 27–41, 1974.
- [16] Paolo Ferraris. Logic programs with propositional connectives and aggregates. ACM Transactions on Computational Logic, 12(4):25, 2011.

- [17] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription. *Artif. Intell.*, 175(1):236–263, 2011.
- [18] Paolo Ferraris and Vladimir Lifschitz. On the stable model semantics of first-order formulas with aggregates. In *Proceedings of the 2010 Workshop on Nonmonotonic Reasoning*, 2010.
- [19] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. On the implementation of weigh constraints in conflict-driven ASP solvers. In *ICLP'09*, volume 5649, pages 250–264. Springer Verlag, 2009.
- [20] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press, 1988.
- [21] Joohyung Lee, Vladimir Lifschitz, and Ravi Palla. A reductive semantics for counting and choice in answer set programming. In AAAI, pages 472–479, 2008.
- [22] Joohyung Lee and Yunsong Meng. On reductive semantics of aggregates in answer set programming. In LPNMR, pages 182–195, 2009.
- [23] Lengning Liu and Miroslaw Truszczynski. Properties and applications of programs with monotone and convex constraints. J. Artif. Intell. Res. (JAIR), 27:299–334, 2006.
- [24] Victor W. Marek and Miroslaw Truszczynski. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a* 25-Year Perspective, pages 375–398. Springer-Verlag, 1999.
- [25] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. and AI*, 25(3-4):241–273, 1999.
- [26] Nikolay Pelov, Marc Denecker, and Maurice Bruynooghe. Partial stable models for logic programs with aggregates. In *LPNMR*, pages 207–219, 2004.
- [27] Nikolay Pelov, Marc Denecker, and Maurice Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. *TPLP*, 7(3):301– 353, 2007.

- [28] Enrico Pontelli, Tran Cao Son, and Islam Elkabani. A treatment of aggregates in ASP (system description). In *LPNMR-2004*, pages 356–360. Springer, 2004.
- [29] Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artif. Intell.*, 138(1-2):181–234, 2002.
- [30] Tran Cao Son and Enrico Pontelli. A constructive semantic characterization of aggregates in answer set programming. *TPLP*, 7(3):355–375, 2007.
- [31] Tran Cao Son, Enrico Pontelli, and Phan Huy Tu. Answer sets for logic programs with arbitrary abstract constraint atoms. J. Artif. Intell. Res. (JAIR), 29:353–389, 2007.

Appendix A. Proof of Lemma 2

We present the full proof of Lemma 2 in this section. In the following, we always assume a non-convex aggregate context \mathcal{AC} and a disjunctive program Γ_D . In Appendix A.1, we will define a translation from Γ_D to a normal program Γ_N with the same type of aggregates. Lemma 2 can be decomposed into the following two lemmas.

Lemma 3. Let \mathcal{M}_N be a structure on $\tau(\Gamma_N)$. If \mathcal{M}_N is a stable model of Γ_N , then there exists a stable model \mathcal{M}_D of Γ_D such that \mathcal{M}_N and \mathcal{M}_D agree on all interpretations of constants and predicates in $\tau(\Gamma_D)$.

Lemma 4. Let \mathcal{M}_D be a structure on $\tau(\Gamma_D)$. If \mathcal{M}_D is a stable model of Γ_D , then there exists a stable model \mathcal{M}_N of Γ_N such that \mathcal{M}_D and \mathcal{M}_N agree on all interpretations of constants and predicates in $\tau(\Gamma_D)$.

Lemma 3 will be shown in Appendix A.2, and Lemma 4 will be shown in Appendix A.3.

We first introduce some notations. Recall that the aggregate context \mathcal{AC} is nonconvex. By Definition 6, there exist three multisets of tuples M_1, M_2, M_3 such that $M_1 \subseteq M_2$, $OP(M_1) \preceq N$ and $OP(M_3) \preceq N$ hold while $OP(M_1) \preceq N$ does not hold, where $OP \in \mathcal{AC}_{ag}$ is an aggregate symbol, $\preceq \in \mathcal{AC}_{op}$ is a comparison operator, and $N \in \mathcal{AC}_{num}$ is a number. Let

- $M_1 = \{\!\!\{\mathbf{m}_1, \ldots, \mathbf{m}_{N_1}\}\!\!\},\$
- $M_2 \setminus M_1 = \{\{\mathbf{m}_{N_1+1}, \dots, \mathbf{m}_{N_2}\}\},\$

•
$$M_3 \setminus M_2 = \{\{\mathbf{m}_{N_2+1}, \ldots, \mathbf{m}_{N_3}\}\},\$$

where N_1, N_2, N_3 be the size of M_1, M_2, M_3 respectively, and $0 \le N_1 < N_2 < N_3$. Without loss of generality, we assume all tuples in M_3 are of the same length K.

We introduce some new constants. Let C_{new} be a set of new constants

$$C_{new} = \{c_{m,s} \mid \mathbf{m}_s \in M_3, m \in \mathbf{m}_s, (1 \le s \le N_3)\} \cup \{cn, cc\}.$$

where cn is a new constant for N, cc is a special new constant which is used in rule (A.12) for Proposition 5, and the rest of new constants are for the tuples of numbers in M_1, M_2, M_3 . We use D to denote the following set of tuples of constants

$$D = \{\mathbf{d}_1, \ldots, \mathbf{d}_{N_1}, \ldots, \mathbf{d}_{N_2}, \ldots, \mathbf{d}_{N_3}\},\$$

where \mathbf{d}_s $(1 \leq s \leq N_3)$ is the tuple of constants obtained from $\mathbf{m}_s \in M_3$ by replacing each $m \in \mathbf{m}_s$ by $c_{m,s}$. Notice that \mathbf{d}_i and \mathbf{d}_j , $(1 \leq i \neq j \leq N_3)$, are always different tuple of constants. We use $NotLitNew(\mathbf{x})$ to denote the set of negated atoms

{not
$$x = c \mid x \in \mathbf{x}, c \in \mathcal{C}_{new} \setminus \{cc\}$$
},

where x is a tuple of variables. Intuitively, NotLitNew(x) means that variables in x and constants in $C_{new} \setminus \{cc\}$ will never be mapped to the same domain elements.

We introduce some new predicates for the program. Let \mathcal{P}_{new} be a set of new predicates

$$\mathcal{P}_{new} = \mathcal{Q} \cup \mathcal{QD} \cup \mathcal{QM},$$

where

$$\mathcal{Q} = \{Q_{r,i,j} \mid r \in \Gamma \text{ is a rule of the form (27)}, 1 \le i \ne j \le k\},\$$
$$\mathcal{QD} = \{QD_i \mid 1 \le i \le K\},\$$
$$\mathcal{QM} = \{QM_i \mid i = 1, 2, 3\}.$$

Intuitively, Q is a set of predicates for constructing aggregate atoms, QD for representing elements in a tuple, and QM for encoding M_1, M_2, M_3 .

Furthermore, we introduce some new predicates and tuples of predicates for the second-order formulas like (8). Let U be a set of new predicates

$$\mathcal{U} = \{U_P \mid P \in \mathcal{P}_{int}(\Gamma_D) \cup \mathcal{P}_{new}, U_P \text{ and } P \text{ have the same arity.}\}$$

We use \mathbf{P}_D and \mathbf{U}_D to denote the tuple of predicates in $\mathcal{P}_{int}(\Gamma_D)$ and the tuple of the correspondence predicates in $\{U_P \mid P \in \mathcal{P}_{int}(\Gamma_D)\}$ respectively. We use \mathbf{P}_N and \mathbf{U}_N to denote the tuple of predicates in $\mathcal{P}_{int}(\Gamma_D) \cup \mathcal{P}_{new}$ and the tuple of the correspondence predicates in \mathcal{U} respectively.

Appendix A.1. The Translation from Disjunctive Programs to Normal Programs with Non-convex Aggregates

In this section, we define a normal program with aggregates, denoted by Γ_N , which captures the answer sets of Γ_D . In general, Γ_N is a normal program with aggregates built from $\tau(\Gamma_D)$ together with the set of new constants C_{new} and the set of new predicates \mathcal{P}_{new} . It is the union of three sets of rules:

$$\Gamma_N = \mathsf{AGG} \cup \mathsf{DEF} \cup \mathsf{CST},$$

where

- AGG is a set of normal rules with aggregates, which simulates the disjunctive rules;
- DEF is a set of rules about the new predicates in Q;
- CST is a set of rules about the new predicates in QD and QM.

Now we define the sets of rules AGG, DEF and CST. Without loss of generality, we assume that there is no constant in the heads of the rules in Γ_D . Otherwise, we remove the constants in the head of a rule by using equalities. For example, $P(x,c) \leftarrow E(x,y)$ can be rewritten as $P(x,z) \leftarrow E(x,y), z = c$ when c is a constant.

Let $r \in \Gamma_D \setminus \Gamma_D^{\perp}$ be a rule of the form (27). Let $\mathbf{v}_1, \ldots, \mathbf{v}_k$ be the tuples of variables occurring in $\alpha_1, \ldots, \alpha_k$ in the head of r. By r_i^N , we denote the rule:

$$\alpha_{i} \leftarrow \beta_{1}, \dots, \beta_{l}, \operatorname{not} \gamma_{1}, \dots, \operatorname{not} \gamma_{m},$$

$$OP\langle \mathbf{x} : Q_{r,i,1}(\mathbf{x}, \mathbf{v}_{i}, \mathbf{v}_{1}) \rangle \preceq cn,$$

$$\dots,$$

$$OP\langle \mathbf{x} : Q_{r,i,k}(\mathbf{x}, \mathbf{v}_{i}, \mathbf{v}_{k}) \rangle \preceq cn,$$

$$NotLitsNew(Var(r)),$$
(A.1)

where:

- $1 \le i \le k$, and k is the number of atoms in the head of r;
- x is a tuple of distinct new variables and $|\mathbf{x}| = K$;
- Var(r) is the tuple of variables in r.

We will show that, together with the rules in DEF and CST, normal rules r_1^N, \ldots, r_k^N exactly capture the disjunctive rule r. Let AGG be the following set of rules:

$$\Gamma_D^{\perp} \cup \{r_i^N \mid r \in \Gamma_D \setminus \Gamma_D^{\perp} \text{ is a rule of the form (27) }, (1 \le i \le k)\}.$$
 (A.2)

Then, we define the rules in DEF. Let r be a rule of the form (27), and assume that $\alpha_i = P_i(\mathbf{v}_i)$ and $\alpha_j = P_j(\mathbf{v}_j)$. For every predicate $Q_{r,i,j} \in \mathcal{Q}$, we use $\mathsf{DEF}(Q_{r,i,j})$ to denote the set of rules:

$$Q_{r,i,j}(\mathbf{d}_s, \mathbf{v}_i, \mathbf{v}_j) \leftarrow NotLitsNew(\mathbf{v}_i \mathbf{v}_j), \text{ for } 1 \le s \le N_1,$$
(A.3)

$$Q_{r,i,j}(\mathbf{d}_s, \mathbf{v}_i, \mathbf{v}_j) \leftarrow P_j(\mathbf{v}_j), NotLitsNew(\mathbf{v}_i), \text{ for } N_1 < s \le N_2,$$
(A.4)

$$Q_{r,i,j}(\mathbf{d}_s, \mathbf{v}_i, \mathbf{v}_j) \leftarrow P_i(\mathbf{v}_i), P_j(\mathbf{v}_j), \text{ for } N_2 < s \le N_3,$$
(A.5)

$$\leftarrow Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j), \operatorname{not} P_j(\mathbf{v}_j), \operatorname{not} QD_1(\mathbf{x}), \dots, \operatorname{not} QD_{N_1}(\mathbf{x}),$$
(A.6)

$$\leftarrow Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j), \text{ not } P_i(\mathbf{v}_i), \text{ not } QD_1(\mathbf{x}), \dots, \text{ not } QD_{N_2}(\mathbf{x}),$$
(A.7)

$$\leftarrow Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j), \operatorname{not} QD_1(\mathbf{x}), \dots, \operatorname{not} QD_{N_3}(\mathbf{x}),$$
(A.8)

Let DEF be the following set of rules:

$$\bigcup_{Q_{r,i,j} \in \mathcal{Q}} \mathsf{DEF}(Q_{r,i,j}).$$
(A.9)

Finally, let CST be the set of the following rules:

• for predicate $P \in \tau(\Gamma_D), c \in \mathcal{C}_{new} \setminus \{cc\}$ and $1 \le i \le j$, $\leftarrow P(x_1, \dots, x_s), x_i = c$, (A.10)

where j is the arity of P;

• for constant $c_1 \in \tau(\Gamma_D)$ and $c_2 \in \mathcal{C}_{new} \setminus \{cc\}$,

$$\leftarrow c_1 = c_2, \tag{A.11}$$

• for
$$c \in \mathcal{C}_{new} \setminus \{cc\}$$
,

$$\leftarrow c = cc, \tag{A.12}$$

• for two different constants $c_1, c_2 \in \mathcal{C}_{new} \setminus \{cn, cc\}$,

$$\leftarrow c_1 = c_2, \tag{A.13}$$

• $\leftarrow \operatorname{not} \operatorname{OP}\langle \mathbf{x} : QM_1(\mathbf{x}) \rangle \preceq cn,$ (A.14)

$$\leftarrow \operatorname{OP}\langle \mathbf{x} : QM_2(\mathbf{x}) \rangle \preceq cn, \tag{A.15}$$

$$\leftarrow \operatorname{not} \operatorname{OP}\langle \mathbf{x} : QM_3(\mathbf{x}) \rangle \preceq cn, \tag{A.16}$$

• for
$$1 \le i \le N_1$$
,
 $QM_1(\mathbf{d}_i) \leftarrow$, (A.17)

• for
$$1 \le i \le N_2$$
,
 $QM_2(\mathbf{d}_i) \leftarrow$, (A.18)

• for
$$1 \le i \le N_3$$
,

$$QM_3(\mathbf{d}_i) \leftarrow, \tag{A.19}$$

٠

$$\leftarrow QM_1(\mathbf{x}), \operatorname{not} QD_1(\mathbf{x}), \dots, \operatorname{not} QD_{N_1}(\mathbf{x}), \qquad (A.20)$$

$$\leftarrow QM_2(\mathbf{x}), \operatorname{not} QD_1(\mathbf{x}), \dots, \operatorname{not} QD_{N_2}(\mathbf{x}), \qquad (A.21)$$

$$\leftarrow QM_3(\mathbf{x}), \operatorname{not} QD_1(\mathbf{x}), \dots, \operatorname{not} QD_{N_3}(\mathbf{x}), \qquad (A.22)$$

• for $1 \leq i \leq N_3$,

$$QD_i(\mathbf{d}_i) \leftarrow,$$
 (A.23)

• for $1 \le i \le N_3$ and $1 \le j \le K$,

$$\leftarrow QD_i(x_1, \dots, x_j, \dots, x_K), \operatorname{not} x_j = c_{\mathbf{m}_i, j}, \qquad (A.24)$$

where $\mathbf{m}_i = (m_1, \ldots, m_j, \ldots, m_K)$ is a tuple in M_3 .

Given a disjunctive program Γ_D , the number of the rules of Γ_N is polynomial, and the length of each rule is also polynomial. So, Γ_N can be built in polynomial time with respect to the length of Γ_D .

Next, we prove some propositions which could illustrate why we define these rules. Proposition 2 shows the properties of the sets of new predicates QD and QM. Proposition 3 and Proposition 4 show that the implication can also be simulated by aggregate atoms when the aggregate context is non-convex.

Let \mathcal{M} be a structure on $\tau(\Gamma_N)$. We use $Dom_1(\mathcal{M})$ and $Dom_2(\mathcal{M})$ to denote two subsets of $Dom(\mathcal{M})$ such that

$$Dom_2(\mathcal{M}) = \{ c^{\mathcal{M}} \mid c \in \mathcal{C}_{new} \setminus \{cc\} \},\$$
$$Dom_1(\mathcal{M}) = Dom(\mathcal{M}) \setminus Dom_2(\mathcal{M}).$$

We have the following proposition.

Proposition 2. Let \mathcal{M} be a structure on $\tau(\Gamma_N)$ such that $\mathcal{M} \models \mathsf{DEF} \cup \mathsf{CST}$. Then,

- (1) For predicate $P \in \tau(\Gamma_D)$, if $\mathbf{a} \in P^{\mathcal{M}}$ then $\mathbf{a} \in Dom_1(\mathcal{M})^{|\mathbf{a}|}$.
- (2) For predicate QD_i , $(1 \le i \le N_3)$, $QD_i^{\mathcal{M}} = \{\mathbf{d}_i^{\mathcal{M}}\}$.
- (3) For predicate $QM_i (i = 1, 2, 3), QM_i^{\mathcal{M}} = \{\mathbf{d}_1^{\mathcal{M}}, \dots, \mathbf{d}_{N_i}^{\mathcal{M}}\}.$

Proof: \mathcal{M} is a model of rule (A.10), so (1) holds. \mathcal{M} is a model of rules (A.23) and (A.24), so (2) holds. \mathcal{M} is a model of rules (A.17) - (A.19) and (A.20) - (A.22), so (3) holds. \Box

Proposition 3. Let \mathcal{M} be a structure on $\tau(\Gamma_N)$ such that $\mathcal{M} \models \mathsf{DEF} \cup \mathsf{CST}$. Then,

$$\mathcal{M} \models (P_j(\mathbf{v}_j) \to P_i(\mathbf{v}_i)) \leftrightarrow (\operatorname{OP}\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn)[\mathbf{v}_1 \mathbf{v}_j / \mathbf{bc}]$$

where

- $r \in \Gamma_D$ be a rule of the form (27),
- $\alpha_i = P_i(\mathbf{v}_i)$ and $\alpha_j = P_j(\mathbf{v}_j)(1 \le i \ne j \le k)$ are two atoms in the head of r,
- $\mathbf{b} \in Dom(\mathcal{M})^{|\mathbf{b}|}$ and $\mathbf{c} \in Dom(\mathcal{M})^{|\mathbf{c}|}$.

Proof: The proposition is a direct consequence of the following three statements:

- (1) $\mathcal{M} \models \neg P_j(\mathbf{v}_j) \to (\operatorname{OP}\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}];$
- (2) $\mathcal{M} \models (\neg P_i(\mathbf{v}_i) \land P_j(\mathbf{v}_j)) \to (\operatorname{OP}\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \not\preceq cn)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}];$ and
- (3) $\mathcal{M} \models (P_i(\mathbf{v}_i) \land P_j(\mathbf{v}_j)) \to (OP\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$

(1) Assume that $\mathcal{M} \models \neg P_j(\mathbf{v}_j)[\mathbf{v}_1\mathbf{v}_j/\mathbf{bc}]$. Let $\mathbf{a} \in Dom(\mathcal{M})^K$ be a tuple of domain elements. By rules (A.3) and (A.6), $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$ if and only if there exists $1 \leq s \leq N_1$ such that $\mathcal{M} \models QD_s(\mathbf{x})[\mathbf{x}/\mathbf{a}]$. Furthermore, by rules (A.17) and (A.20), $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$ if and only if $\mathbf{a} \in QM_1^{\mathcal{M}}$. We also have

$$\mathcal{M} \models \operatorname{OP}\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$$
(A.25)

if and only if

$$\mathcal{M} \models \operatorname{OP}\langle x : QM_1(\mathbf{x}) \rangle \preceq cn. \tag{A.26}$$

So, both rule (A.25) and rule (A.26) hold, since \mathcal{M} is a model of rule (A.14).

(2) Assume that $\mathcal{M} \models (\neg P_i(\mathbf{v}_i \land P_j(\mathbf{v}_j))[\mathbf{v}_1\mathbf{v}_j/\mathbf{bc}]$. Let $\mathbf{a} \in Dom(\mathcal{M})^K$ be a tuple of domain elements. By rules (A.3), (A.4) and (A.7), $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$ if and only if there exists $1 \leq s \leq N_2$ such that $\mathcal{M} \models QD_s(\mathbf{x})[\mathbf{x}/\mathbf{a}]$. Furthermore, by rules (A.18) and (A.21), $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$ if and only if $\mathbf{a} \in QM_2^{\mathcal{M}}$. We have

$$\mathcal{M} \models \operatorname{OP}\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$$
(A.27)

if and only if

$$\mathcal{M} \models \operatorname{OP}\langle x : QM_2(\mathbf{x}) \rangle \preceq cn.$$
(A.28)

So, neither rule (A.27) nor rule (A.28) holds, since \mathcal{M} is a model of rule (A.15).

(3) Assume that $\mathcal{M} \models (P_i(\mathbf{v}_i \wedge P_j(\mathbf{v}_j))[\mathbf{v}_1\mathbf{v}_j/\mathbf{bc}]$. Let $\mathbf{a} \in Dom(\mathcal{M})^K$ be a tuple of domain elements. By rules (A.3) - (A.5) and (A.8), $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$ if and only if there exists $1 \leq s \leq N_3$ such that $\mathcal{M} \models QD_s(\mathbf{x})[\mathbf{x}/\mathbf{a}]$. Furthermore, by rule (A.19) and (A.22), $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$ if and only if $\mathbf{a} \in QM_3^{\mathcal{M}}$. We have

$$\mathcal{M} \models \operatorname{OP}\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$$
(A.29)

if and only if

$$\mathcal{M} \models \operatorname{OP}\langle x : QM_3(\mathbf{x}) \rangle \preceq cn. \tag{A.30}$$

So, both rule (A.29) and rule (A.30) hold, since \mathcal{M} is a model of rule (A.16).

Let \mathcal{M} be a structure on $\tau(\Gamma_N)$ and \mathcal{U} a structure on $\tau(\Gamma_N) \cup \mathbf{U}_N$. \mathcal{U} is called a \mathcal{Q} -reserve extension of \mathcal{M} if \mathcal{U} is an extension of \mathcal{M} and

•
$$U_P^{\mathcal{U}} = U_P^{\mathcal{M}}$$
 if $U_P \in \mathbf{U}_N$ and $P \in \mathcal{QD} \cup \mathcal{QM}$,

• $\mathbf{abc} \in U_P^{\mathcal{U}}$ if and only if at least one of the following holds:

$$\mathbf{a} \in QM_1^{\mathcal{U}} \text{ and } \mathbf{bc} \in Dom_1(\mathcal{M})^{|\mathbf{bc}|},$$
 (A.31)

$$\mathbf{a} \in QM_2^{\mathcal{U}}, \mathbf{b} \in Dom_1(\mathcal{M})^{|\mathbf{b}|}, \text{ and } \mathbf{c} \in U_{P_i}^{\mathcal{U}},$$
 (A.32)

$$\mathbf{a} \in QM_2^{\mathcal{U}}, \mathbf{b} \in D \text{ onl}_1(\mathbf{b}, \mathbf{c})^{\mathcal{U}}, \text{ and } \mathbf{c} \in O_{P_j}^{\mathcal{U}},$$
(A.33)
$$\mathbf{a} \in QM_3^{\mathcal{U}}, \mathbf{b} \in U_{P_i}^{\mathcal{U}}, \text{ and } \mathbf{c} \in U_{P_j}^{\mathcal{U}}.$$

where

- *P* is a predicate of the form $Q_{r,i,j} \in \mathcal{Q}$, $r \in \Gamma_D$ is a rule of the form (27), and $\alpha_i = P_i(\mathbf{v}_i)$ and $\alpha_j = P_j(\mathbf{v}_j)$ are two atoms in the head of *r*,
- a, b, c are tuples of domain elements such that $|\mathbf{a}| = K$, and b and c match the arity of P_i and P_j respectively.

Proposition 4. Let \mathcal{M} be a structure on $\tau(\Gamma_N)$ such that $\mathcal{M} \models \mathsf{DEF} \cup \mathsf{CST}$. If \mathcal{U} is a \mathcal{Q} -reserve extension of \mathcal{M} then,

$$\mathcal{U} \models (U_{P_j}(\mathbf{v}_j) \to U_{P_i}(\mathbf{v}_i)) \leftrightarrow (\operatorname{OP}\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn)[\mathbf{v}_1 \mathbf{v}_j / \mathbf{bc}]$$

where

- $r \in \Gamma_D$ is a rule of form (27),
- $\alpha_i = P_i(\mathbf{v}_i)$ and $\alpha_j = P_j(\mathbf{v}_j)(1 \le i \ne j \le k)$ are two atoms in the head of r,
- $\mathbf{b} \in Dom(\mathcal{M})^{|\mathbf{b}|}$ and $\mathbf{c} \in Dom(\mathcal{M})^{|\mathbf{c}|}$.

Proof: The proof is similar to that of Proposition 3. We will show:

(1)
$$\mathcal{U} \models \neg U_{P_j}(\mathbf{v}_j) \to (\operatorname{OP}\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn) [\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$$

(2) $\mathcal{U} \models (\neg U_{P_i}(\mathbf{v}_i) \land U_{P_j}(\mathbf{v}_j)) \to (\operatorname{OP}\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \not\preceq cn) [\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$

(3)
$$\mathcal{U} \models (U_{P_i}(\mathbf{v}_i) \land U_{P_j}(\mathbf{v}_j)) \to (OP\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$$

(1) Assume that $\mathcal{M} \models \neg U_{P_j}(\mathbf{v}_j)[\mathbf{v}_1\mathbf{v}_j/\mathbf{bc}]$. Let $\mathbf{a} \in Dom(\mathcal{M})^K$ be a tuple of domain elements. By (A.31), if $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$ then $\mathbf{a} \in QM_1^{\mathcal{M}}$. We have

$$\mathcal{U} \models \operatorname{OP}\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$$
(A.34)

if and only if

$$\mathcal{U} \models \operatorname{OP}\langle x : U_{QM_1}(\mathbf{x}) \rangle \preceq cn.$$
(A.35)

So, both rule (A.34) and rule (A.35) hold, since \mathcal{U} is a model of rule (A.14) and $QM_1^{\mathcal{U}} = U_{QM_1}^{\mathcal{U}}$.

(2) Assume that $\mathcal{U} \models (\neg U_{P_i}(\mathbf{v}_i \wedge U_{P_j}(\mathbf{v}_j))[\mathbf{v}_1\mathbf{v}_j/\mathbf{bc}]$. Let $\mathbf{a} \in Dom(\mathcal{M})^K$ be a tuple of domain elements. By (A.31), if $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$, then $\mathbf{a} \in QM_2^{\mathcal{M}}$. We have

$$\mathcal{U} \models \operatorname{OP}\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$$
(A.36)

if and only if

$$\mathcal{U} \models \operatorname{OP}\langle x : U_{QM_2}(\mathbf{x}) \rangle \preceq cn.$$
(A.37)

So, neither rule (A.36) nor rule (A.37) holds, since \mathcal{U} is a model of rule (A.15), and $QM_2^{\mathcal{U}} = U_{OM_2}^{\mathcal{U}}$.

(3) Assume that $\mathcal{U} \models (U_{P_i}(\mathbf{v}_i \wedge U_{P_j}(\mathbf{v}_j))[\mathbf{v}_1\mathbf{v}_j/\mathbf{bc}]$. Let $\mathbf{a} \in Dom(\mathcal{M})^K$ be a tuple of domain elements. By (A.33), if $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}}$ then $\mathbf{a} \in QM_3^{\mathcal{M}}$. We have

$$\mathcal{U} \models \operatorname{OP}\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$$
(A.38)

if and only if

$$\mathcal{U} \models \operatorname{OP}\langle x : U_{QM_3}(\mathbf{x}) \rangle \preceq cn.$$
(A.39)

So, both rule (A.38) and rule (A.39) hold, since \mathcal{U} is a model of rule (A.16), and $QM_3^{\mathcal{U}} = U_{QM_3}^{\mathcal{U}}$. \Box

Appendix A.2. The Proof of Lemma 3

In this section, we will show that if \mathcal{M}_N is a stable model of Γ_N , then there exists a stable model \mathcal{M}_D of Γ_D such that \mathcal{M}_N and \mathcal{M}_D agree on all interpretations of constants and predicates in $\tau(\Gamma_D)$.

Let \mathcal{M}_N be a stable model of Γ_N . By \mathcal{M}_D , we denote the structure on $\tau(\Gamma_D)$ such that

- $Dom(\mathcal{M}_D) = Dom_1(\mathcal{M}_N),$
- $f_{\mathcal{M}_D}(c) = f_{\mathcal{M}_N}(c)$, for every constant $c \in Dom(\mathcal{M}_D)$,
- $c^{\mathcal{M}_D} = c^{\mathcal{M}_N}$, for every constant $c \in Dom(\mathcal{M}_D)$,
- $\mathbf{a} \in P^{\mathcal{M}_D}$ if and only if $\mathbf{a} \in P^{\mathcal{M}_N}$ and $\mathbf{a} \in Dom(\mathcal{M}_D)^{|\mathbf{a}|}$, for every predicate $P \in \tau(\Gamma_D)$.

Proposition 5 shows that \mathcal{M}_D is well defined in the sense that the domain of \mathcal{M}_D is not empty. Proposition 6 shows that \mathcal{M}_N and \mathcal{M}_D agree on the interpretations of all constants and predicates in $\tau(\Gamma_D)$.

Proposition 5. $Dom(\mathcal{M}_D)$ contains at least one domain elements.

Proof: \mathcal{M}_N is a model of rule (A.12). So cc is interpreted to a domain element different to any other constants in \mathcal{C}_{new} . Thus, $cc^{\mathcal{M}_N} \in Dom(\mathcal{M}_D)$. \Box

Proposition 6. Let $P \in \tau(\Gamma_D)$ be a predicate, and $\mathbf{a} \in Dom(\mathcal{M}_N)^{|\mathbf{a}|}$ a tuple of domain elements that matches the arity of P. Then, $\mathbf{a} \in P^{\mathcal{M}_D}$ if and only if $\mathbf{a} \in P^{\mathcal{M}_N}$.

Proof: By Proposition 2, if $\mathbf{a} \in P^{\mathcal{M}_N}$ then $\mathbf{a} \in Dom(\mathcal{M}_D)^{|\mathbf{a}|}$. So, by the construction of \mathcal{M}_D , $\mathbf{a} \in P^{\mathcal{M}_D}$ if and only if $\mathbf{a} \in P^{\mathcal{M}_N}$. \Box

Now, we present the proof of Lemma 3.

Proof: We first show that $\mathcal{M}_D \models \widehat{\Gamma_D}$. Otherwise, there exists a rule r of the form (27) and a substitution θ on Var(r) such that $\mathcal{M}_D \not\models \widehat{r}\theta$. So $\mathcal{M}_D \models Body(r)\theta$ and $\mathcal{M}_D \not\models \alpha_i \theta$, $(1 \leq i \leq k)$. We also have $\mathcal{M}_N \not\models \alpha_i \theta$, $(1 \leq i \leq k)$, by Proposition 6. Then, by Proposition 3, we have $\mathcal{M}_N \models OP\langle \mathbf{x} : Q_{r,i,j}\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j \rangle \rangle \leq cn\theta$, $(1 \leq i \neq j \leq k)$. When we consider the rule $r' \in \Gamma_N$ of the form (A.1), we have $\mathcal{M}_N \models Body(r')\theta$ and $\mathcal{M}_N \not\models Head(r')\theta$. This is a contradiction to the fact that \mathcal{M}_N is a stable model of Γ_N .

It remains to show

$$\mathcal{M}_D \models \neg \exists \mathbf{U}_D(\mathbf{U}_D < \mathbf{P}_D \land \bigwedge_{r \in \Gamma_D \setminus \Gamma_D^{\perp}} \widehat{r^*}).$$
(A.40)

We show this by contradiction. Otherwise, there exists a structure \mathcal{U}_D on $\tau(\Gamma_D) \cup U_D$ such that $\mathcal{U}_D \models U_D < \mathbf{P}_D$ and $\mathcal{U}_D \models \bigwedge_{r \in \Gamma_D} \widehat{r^*}$.

Let \mathcal{U}_N be a \mathcal{Q} -reserve extension of \mathcal{M}_N such that $U_P^{\mathcal{U}_N} = U_P^{\mathcal{U}_D}$ when P is an intensional predicate in Γ_D . In the following, we will show that

$$\mathcal{U}_N \models \mathbf{U}_N < \mathbf{P}_N \land \bigwedge_{r \in \Gamma_N \setminus \Gamma_N^{\perp}} \widehat{r^*}, \tag{A.41}$$

which is a contradiction to the fact that \mathcal{M}_N is a stable model of Γ_N .

- It suffices to show: (a) $U_N \models U_N < P_N$:
- (b) $\mathcal{U}_N \models r^*$, for rule $r \in \mathsf{CST}$,
- (c) $\mathcal{U}_N \models r^*$, for rule $r \in \mathsf{DEF}$,
- (d) $\mathcal{U}_N \models r^*$, for rule $r \in \mathsf{AGG}$.

Note that for (b)-(d), we do not need to consider the constraints.

(a) Consider the predicates P and U_P :

- If P is an intensional predicate in Γ_D , then $U_P^{\mathcal{U}_N} = U_P^{\mathcal{U}_D}$. We also have $\mathcal{U}_D \models \mathbf{U}_D < \mathbf{P}_D$. So we have $\mathcal{U}_N \models \mathbf{U}_D < \mathbf{P}_D$;
- If $P \in \mathcal{QD} \cup \mathcal{QM}$, then $U_P^{\mathcal{U}_N} = U_P^{\mathcal{M}_N}$. So we have $\mathcal{U}_N \models U_P = P$;
- If P is a predicate of the form Q_{r,i,j} ∈ Q, r ∈ Γ_D is a rule of the form (27), and α_i = P_i(**v**_i) and α_j = P_j(**v**_j) are two atoms in the head of r, we also have U_D ⊨ **U**_D < **P**_D. Notice that QM^{U_N}_i = QM^{M_N}_i, i = 1, 2, 3, and that P^{U_N}_i ⊆ P^{M_N}_i and P^{U_N}_j ⊆ P^{M_N}_j.

So we have $\mathcal{U}_N \models \mathbf{U}_N < \mathbf{P}_N$.

(b) Let r be a rule of the form (A.17), (A.18), (A.19), or (A.23), we have $\mathcal{U}_N \models r^*$, since $U_P^{\mathcal{U}_N} = U_P^{\mathcal{M}_N}$ for $P \in \mathcal{QD} \cup \mathcal{QM}$.

(c) Let r be a rule of the form (A.3). Let $\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}$ be an assignment on $Dom(\mathcal{U}_N)$, and \mathbf{d}_s a tuple in D, where $(1 \leq s \leq N_1)$. We have $\mathbf{d}_s^{\mathcal{U}_N} \in QM_1^{\mathcal{U}_N}$, since \mathcal{U}_N is a model of rules (A.17) and (A.20). If $\mathcal{U}_N \models NotLitsNew(\mathbf{v}_i \mathbf{v}_j)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$, we have $\mathbf{bc} \in Dom(\mathcal{M}_D^{|\mathbf{bc}|})$. By (A.31), $\mathcal{U}_N \models U_{Q_{r,i,j}}(\mathbf{d}_s, \mathbf{v}_i, \mathbf{v}_j)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$. Thus, $\mathcal{U}_N \models r^*$. Let r be a rule of the form (A.4). Let $\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}$ be an assignment on $Dom(\mathcal{U}_N)$, and \mathbf{d}_s a tuple in D, where $(N_1 < s \leq N_2)$. We have $\mathbf{d}_s^{\mathcal{U}_N} \in QM_2^{\mathcal{U}_N}$, since \mathcal{U}_N is a model of rules (A.18) and (A.21). If $\mathcal{U}_N \models U_{P_j}(\mathbf{v}_j) \wedge NotLitsNew(\mathbf{v}_i \mathbf{v}_j)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$, we have $\mathbf{b} \in Dom(\mathcal{M}_D)^{|\mathbf{b}|}$ and $\mathbf{c} \in U_{P_j}^{\mathcal{U}_N}$. By (A.32), $\mathcal{U}_N \models U_{Q_{r,i,j}}(\mathbf{d}_s, \mathbf{v}_i, \mathbf{v}_j)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$. Thus, $\mathcal{U}_N \models r^*$.

Let r be a rule of the form (A.5). Let $\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}$ be an assignment on $Dom(\mathcal{U}_N)$, and \mathbf{d}_s , $(N_2 < s \le N_3)$, be a tuple in D. We have $\mathbf{d}_s^{\mathcal{U}_N} \in QM_3^{\mathcal{U}_N}$, since \mathcal{U}_N is a model of rules (A.19) and (A.22). If $\mathcal{U}_N \models U_{P_i}(\mathbf{v}_i) \land U_{P_j}(\mathbf{v}_j)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$, we have $\mathbf{b} \in U_{P_i}^{\mathcal{U}_N}$ and $\mathbf{c} \in U_{P_j}^{\mathcal{U}_N}$. By (A.33), $\mathcal{U}_N \models U_{Q_{r,i,j}}(\mathbf{d}_s, \mathbf{v}_i, \mathbf{v}_j)[\mathbf{v}_i \mathbf{v}_j / \mathbf{bc}]$. Thus, $\mathcal{U}_N \models r^*$.

(d) Let r be a rule of the form (27), and r_i^N a rule of the (A.1), $1 \le i \le k$. We will show $\mathcal{U}_N \models (r_i^N)^*$ by contradiction.

Assume that there exists an assignment θ such that $\mathcal{U}_N \models Body(r_i^{N^*})\theta$ and $\mathcal{U}_N \nvDash \alpha_i^* \theta$. By $\mathcal{U}_N \models Body(r_i^{N^*})$, we have

$$\mathcal{U}_N \models \operatorname{OP}\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn\theta,$$

where $i \neq j$ and $1 \leq j \leq k$. So we have

$$\mathcal{U}_N \models \alpha_i^* \to \alpha_i^*$$

by Proposition 4. Since $\mathcal{U}_N \not\models \alpha_i^* \theta$, we have $\mathcal{U}_N \not\models \alpha_j^* \theta$, $1 \le j \le k$.

However, \mathcal{U}_D , \mathcal{U}_D and \mathcal{U}_N agree on every predicate in $\tau(\Gamma_D)$. Therefore, we also have $\mathcal{U}_D \models Body(r^*)\theta$ and $\mathcal{U}_D \not\models \alpha_j^*\theta$, $1 \leq j \leq k$. This is a contradiction, since $\mathcal{U}_D \models r^*$. \Box

Appendix A.3. The Proof of Lemma 4

In this section, we will show that if \mathcal{M}_D is a stable model of Γ_D , then there exists a stable model \mathcal{M}_N of Γ_N such that \mathcal{M}_D and \mathcal{M}_N agree on all interpretations of constants and predicates in $\tau(\Gamma_D)$.

Let \mathcal{M}_D be a structure on $\tau(\Gamma)$. By \mathcal{M}_N , we denote the structure on $\tau(\Gamma_N)$ such that:

Dom(M_N) = Dom₁(M_N)∪Dom₂(M_N), where Dom₁(M_N) = Dom(M_D) and Dom₂(M_N) = {c^{M_N} | c ∈ C_{new} \ {cc}} is a set of new domain elements;

- the constants in τ(Γ_D) are interpreted the same as those in M_D, the new constant c ∈ C_{new} \ {cc} is interpreted as the new domain element c^{M_N}, and cc is interpreted as any of the domain element in Dom₁(M_D);
- ٠

$$f_{\mathcal{M}_N}(c) = \begin{cases} f_{\mathcal{M}_D}(c), & c \in Dom(\mathcal{M}_D) \\ m, & c \text{ is a domain element of the form } c_{m,s}^{\mathcal{M}_N} \\ N, & c \text{ is a domain element of the form } (cn)^{\mathcal{M}_N} \end{cases}$$

- $P^{\mathcal{M}_N} = P^{\mathcal{M}_D}$ if P is a predicate in $\tau(\Gamma_D)$;
- $QM_i^{\mathcal{M}_N} = \{\mathbf{d}_1^{\mathcal{M}_N}, \dots, \mathbf{d}_{N_i}^{\mathcal{M}_N}\}$ if QM_i is a predicate in $\mathcal{QM}, i = 1, 2, 3;$
- $QD_i^{\mathcal{M}_N} = \{\mathbf{d}_i^{\mathcal{M}_N}\}$ if QD_i is a predicate in $\mathcal{QD}, 1 \leq i \leq N_3$;
- $\mathbf{abc} \in Q_{r,i,j}^{\mathcal{M}_N}$ if and only if at least one of the following holds:

$$\mathbf{a} \in QM_1^{\mathcal{M}_N}, \text{ and } \mathbf{bc} \in Dom_1(\mathcal{M}_N)^{|\mathbf{bc}|}$$
 (A.42)

$$\mathbf{a} \in QM_2^{\mathcal{M}_N}, \mathbf{b} \in Dom_1(\mathcal{M}_N)^{|\mathbf{b}|}, \text{ and } \mathbf{c} \in P_j^{\mathcal{M}_N}$$
 (A.43)

$$\mathbf{a} \in QM_3^{\mathcal{M}_N}, \mathbf{b} \in P_i^{\mathcal{M}_N}, \text{ and } \mathbf{c} \in P_i^{\mathcal{M}_N}.$$
 (A.44)

where

- $Q_{r,i,j} \in \mathcal{Q}, r \in \Gamma_D$ is a rule of the form (27), and $\alpha_i = P_i(\mathbf{v}_i)$ and $\alpha_j = P_j(\mathbf{v}_j)$ are two atoms in the head of r,
- a, b, c are tuples of domain elements such that $|\mathbf{a}| = K$, and b and c match the arity of P_i and P_j respectively.

Now, we present the proof of Lemma 4.

Proof: First, we show that \mathcal{M}_N is a model of Γ_N . It suffices to show that Γ_N is a model of the rules in CST, DEF and AGG.

(1) Consider the rules in CST.

- \mathcal{M}_N is a model of rule (A.10), since $P^{\mathcal{M}_N} = P^{\mathcal{M}_D}$ if P is a predicate in $\tau(\Gamma_D)$;
- \mathcal{M}_N is a model of rules (A.11) and (A.12), since the constant *cc* and constants in $\tau(\Gamma_D)$ are interpreted as the domain elements in $Dom_1(\mathcal{M}_N)$ while constants in C_{new} are interpreted as domain elements in $Dom_2(\mathcal{M}_N)$;

- *M_N* is a model of rule (A.13), since constants in *C_{new}* \ {*cc*} are always interpreted as new distinct domain elements in *Dom*₂(*M_N*);
- \mathcal{M}_N is a model of rule (A.14), since

$$QM_{1}^{\mathcal{M}_{N}} = \{\mathbf{d}_{1}^{\mathcal{M}_{N}}, \dots, \mathbf{d}_{N_{1}}^{\mathcal{M}_{N}}\},\$$

$$f_{\mathcal{M}_{N}}(\{\mathbf{d}_{1}^{\mathcal{M}_{N}}, \dots, \mathbf{d}_{N_{1}}^{\mathcal{M}_{N}}\}) = \{\mathbf{m}_{1}, \dots, \mathbf{m}_{N_{1}}\} = M_{1},\$$

and $OP(M_1) \preceq N$;

- Similarly to rule (A.14), \mathcal{M}_N is a model of rules (A.15) and (A.16);
- \mathcal{M}_N is a model of rules (A.17 A.24), since $QM_i^{\mathcal{M}_N} = \{\mathbf{d}_1^{\mathcal{M}_N}, \dots, \mathbf{d}_{N_i}^{\mathcal{M}_N}\}(i = 1, 2, 3)$ and $QD_i^{\mathcal{M}_N} = \{\mathbf{d}_i^{\mathcal{M}_N}\}, (1 \le i \le N_3).$

(2) Consider the rules in DEF.

- Let r be a rule of the form (A.3). Let v_iv_j/bc be an assignment on Dom(M_N) and d_s a tuple in D, where (1 ≤ s ≤ N₁). If M_N ⊨ NotLitsNew(v_iv_j)[v_iv_j/bc], then bc ∈ Dom₁(M_D)^{|bc|}. By (A.44), we have M_N ⊨ Q_{r,i,j}(d_s, v_i, v_j)[v_iv_j/bc]. So, M_N is a model of r. Similarly, M_N is a model of rules of the form (A.4) and (A.5).
- Let r be a rule of the form (A.6). Let $\mathbf{xv}_i \mathbf{v}_j / \mathbf{abc}$ be an assignment on $Dom(\mathcal{M}_N)$. If $\mathcal{M}_N \models Q_{r,i,j}(\mathbf{d}_s, \mathbf{v}_i, \mathbf{v}_j) \land \neg P_j(\mathbf{v}_j)[\mathbf{xv}_i \mathbf{v}_j / \mathbf{abc}]$, then (A.42) holds and (A.43) and (A.43) do not hold. So we have $\mathbf{a} \in {\mathbf{d}_1^{\mathcal{M}_N}, \ldots, \mathbf{d}_{N_1}^{\mathcal{M}_N}}$. Thus, \mathcal{M}_N is a model of rule (A.6). Similarly, \mathcal{M}_N is a model of rules of the form (A.7) and (A.8).

(3) Consider the rules in AGG. Let r be a rule of the form (27), and r_i^N a rule of the (A.1), $1 \le i \le k$. We will show $\mathcal{M}_N \models (r_i^N)$ by contradiction.

Assume that there exists an assignment θ such that $\mathcal{M}_N \models Body(r_i^N)\theta$ and $\mathcal{M}_N \not\models \alpha_i \theta$. We have

$$\mathcal{M}_N \models \operatorname{OP}\langle x : Q_{r,i,j}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \rangle \preceq cn\theta$$

where $i \neq j$ and $1 \leq j \leq k$. By Proposition 3, we have

$$\mathcal{M}_N \models (\alpha_j \to \alpha_i)\theta.$$

Note that we have already shown $\mathcal{M}_N \models \mathsf{DEF} \cup \mathsf{CST}$. Together with $\mathcal{M}_N \not\models \alpha_i \theta$, we have $\mathcal{M}_N \not\models \alpha_i \theta$, for all $1 \leq j \leq k$. In addition, \mathcal{M}_D and \mathcal{M}_N agree on every

predicates in $\tau(\Gamma_D)$. So, we have $\mathcal{M}_D \models Body(r)\theta$ and $\mathcal{M}_D \not\models \alpha_j\theta$, $1 \le j \le k$. This is a contradiction, since $\mathcal{M}_D \models r$.

It remains to show

$$\mathcal{M}_{N} \models \neg \exists \mathbf{U}_{N}(\mathbf{U}_{N} < \mathbf{P}_{N} \land \bigwedge_{r \in \Gamma_{N} \backslash \Gamma_{N}^{\perp}} \widehat{r^{*}}).$$
(A.45)

We show this by contradiction. Otherwise, there exists a structure \mathcal{U}_N on $\tau(\Gamma_N) \cup U_D$ such that \mathcal{U}_N is an extension of \mathcal{M}_N and

$$\mathcal{U}_N \models \mathbf{U}_N < \mathbf{P}_N \land \bigwedge_{r \in \Gamma_N \setminus \Gamma_N^{\perp}} \widehat{r^*}.$$
 (A.46)

Let \mathcal{U}_D be a structure on $\tau(\Gamma_D) \cup \mathbf{U}_N$ such that $Dom(\mathcal{U}_D) = Dom_1(\mathcal{M}_D, c^{\mathcal{U}_D} = c^{\mathcal{U}_N}$ for every constant $c \in \tau(\Gamma_D)$ and $P^{\mathcal{U}_D} = P^{\mathcal{U}_N}$ for every predicate $P \in \tau(\Gamma_D)$. By the construction of \mathcal{U}_N and $\mathcal{M}_N, \mathcal{U}_D$ also agrees on all interpretations of constants and predicates in $\tau(\Gamma_D)$. In the following, we will show

$$\mathcal{U}_{D} \models \mathbf{U}_{D} < \mathbf{P}_{D} \land \bigwedge_{r \in \Gamma_{D} \backslash \Gamma_{D}^{\perp}} \widehat{r^{*}}.$$
(A.47)

This is a contradiction to the fact that \mathcal{M}_D is a stable model of Γ_D .

We first show that \mathcal{U}_N is a \mathcal{Q} -reserve extension of \mathcal{M}_N , and $\mathcal{U}_N \models \mathbf{U}_D < \mathbf{P}_D$ Since $\mathcal{U}_N \models \bigwedge_{r \in \mathsf{CST}} \hat{r^*}$, we have $QM_i^{\mathcal{U}_N} = \{\mathbf{d}_1^{\mathcal{U}_N}, \dots, \mathbf{d}_{N_i}^{\mathcal{U}_N}\}(i = 1, 2, 3)$ and $QD_i^{\mathcal{U}_N} = \{\mathbf{d}_i^{\mathcal{U}_N}\}, (1 \le i \le N_3)$. So, $U_P^{\mathcal{U}_N} = U_P^{\mathcal{M}_N}$ for predicate $U_P \in \mathbf{U}_N$, where $P \in \mathcal{QD} \cup \mathcal{QM}$. Furthermore, we can see that it is impossible that for all predicates $P \in \tau(\Gamma_D), P^{\mathcal{U}_N} = U_P^{\mathcal{U}_N}$. Otherwise, we have $Q_{r,i,j}^{\mathcal{U}_N} = U_{Q_{r,i,j}}^{\mathcal{U}_N}$, since $\mathcal{U}_N \models \bigwedge_{r \in \mathsf{DEF}} \hat{r^*}$. This is a contradiction to the fact that $\mathcal{U}_N \models \mathbf{U}_N < \mathbf{P}_N$.

By the construction of $\mathcal{U}_N, \mathcal{U}_D$ and \mathcal{U}_N agree on the interpretations of all predicates in $\tau(\Gamma_D)$, so we also have $\mathcal{U}_D \models \mathbf{U}_D < \mathbf{P}_D$. It remains to show $\mathcal{U}_D \models \hat{r^*}$, where r is a rule of the form (27). If there is a rule $r \in \Gamma_D$ and $\mathcal{U}_D \not\models r$. Then there exists a substitution θ such that $\mathcal{U}_D \models Body(r^*)\theta$ and $\mathcal{U}_D \not\models \alpha_i^*\theta$ for all $1 \leq i \leq k$. Note that \mathcal{U}_D and \mathcal{U}_N agree on the interpretation of all predicates in $\tau(\Gamma_D)$, we have $\mathcal{U}_N \not\models \alpha_i^*\theta$ for all $1 \leq i \leq k$, thus $\mathcal{U}_N \models (\alpha_j^* \to \alpha_i^*)\theta$ for all $1 \leq i \neq j \leq k$. By Proposition 4, $\mathcal{U}_N \models (\operatorname{OP}\langle x : U_{Q_{r,i,j}}(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_j) \leq cn)\theta$ for all $1 \leq i \neq j \leq k$. This is a contradiction to the fact that $\mathcal{U}_N \models \widehat{r_i^{N^*}}\theta$ and $\mathcal{U}_N \not\models \alpha_i^*\theta$ for all $1 \leq i \leq k$.

This completes the proof. \Box