Gelfond-Zhang aggregates as propositional formulas*

Pedro Cabalar[1], Jorge Fandinno[1], Torsten Schaub[2], and Sebastian Schellhorn[2]

[1]University of Corunna, Spain
{cabalar, jorge.fandino}@udc.es

[2]University of Potsdam, Germany
{torsten, seschell}@cs.uni-potsdam.de

Abstract. We show that any ASP aggregate interpreted under Gelfond and Zhang's (GZ) semantics can be replaced (under strong equivalence) by a propositional formula. Restricted to the original GZ syntax, the resulting formula is reducible to a disjunction of conjunctions of literals but the formulation is still applicable even when the syntax is extended to allow for arbitrary formulas (including nested aggregates) in the condition. Once GZ-aggregates are represented as formulas, we establish a formal comparison (in terms of the logic of Here-and-There) to Ferraris' (F) aggregates, which are defined by a different formula translation involving nested implications. In particular, we prove that if we replace an F-aggregate by a GZ-aggregate in a rule head, we do not lose answer sets (although more can be gained). This extends the previously known result that the opposite happens in rule bodies, i.e., replacing a GZ-aggregate by an F-aggregate in the body may yield more answer sets. Finally, we characterise a class of aggregates for which GZ- and F-semantics coincide.

1 Introduction

Answer Set Programming (ASP [3]) has become an established problem-solving paradigm and a prime candidate for practical Knowledge Representation and Reasoning (KRR). The reasons for this success are manifold. The most obvious one is the availability of effective solvers [12, 8] and a growing list of covered application domains [6]. A probably equally important reason is its declarative semantics, having been generalized from the original *stable models* [13] of normal logic programs up to arbitrary first-order [18, 11] and infinitary [15] formulas. Several logical characterizations of ASP have been obtained, among which *Equilibrium Logic* [17] and its monotonic basis, the intermediate logic of *Here-and-There* (HT), are arguably the most prominent ones. These generalisations have allowed us to understand ASP as a general logical framework for Non-Monotonic Reasoning. Finally, a third relevant cause of ASP's success lies in its flexible specification language [5], offering constructs especially useful for practical KRR. Some of its distinctive constructs are *aggregates*, allowing for operations on sets of elements such as counting the number of instances for which a formula holds, or adding all the integer values for some predicate argument. Unfortunately, there is no clear agreement on the expected behavior of aggregates in ASP, and several alternative semantics

* Partially supported by grants GPC-ED431B 2016/035 and 2016-2019 ED431G/01 (Xunta de Galicia, Spain), TIN 2013-42149-P (MINECO, Spain), and SCHA 550/9 (DFG, Germany).

have been defined [20, 19, 21, 10, 7, 14], among which perhaps Ferraris' [10] and Faber et al's [7] are the two more consolidated ones due to their respective implementations in the ASP solvers `clingo` [12] and `DLV` [8]. Although these two approaches may differ when the aggregates are in the scope of default negation, they coincide for the rest of cases (like all the examples in this paper), even when aggregates are involved in recursive definitions. Ferraris' (F-)aggregates additionally show a remarkable feature: they can be expressed as propositional formulas in the logic of HT, something that greatly simplifies their formal treatment. To illustrate this, let us explore the simple rule:

$$p(a) \leftarrow \texttt{count}\{X : p(X)\} \geq n. \tag{1}$$

where $p(a)$ recursively depends on the number of atoms of the form $p(X)$. Suppose first that $n = 1$. Since the domain only contains a, $\texttt{count}\{X : p(X)\} \geq 1$ is true iff $p(a)$ holds. This is captured in Ferraris' translation of (1) for $n = 1$ that amounts to $p(a) \leftarrow p(a)$, a tautology whose only stable model is $\emptyset$. Suppose now that $n = 0$. Then, the aggregate is considered as tautological and the HT-translation of (1) corresponds to $p(a) \leftarrow \top$ whose unique stable model is $\{p(a)\}$. Finally, as one more elaboration, assume $n = 1$ and suppose we add the fact $p(b)$. Then, (1) becomes the formula $p(a) \leftarrow p(a) \vee p(b)$ that, together with fact $p(b)$, is HT-equivalent to:

$$p(b). \qquad p(a) \leftarrow p(a). \qquad p(a) \leftarrow p(b). \tag{2}$$

This results in the unique stable model $\{p(a), p(b)\}$.

Recently, Gelfond and Zhang [14] (GZ) proposed a more restrictive interpretation of recursive aggregates that imposes the so-called *Vicious Circle Principle*, namely, "*no object or property can be introduced by the definition referring to the totality of objects satisfying this property*." According to this principle, if we have a program whose only definition for $p(a)$ is (1), we may leave $p(a)$ false, but we cannot be forced to derive its truth, since it depends on a set of atoms $\{X : p(X)\}$ that includes $p(a)$ itself. In this way, if $n = 1$, the GZ-stable model for (1) is also $\emptyset$, as there is no need to assume $p(a)$. However, if we have $n = 0$, we cannot leave $p(a)$ false any more (the rule would have a true body and a false head) and, at the same time, $p(a)$ cannot become true because it depends on a vicious circle. Something similar happens for $n = 1$ when adding fact $p(b)$. As shown in [2], GZ-programs are stronger than F-programs in the sense that, when they represent the same problem, any GZ-stable model is also an F-stable model, but the opposite may not hold (as we saw in the examples above). Without entering a discussion of which semantics is more intuitive or suitable for practical purposes, one objective disadvantage of GZ-aggregates is that they lacked a logical representation so far; they were exclusively defined in terms of a reduct, something that made their formal analysis more limited and the comparison to F-aggregates more cumbersome.

In this paper, we show that, in fact, it is also possible to understand a GZ-aggregate as a propositional formula, classically equivalent to the F-aggregate translation, but with a *different meaning* in HT. For instance, the GZ-translation for (1) with $n = 1$ coincides with the F-encoding $p(a) \leftarrow p(a)$, but if we change to $n = 0$ we get the formula $p(a) \leftarrow p(a) \vee \neg p(a)$ whose antecedent is valid in classical logic, but not in HT. In fact, the whole formula is HT-equivalent to the program:

$$p(a) \leftarrow p(a). \qquad p(a) \leftarrow \neg p(a).$$

This makes it now obvious that there is no stable model. Similarly, when we add fact $p(b)$ and $n = 1$, the GZ-translation eventually leads to the propositional program:

$$p(b). \quad p(a) \leftarrow p(a) \wedge \neg p(b). \quad p(a) \leftarrow \neg p(a) \wedge p(b). \quad p(a) \leftarrow p(a) \wedge p(b). \quad (3)$$

Again, it is classically equivalent to the F-translation (2), but quite different in logic programming, where the third rule and fact $p(b)$ enforce the non-existence of stable models.

The rest of the paper is organized as follows. In the next section, we review some basic definitions that will be needed through the paper. In Section 3, we present a generalisation of Ferraris' reduct that covers GZ-aggregates and show that the latter can be replaced, under strong equivalence, by a propositional formula. In Section 4, we show that, in general, GZ-aggregates are stronger than F-aggregates in HT and, as a result, characterise the effect of replacing some occurrence of a GZ-aggregate by a corresponding F-aggregate. We also identify a family of aggregates in which both semantics coincide. Finally, Section 5 concludes the paper.

2 Background

We begin by introducing some basic definitions used in the rest of the paper. Let $\mathcal{L}$ be some syntactic language and assume we have a definition of *stable model* for any theory $\Gamma \subseteq \mathcal{L}$ in that syntax. Moreover, let $SM(\Gamma)$ denote the stable models of Γ. Two theories Γ, Γ' are *strongly equivalent*, written $\Gamma \equiv_s \Gamma'$, iff $SM(\Gamma \cup \Delta) = SM(\Gamma' \cup \Delta)$ for any theory Δ. We will provide a stronger definition of $\equiv_s$ for expressions in $\mathcal{L}$. Let $\Gamma(\varphi)$ denote some theory with a *distinguished occurrence* of a subformula φ and let $\Gamma(\psi)$ be the result of replacing that occurrence φ by ψ in $\Gamma(\varphi)$. Two expressions $\varphi, \psi \in \mathcal{L}$ are said to be *strongly equivalent*, also written $\varphi \equiv_s \psi$, when $\Gamma(\varphi) \equiv_s \Gamma(\psi)$ for an arbitrary[1] $\Gamma(\varphi) \subseteq \mathcal{L}$. We also recall next some basic definitions and results related to the logic of *Here-and-There* (HT). Let At be a set of ground atoms called the *propositional signature*. A *propositional formula* φ is defined using the grammar:

$$\varphi ::= \bot \mid a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \qquad \text{for any } a \in At.$$

We use Greek letters φ and ψ and their variants to stand for propositional formulas. We define the derived operators $\neg\varphi \stackrel{\text{def}}{=} (\varphi \rightarrow \bot)$, $\varphi \leftrightarrow \psi \stackrel{\text{def}}{=} (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$ and $\top \stackrel{\text{def}}{=} \neg\bot$. A propositional formula in which every occurrence of an implication is of the form $\neg\varphi = \varphi \rightarrow \bot$ is called a *nested expression*.

A *classical interpretation* T is a set of atoms $T \subseteq At$. We write $\models_{cl}$ to stand both for classical satisfaction and entailment, and $\equiv_{cl}$ represents classical equivalence. An HT-*interpretation* is a pair $\langle H, T\rangle$ of sets of atoms $H \subseteq T \subseteq At$. An interpretation $\langle H, T\rangle$ *satisfies* a formula φ, written $\langle H, T\rangle \models \varphi$, if any of the following recursive conditions holds:

[1] Note that, for arbitrary languages and semantics, this definition is stronger than usual, as it refers to *any subformula replacement* and not just conjunctions of formulas, as usual. For instance, $\varphi \equiv_s \psi$ also implies $\{\varphi \otimes \alpha\} \equiv_s \{\psi \otimes \alpha\}$ for any binary operator $\otimes$ in our language. When $\mathcal{L}$ is a logical language and $\equiv_s$ amounts to equivalence in HT (or any logic with substitution of equivalents) the distinction becomes irrelevant.

- $\langle H, T\rangle \not\models \bot$
- $\langle H, T\rangle \models p$ iff $p \in H$
- $\langle H, T\rangle \models \varphi \wedge \psi$ iff $\langle H, T\rangle \models \varphi$ and $\langle H, T\rangle \models \psi$
- $\langle H, T\rangle \models \varphi \vee \psi$ iff $\langle H, T\rangle \models \varphi$ or $\langle H, T\rangle \models \psi$
- $\langle H, T\rangle \models \varphi \rightarrow \psi$ iff both (i) $T \models_{cl} \varphi \rightarrow \psi$ and (ii) $\langle H, T\rangle \not\models \varphi$ or $\langle H, T\rangle \models \psi$

It is not difficult to see that $\langle T, T\rangle \models \phi$ iff $T \models_{cl} \phi$. A *(propositional) theory* is a set of propositional formulas. An interpretation $\langle H, T\rangle$ is a *model* of a theory Γ when $\langle H, T\rangle \models \varphi$ for all $\varphi \in \Gamma$. A theory Γ *entails* a formula φ, written $\Gamma \models \varphi$, when all models of Γ satisfy φ. Two theories Γ, Γ' are (HT)-*equivalent*, written $\Gamma \equiv \Gamma'$, if they share the same set of models.

Definition 1 (equilibrium/stable model). *A total interpretation $\langle T, T\rangle$ is an* equilibrium model *of a formula φ iff $\langle T, T\rangle \models \varphi$ and there is no $H \subset T$ such that $\langle H, T\rangle \models \varphi$. If so, we say that T is a* stable model *of φ.* □

Proposition 1. *The following are general properties of HT:*

i) if $\langle H, T\rangle \models \varphi$ then $\langle T, T\rangle \models \varphi$ (i.e., $T \models_{cl} \varphi$)

ii) $\langle H, T\rangle \models \neg\varphi$ iff $T \models_{cl} \neg\varphi$ □

Definition 2 (Ferraris' reduct). *The reduct of a formula φ with respect to an interpretation T, written φ^T, is defined as:*

$$\varphi^T \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } T \not\models_{cl} \varphi \\ a & \text{if } \varphi \text{ is some atom } a \in T \\ \varphi_1^T \otimes \varphi_2^T & \text{if } T \models_{cl} \varphi \text{ and } \varphi = (\varphi_1 \otimes \varphi_2) \text{ for some } \otimes \in \{\wedge, \vee, \rightarrow\} \end{cases}$$

That is, φ^T is the result of replacing by $\bot$ each maximal subformula ψ of φ s.t. $T \not\models_{cl} \psi$.

Proposition 2 (Lemma 1 in [9]). *For any pair of interpretations $H \subseteq T$ and any φ: $H \models_{cl} \varphi^T$ iff $\langle H, T\rangle \models \varphi$.* □

As is well-known, strong equivalence for propositional formulas corresponds to HT-equivalence [16], that is, $\varphi \equiv_s \psi$ iff $\varphi \equiv \psi$ in that language. The following result follows from Corollary 3 in [1].

Proposition 3. *If $\varphi \models \psi$ and $\varphi \equiv_{cl} \psi$, then SM$(\varphi) \supseteq$ SM(ψ).* □

In other words, if φ is stronger than ψ in HT (and so, in classical logic too), but they further happen to be classically equivalent, then φ is weaker with respect to stable models. As an example, note that $(p \vee q) \models (\neg p \rightarrow q)$. As they are classically equivalent, SM$(p \vee q) \supseteq$ SM$(\neg p \rightarrow q)$ which is not such a strong result. However, since HT-entailment is monotonic with respect to conjunction, it follows that $(p \vee q) \wedge \gamma \models (\neg p \rightarrow q) \wedge \gamma$ also holds for any γ, and thus, if we replace a disjunctive rule $(p \vee q)$ by $(\neg p \rightarrow q)$ in any program we may lose some stable models, but the remaining are still applicable to $(p \vee q)$. We can generalize this behavior not only on conjunctions, but also to cover the replacement of any subformula φ. We say that an occurrence φ of a formula is *positive* in a theory $\Gamma(\varphi)$ if the number of implications in $\Gamma(\varphi)$ containing occurrence φ in the antecedent is even. It is called *negative* otherwise.

Proposition 4. *Let φ and ψ be two formulas satisfying $\varphi \models \psi$ and $\varphi \equiv_{cl} \psi$. Then:*
i) SM($\Gamma(\varphi)$) $\supseteq$ SM($\Gamma(\psi)$) for any theory $\Gamma(\varphi)$ where occurrence φ is positive;
ii) SM($\Gamma(\varphi)$) $\subseteq$ SM($\Gamma(\psi)$) for any theory $\Gamma(\varphi)$ where occurrence φ is negative. □

Back to the example, note that $(p \vee q)$ occurs positively in $(p \vee q) \wedge \gamma$ and so, $(\neg p \rightarrow q) \wedge \gamma$ has a subset of stable models. On the other hand, it occurs negatively in $(p \vee q) \rightarrow \gamma$, and so, $(\neg p \rightarrow q) \rightarrow \gamma$ has a superset of stable models.

3 Aggregates as formulas

To deal with aggregates, we consider a simplified first order signature $\Sigma = \langle \mathcal{C}, \mathcal{A}, \mathcal{P} \rangle$ formed by three pairwise disjoint sets respectively called *constants*, *aggregate symbols* and *predicate symbols*.Since we focus on translations to propositional expressions, we do not consider functions or arithmetic operations[2] (other than aggregates). As a result, a *term* can only be either a constant $c \in \mathcal{C}$ or a variable X. We use boldface symbols to represent tuples of terms, such as $\mathbf{t}$, and write $|\mathbf{t}|$ to stand for the tuple's arity. As usual, a *predicate atom* (or *atom* for short) is an expression of the form $p(\mathbf{t})$ where $\mathbf{t}$ is a tuple of terms; moreover, $p(\mathbf{t})$ is said to be *ground* iff all its terms are constants $\mathbf{t} \subseteq \mathcal{C}^{|\mathbf{t}|}$. We write $At(\mathcal{C}, \mathcal{P})$ to stand for the set of ground atoms for predicates $\mathcal{P}$ and constants $\mathcal{C}$. A *literal* is either an atom a (positive literal) or its default negation *not* a (negative literal). An *(aggregate) formula* φ is recursively defined by the following grammar:

$$\varphi ::= \bot \mid p(\mathbf{t}) \mid f\{\mathbf{X} : \varphi\} \prec n \mid f\{\mathbf{c} : \varphi, \ldots, \mathbf{c} : \varphi\} \prec n \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi$$

where $p(\mathbf{t})$ is an atom, $f \in \mathcal{A}$ is an aggregate symbol, $\mathbf{X}$ is a non-empty tuple of variables, $\mathbf{c}$ is a non-empty tuple of constants, $\prec \in \{=, \neq, \leq, \geq, <, >\}$ is an *arithmetic relation*, and $n \in \mathbb{Z}$ is an integer constant. A *(aggregate) theory* is a set of aggregate formulas. As we can see, we have two types of *aggregates*: $f\{\mathbf{X} : \varphi\} \prec n$ called *GZ-aggregates* (or *set atoms*); and $f\{\mathbf{c}_1 : \varphi_1, \ldots, \mathbf{c}_m : \varphi_m\} \prec n$, with $\mathbf{c}_i$ of same arity, called *F-aggregates*. This syntactic distinction respects the original syntax[3] and also turns out to be convenient for comparison purposes, since we can assign a different semantics to each type of aggregate without ambiguity. An important observation is that, in our general language, it is possible to nest GZ and F-aggregates in a completely arbitrary way, since φ and $\varphi_1, \ldots, \varphi_m$ inside brackets are aggregate formulas in their turn. Achieving this generalisation is not surprising, once aggregates can be seen as propositional formulas. A *GZ-formula* (resp. *F-formula*) is one in which all its aggregates are GZ-aggregates (resp. F-aggregates). We sometimes informally talk about *rules* (resp. *programs*) instead of formulas (resp. theories) when the syntax coincides with the usual in logic programming.

The technical treatment of F-aggregates is directly extracted from [10], so the focus in this section is put on GZ-aggregates, where our contribution lies. One of their distinctive features is the use of variables $\mathbf{X}$. In fact, a formula φ inside $A = f\{\mathbf{X} : \varphi\} \prec n$ (called the *condition* of A) normally contains occurrences of $\mathbf{X}$, so we usually write it as $cond(\mathbf{X})$. Moreover, the occurrences of variables $\mathbf{X}$ in A are said to be *bound* to A. A

[2] An extension to a full first-order language is under development.
[3] Ferraris actually uses $\varphi_i = w$ rather than $\mathbf{c} : \varphi$, but this is not a substantial difference, assuming w is the first element in tuple $\mathbf{c}$.

variable occurrence X in a formula φ is *free* if it is not bound to any GZ-aggregate in φ. A formula or theory is *closed* iff it contains no free variables. We define the grounding of a formula $\varphi(\mathbf{X})$ with free variables $\mathbf{X}$ as expected: $\mathtt{Gr}(\ \varphi(\mathbf{X})\ ) \stackrel{\text{def}}{=} \{\varphi(\mathbf{c}) \mid \mathbf{c} \in \mathcal{C}^{|\mathbf{X}|}\}$. In the rest, we assume that all theories are closed. This is not a limitation, since a non-closed formula $\varphi(\mathbf{X})$ in some theory can be understood as an abbreviation of its grounding $\mathtt{Gr}(\varphi(\mathbf{X}))$, as usual. Given a set of formulas S, we write $\bigwedge S$ and $\bigvee S$ to stand for their conjunction and disjunction, respectively; we let $\bigvee \emptyset = \bot$ and $\bigwedge \emptyset = \top$.

To define the semantics, we assume that for all aggregate symbols $f \in \mathcal{A}$ and arities $m \geq 1$, there exists a predefined associated partial function $\hat{f}_m : 2^{\mathcal{C}^m} \to \mathbb{Z}$ that, for each set S of m-tuples of constants, either returns a number $\hat{f}_m(S)$ or is undefined. This predefined value is the expected one for the usual aggregate functions $\mathtt{sum}, \mathtt{count}, \mathtt{max}$, etc. For example, for aggregate symbol $\mathtt{sum}$ and arity $m = 1$ the function returns the aggregate addition when the set consists of (1-tuples of) integer numbers. For instance, $\widehat{\mathtt{sum}}_1(\{7, 2, -4\}) = 5$ and $\widehat{\mathtt{sum}}_1(\emptyset) = 0$ but $\widehat{\mathtt{sum}}_1(\{7, a, 3, b\})$ is undefined. For integer aggregate functions of arity $m > 1$, we assume that the aggregate is applied on the leftmost elements in the tuples when all of them are integer, so that, for instance, $\widehat{\mathtt{sum}}_2(\{\langle 7, a\rangle, \langle 2, b\rangle, \langle 2, a\rangle\}) = 11$. We omit the arity when clear from the context.

A *classical interpretation* T is a set of ground atoms $T \subseteq At(\mathcal{C}, \mathcal{P})$.

Definition 3. *A classical interpretation* T satisfies *a formula* φ*, denoted by* $T \models_{cl} \varphi$*, iff*

i) $T \not\models_{cl} \bot$
ii) $T \models_{cl} p(\mathbf{c})$ *iff* $p(\mathbf{c}) \in T$ *for any ground atom* $p(\mathbf{c})$
iii) $T \models_{cl} f\{\mathbf{X} : cond(\mathbf{X})\} \prec n$ *iff* $\hat{f}_{|\mathbf{X}|}\big(\ \{\ \mathbf{c} \in \mathcal{C}^{|\mathbf{X}|} \ \big|\ T \models_{cl} cond(\mathbf{c})\ \}\ \big)$ *has some value* $k \in \mathbb{Z}$ *and* $k \prec n$ *holds for the usual meaning of arithmetic relation* $\prec$
iv) $T \models_{cl} f\{\mathbf{c}_1 : \varphi_1, \ldots, \mathbf{c}_m : \varphi_m\} \prec n$ *iff* $\hat{f}_{|\mathbf{c}_1|}\big(\ \{\ \mathbf{c}_i \ \big|\ T \models_{cl} \varphi_i\ \}\ \big)$ *has some value* $k \in \mathbb{Z}$ *and, again,* $k \prec n$ *holds for its usual meaning*
v) $T \models_{cl} \varphi \wedge \psi$ *iff* $T \models_{cl} \varphi$ *and* $T \models_{cl} \psi$
vi) $T \models_{cl} \varphi \vee \psi$ *iff* $T \models_{cl} \varphi$ *or* $T \models_{cl} \psi$
vii) $T \models_{cl} \varphi \to \psi$ *iff* $T \not\models_{cl} \varphi$ *or* $T \models_{cl} \psi$.

We say that T *is a* model *of a theory* Γ *iff* $T \models_{cl} \varphi$ *for all* $\varphi \in \Gamma$. □

Given interpretation T, we divide any theory Γ into the two disjoint subsets:

$$\Gamma_T^+ \stackrel{\text{def}}{=} \{\varphi \in \Gamma \mid T \models_{cl} \varphi\} \qquad \Gamma_T^- \stackrel{\text{def}}{=} \Gamma \setminus \Gamma_T^+$$

that is, the formulas in Γ satisfied by T and not satisfied by T, respectively. When set Γ is parametrized, say $\Gamma(z)$, we write $\Gamma_T^+(z)$ and $\Gamma_T^-(z)$ instead of $\Gamma(z)_T^+$ and $\Gamma(z)_T^-$. For instance, $\mathtt{Gr}_T^+(\varphi)$ collects the formulas from $\mathtt{Gr}(\varphi)$ satisfied by T.

Definition 4 (reduct). *The* reduct *of a GZ-aggregate* $A = f\{\mathbf{X} : cond(\mathbf{X})\} \prec n$ *with respect to a classical interpretation* T *is the formula:*

$$A^T \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } T \not\models_{cl} A \\ \big(\ \bigwedge \mathtt{Gr}_T^+(cond(\mathbf{X}))\ \big)^T & \text{otherwise} \end{cases}$$

The reduct of an F-aggregate $B = f\{\mathbf{c}_1 : \varphi_1, \ldots, \mathbf{c}_m : \varphi_m\} \prec n$ *is the formula:*

$$B^T \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } T \not\models_{cl} B \\ f\{\mathbf{c}_1 : \varphi_1^T, \ldots, \mathbf{c}_m : \varphi_m^T\} \prec n & \text{otherwise} \end{cases}$$

The reduct of any other formula is just as in Definition 2. The reduct of a theory is the set of reducts of its formulas. □

Definition 5 (stable model). *A classical interpretation T is a* stable model *of a theory Γ iff T is a $\subseteq$-minimal model of Γ^T.* □

Note that, when restricted to F-formulas, Definition 3 exactly matches the reduct definition for aggregate theories by Ferraris [10].On the other hand, when restricted to GZ-formulas, it generalises the reduct definition by Gelfond-Zhang [14] allowing arbitrary formulas in $cond(\mathbf{X})$, including nested aggregates. For this reason, in our setting, the reduct is inductively applied to $(\bigwedge \mathtt{Gr}^+_T(cond(\mathbf{X})))^T$. In the original case [14], $cond(\mathbf{X})$ was a conjunction of atoms, but it is straightforward to see that, then, $(\bigwedge \mathtt{Gr}^+_T(cond(\mathbf{X})))^T = \bigwedge \mathtt{Gr}^+_T(cond(\mathbf{X}))$. To sum up, the above definitions of stable model and reduct correspond to the original ones for Ferraris [10] and Gelfond-Zhang [14] when restricted to their respective syntactic fragments.

Proposition 5. *Stable models are classical models.* □

Example 1. Let P_1 be the program consisting of fact $p(b)$ and rule (1) with $n = 1$, and let A_1 be the GZ-aggregate in that rule. We show that P_1 has no stable models. Given ground atoms $p(a)$ and $p(b)$, the only model of the program is $T = \{p(a), p(b)\}$, since $p(b)$ is fixed as a fact, and so, A_1 must be true, so (1) entails $p(a)$ too. Since $T \models_{cl} A_1$, the reduct becomes $A_1^T = p(a) \wedge p(b)$, and so, P_1^T contains the rules $p(b)$ and $p(a) \leftarrow p(a) \wedge p(b)$, being their least model $\{p(b)\}$, so T is not stable. As another example of the aggregate reduct, if we took $T = \emptyset$, then $T \not\models_{cl} A_1$ and $A_1^T = \bot$. □

Example 2. As an example of nested GZ-aggregates, consider:

$$A_2 = \mathtt{count}\{\ X : \mathtt{sum}\{Y : owns(X, Y)\} \geq 10\ \} \geq 2$$

and imagine that $owns(X, Y)$ means that X owns some item Y whose cost is also Y. Accordingly, A_2 checks whether there are 2 or more persons X that own items for a total cost of at least 10. Suppose we have the interpretation:

$$T = \{owns(a, 6), owns(a, 8), owns(b, 2), owns(b, 3), owns(c, 12)\}$$

Then A_2 holds in T since both a and c have total values greater than 10: 14 for a and 12 for c. Therefore, A_2^T corresponds to $(\bigwedge \mathtt{Gr}^+_T(\mathtt{sum}\{Y : owns(X, Y)\} \geq 10))^T$. After grounding free variable X, we obtain:

$$(\mathtt{sum}\{Y : owns(a, Y)\} \geq 10 \wedge \mathtt{sum}\{Y : owns(c, Y)\} \geq 10)^T$$

Note that b does not occur, since its total sum is lower than 10 in T. If we apply again the reduct to the conjuncts above, we eventually obtain the conjunction: $owns(a, 6) \wedge owns(a, 8) \wedge owns(c, 12)$. □

Proposition 6. *Given formulas φ and ψ, the following conditions hold:*

i) $H \models_{cl} \varphi^T$ implies $T \models_{cl} \varphi$, and
ii) $T \models_{cl} \varphi^T$ iff $T \models_{cl} \varphi$

for any pair of interpretations $H \subseteq T$. Furthermore, the following condition also holds

iii) if $H \models_{cl} \varphi^T$ iff $H \models_{cl} \psi^T$ for all interpretations $H \subseteq T$, then φ and ψ are strongly equivalent, $\varphi \equiv_s \psi$. □

Proposition 6 generalises results from [10] to our extended language combining GZ and F-aggregates. In particular, item iii) provides a sufficient condition for strong equivalence that, in the case of propositional formulas, amounts to HT-equivalence.

We now move to consider propositional translations of aggregates. As said in the introduction, any F-aggregate can be understood as a propositional formula. Take any F-aggregate $B = f\{\mathbf{c}_1 : \varphi_1, \ldots, \mathbf{c}_m : \varphi_m\} \prec n$ where all formulas in $\{\varphi_1, \ldots, \varphi_m\}$ are propositional – moreover, let us call $conds$ (the *conditions*) to this set of formulas. By $\Phi[B]$, we denote the propositional formula:

$$\Phi[B] \quad \stackrel{\text{def}}{=} \quad \bigwedge_{T : T \not\models_{cl} B} \left(\left(\bigwedge conds_T^+ \right) \rightarrow \left(\bigvee conds_T^- \right) \right) \tag{4}$$

The following result directly follows from Proposition 12 in [10].

Proposition 7. *For any F-aggregate B with propositional conditions, $B \equiv_s \Phi[B]$.* □

Given an F-formula φ, we can define its (strongly equivalent) propositional translation $\Phi[\varphi]$ as the result of the exhaustive replacement of non-nested aggregates B by $\Phi[B]$ until all aggregates are eventually removed.

Our main contribution is to provide an analogous propositional encoding for GZ-aggregates. Given a GZ-aggregate $A = f\{\mathbf{X} : cond(\mathbf{X})\} \prec n$ with a propositional condition $cond(\mathbf{X})$, we define the propositional formula $\Phi[A]$ as

$$\Phi[A] \quad \stackrel{\text{def}}{=} \quad \bigvee_{T : T \models_{cl} A} \left(\bigwedge \mathtt{Gr}_T^+(cond(\mathbf{X})) \wedge \neg \bigvee \mathtt{Gr}_T^-(cond(\mathbf{X})) \right) \tag{5}$$

Proposition 8. *For any GZ-aggregate A with a propositional condition, $A \equiv_s \Phi[A]$.* □

This result allows us to define, for any *arbitrary* aggregate formula φ, its (strongly equivalent) propositional translation $\Phi[\varphi]$, again by exhaustive replacement of non-nested aggregates (now of any kind) C by their propositional formulas $\Phi[C]$. For any theory Γ, its (strongly equivalent) propositional translation is defined as $\Phi[\Gamma] \stackrel{\text{def}}{=} \{\, \Phi[\varphi] \mid \varphi \in \Gamma \,\}$.

Example 3. Take again the aggregate A_1 in the body of rule (1) with $n = 1$ and assume we have constants a, b. The classical models of A_1 are $\{p(a)\}$, $\{p(b)\}$ and $\{p(a),\ p(b)\}$, since some atom $p(X)$ must hold. As a result:

$$\Phi[A_1] \quad = \quad p(a) \wedge \neg p(b) \quad \vee \quad p(b) \wedge \neg p(a) \quad \vee \quad p(a) \wedge p(b)$$

and $\Phi[(1)]$ amounts to the last three rules in (3). □

Example 4. Take GZ-aggregate $A_3 = \mathtt{count}\{X : p(X)\} = 1$ and assume we have constants $\mathcal{C} = \{a, b, c\}$. The classical models of A_3 are $\{p(a)\}$, $\{p(b)\}$ and $\{p(c)\}$. Accordingly:

$$\begin{array}{rcccc} & & p(a) \wedge \neg(p(b) \vee p(c)) & & p(a) \wedge \neg p(b) \wedge \neg p(c) \\ \Phi[A_3] & = & \vee\ p(b) \wedge \neg(p(a) \vee p(c)) & \equiv & \vee\ p(b) \wedge \neg p(a) \wedge \neg p(c) \\ & & \vee\ p(c) \wedge \neg(p(a) \vee p(b)) & & \vee\ p(c) \wedge \neg p(a) \wedge \neg p(b) \end{array}$$

Theorem 1 (Main Result). *Any aggregate theory Γ is strongly equivalent to its propositional translation $\Phi[\Gamma]$, that is, $\Gamma \equiv_s \Phi[\Gamma]$.* □

4 Relation to Ferraris Aggregates

In this section, we study the relation between GZ and F-aggregates. One first observation is that GZ-aggregates are first-order structures with quantified variables, while F-aggregates allow sets of propositional expressions. Encoding a GZ-aggregate as an F-aggregate is easy: we can just ground the variables. The other direction, however, is not always possible, since the set of conditions in the F-aggregate may not have a regular representation in terms of variable substitutions. Given a GZ-aggregate $A = f\{\mathbf{X} : cond(\mathbf{X})\} \prec n$ we define its corresponding F-aggregate $\mathtt{F}[A]$:

$$\mathtt{F}[A] \stackrel{\text{def}}{=} f\{\, \mathbf{c} : cond(\mathbf{c}) \;\big|\; cond(\mathbf{c}) \in \mathtt{Gr}(cond(\mathbf{X}))\,\} \prec n \tag{6}$$

This correspondence is analogous to the process of *instantiation* used in [7] to ground aggregates with variables. It is easy to check that A and $\mathtt{F}[A]$ are classically equivalent. This can be checked using satisfaction from Definition 3 or classical logic for their propositional representations $\Phi[A] \equiv_{cl} \Phi[\mathtt{F}[A]]$. Moreover, it can be observed that these two logical representations are somehow *dual*. Indeed, $\Phi[\mathtt{F}[A]]$ eventually amounts to:

$$\Phi[\mathtt{F}[A]] \;\equiv\; \bigwedge_{T:T\not\models_{cl} A} \left(\left(\bigwedge \mathtt{Gr}^{+}_{T}(cond(\mathbf{X})) \right) \rightarrow \left(\bigvee \mathtt{Gr}^{-}_{T}(cond(\mathbf{X})) \right) \right) \tag{7}$$

which is a conjunction of formulas like $\alpha \rightarrow \beta$ for *countermodels* of A, whereas $\Phi[A]$, formula (5), is a disjunction of formulas like $\alpha \wedge \neg\beta$ for *models* of A. Another interesting consequence of the classical equivalence of A and $\mathtt{F}[A]$ is that, due to Proposition 1- ii), we can safely replace one by another when negated. In other words:

Proposition 9. *GZ-aggregate A and its corresponding F-aggregate $\mathtt{F}[A]$ are strongly equivalent when occurring in the scope of negation.* □

However, as we saw in the introduction examples, replacing a GZ-aggregate A by its F-aggregate version $\mathtt{F}[A]$ may change the program semantics. Still, the stable models obtained after such replacement are not arbitrary. As we said, [2] proved that if the GZ-aggregate A occurs in a positive rule body, then the replacement by the F-aggregate $\mathtt{F}[A]$ preserves the stable models, but may yield more. Next, we generalise this result to aggregate theories without nested GZ-aggregates. To this aim, we make use of the following proposition asserting that, indeed, $\Phi[A]$ is stronger than $\Phi[\mathtt{F}[A]]$ in HT.

Proposition 10. *For any GZ-aggregate A, $\Phi[A] \models \Phi[\mathtt{F}[A]]$ in HT.* □

Theorem 2. *For any occurrence A of a GZ-aggregate without nested aggregates:*

i) $SM(\Gamma(A)) \supseteq SM(\Gamma(\mathtt{F}[A]))$ for any theory $\Gamma(A)$ where occurrence A is positive;
ii) $SM(\Gamma(A)) \subseteq SM(\Gamma(\mathtt{F}[A]))$ for any theory $\Gamma(A)$ where occurrence A is negative. □

Proof. From $\Phi[A] \equiv_{cl} \Phi[\mathtt{F}[A]]$ and Proposition 10 we directly apply Proposition 4. □

In particular, this means that if we replace a (non-nested) GZ-aggregate A by its F-version $\mathtt{F}[A]$ in the positive head of some rule, we still get stable models of the original program, but perhaps not all of them. Theorem 2 is not directly applicable to theories with nested aggregates because applying operator $\Phi[\,\cdot\,]$ produces a new formula in which nested aggregates may occur both positively and negatively.

It is well known that GZ and F-semantics do not agree even in the case of monotonic aggregates as illustrated by the example in the introduction. Nevertheless, we identify next a more restricted family of aggregates for which both semantics coincide.

Proposition 11. *Any GZ-aggregate A of the following types satisfies $A \equiv_s \mathtt{F}[A]$:*

i) $A = (\mathtt{count}\{\mathbf{X} : cond(\mathbf{X})\} = n)$

ii) $A = (\mathtt{sum}\{\mathbf{X} : cond(\mathbf{X})\} = n)$ *when:* $\mathcal{C} \cap \mathbb{Z}_{\leq 0} = \emptyset$, *or* $\mathcal{C} \cap \mathbb{Z}_{\geq 0} = \emptyset$. □

Note that the result $ii)$ of Proposition 11 does not apply when 0 is in the domain. For instance, the program consisting of the rule $p(0) \leftarrow \mathtt{sum}\{\mathbf{X} : p(\mathbf{X})\} = 0$ has a unique stable model $\{p(0)\}$ under Ferraris' semantic but no stable model under GZ's one.

5 Conclusions

We have provided a (strong equivalence preserving) translation from logic programs with GZ-aggregates to propositional theories in *Equilibrium Logic*. Once we understand aggregates as propositional formulas, it is straightforward to extend the syntax to arbitrary nesting of aggregates (both GZ and F-aggregates) plus propositional connectives, something we called *aggregate theories*. We have provided two alternative semantics for these theories: one based on a direct, combined extension of GZ and F-reducts, and the other on a translation to propositional formulas. The propositional formula translation has helped us to characterise the effect (with respect to the obtained stable models) of replacing a GZ-aggregate by its corresponding F-aggregate. Moreover, we have been able to prove that both aggregates have the same behaviour in the scope of negation. Finally, we identified a class of aggregates in which the GZ and F-semantics coincide. It is worth to mention that a close look at the proof of this result also suggests an extension to a broader class, something that will be studied in the future.

We expect that the current propositional formula translations will open new possibilities to explore formal properties and potential implementations of both GZ and F-aggregates, possibly extending the idea of [2] to our general aggregate theories. Finally, an extension of the current approach to a full first-order language with partial, evaluable functions (such as [4]), is currently under development.

Proofs of Propositions 8 and 11

Proof of Proposition 8. Note that, from Proposition 6, if $H \models_{cl} A^T$ iff $H \models_{cl} \Phi[A]^T$ holds then $A \equiv_s \Phi[A]$ holds. Hence, it is enough to show $H \models_{cl} A^T$ iff $H \models_{cl} \Phi[A]^T$.

Let us show that $H \models_{cl} A^T$ implies $H \models_{cl} \Phi[A]^T$ for any pair of classical interpretations such that $H \subseteq T$. Note that $T \not\models_{cl} A$ implies that $A^T = \bot$ and, thus, $H \not\models_{cl} A^T$ and the statement holds vacuous.

Then, we may assume without loss of generality that $T \models_{cl} A$ and, thus, to show that $H \models_{cl} \Phi[A]^T$, it is enough to show that the following two conditions hold:
(a) $H \models_{cl} \quad cond(\mathbf{c})^T$ for every $\mathbf{c} \in \mathcal{C}^{\mathbf{X}}$ s.t. $T \models_{cl} cond(\mathbf{c})$, and
(b) $H \models_{cl} (\neg cond(\mathbf{c}))^T$ for every $\mathbf{c} \in \mathcal{C}^{\mathbf{X}}$ s.t. $T \not\models_{cl} cond(\mathbf{c})$
Furthermore, by definition, $T \models_{cl} A$ implies

$$A^T \;=\; \bigwedge \{\; cond(\mathbf{c})^T \;\big|\; T \models_{cl} cond(\mathbf{c}) \text{ with } \mathbf{c} \in \mathcal{C}^{\mathbf{X}} \;\} \tag{8}$$

and, thus, $H \models_{cl} A^T$ implies that (a) holds. Moreover, $T \not\models_{cl} cond(\mathbf{c})$ implies $cond(\mathbf{c})^T = \bot$ which, in its turn, implies $H \models_{cl} (\neg cond(\mathbf{c}))^T$ and, thus, (b) follows.

The other way around. Assume that $H \models_{cl} \Phi[A]^T$. Then, there is $I \models_{cl} A$ satisfying the following two conditions:
(c) $H \models_{cl} \quad cond(\mathbf{c})^T$ for every $\mathbf{c} \in \mathcal{C}^{\mathbf{X}}$ such that $I \models_{cl} cond(\mathbf{c})$, and
(d) $H \models_{cl} (\neg cond(\mathbf{c}))^T$ for every $\mathbf{c} \in \mathcal{C}^{\mathbf{X}}$ such that $I \not\models_{cl} cond(\mathbf{c})$
From Proposition 6 and the fact that $H \subseteq T$, it follows that $H \models_{cl} cond(\mathbf{c})^T$ implies that $T \models_{cl} cond(\mathbf{c})$ and, thus, (c) implies
(c') $T \models_{cl} cond(\mathbf{c})$ for every $\mathbf{c} \in \mathcal{C}^{\mathbf{X}}$ s.t. $I \models_{cl} cond(\mathbf{c})$
Similarly, $H \models_{cl} (\neg cond(\mathbf{c}))^T$ implies $T \models_{cl} (\neg cond(\mathbf{c}))$ which, in its turn, implies that $T \not\models_{cl} cond(\mathbf{c})$. Thus, (d) implies
(d') $T \not\models_{cl} cond(\mathbf{c})$ for every $\mathbf{c} \in \mathcal{C}^{\mathbf{X}}$ such that $I \not\models_{cl} cond(\mathbf{c})$
From (c') and (d'), it follows that
(e) $T \models_{cl} cond(\mathbf{c})$ iff $I \models_{cl} cond(\mathbf{c})$ holds for every $\mathbf{c} \in \mathcal{C}^{\mathbf{X}}$
In its turn, this implies that $T \models_{cl} A$ iff $I \models_{cl} A$ and, since $I \models_{cl} A$, it follows that $T \models_{cl} A$ and, thus, we have that (8) holds (note that $T \not\models_{cl} A$ would imply that $A^T = \bot$). Then, to show that $H \models_{cl} A^T$ is enough to show that $H \models_{cl} cond(\mathbf{c})^T$ for every $\mathbf{c} \in \mathcal{C}^{\mathbf{X}}$ such that $T \models cond(\mathbf{c})$ which follows from (c) and (e). □

Proof of Proposition 11

We will need the following notation. Let $\prec$ denote any relation symbol. We say that $A = (f\{\mathbf{X} : cond(\mathbf{X})\} \prec n)$ is *monotone* (resp. *antimonotone*) iff $\hat{f}(W_1) \prec n$ implies $\hat{f}(W_2) \prec n$ for all sets of tuples $W_1 \subseteq W_2 \subseteq \mathcal{C}^{|\mathbf{X}|}$ (resp. $W_2 \subseteq W_1 \subseteq \mathcal{C}^{|\mathbf{X}|}$). It is *regular* iff for any pair W_1, W_2 of sets of tuples of constants s.t. $W_1 \subset W_2$ satisfies that either $\hat{f}(W_1) \not\prec n$ or $\hat{f}(W_2) \not\prec n$. Furthermore, by $A^{\geq}$ we denote the GZ-aggregate $f^{\geq}\{\mathbf{X} : cond(\mathbf{X})\} \geq n$ "testing" the greater or equal relation, that is, its function $\hat{f}^{\geq}$ is defined so that $\hat{f}^{\geq}(W') \geq n$ iff $\hat{f}(W) \prec n$ and $W' \supseteq W$. Analogolusly, $A^{\leq}$ stands for $f^{\leq}\{\mathbf{X} : cond(\mathbf{X})\} \leq n$ whose function $\hat{f}^{\leq}$ is defined so that $\hat{f}^{\leq}(W') \leq n$ iff $\hat{f}(W) \prec n$ and $W' \subseteq W$. Then, by $\Phi^{\mathrm{D}}[A]$, we denote the following formula:

$$\bigwedge_{T \not\in [\![\, A^{\leq} \,]\!]} \Big(\neg \bigwedge \mathtt{Gr}^{+}_{T}(cond(\mathbf{X})) \Big) \wedge \bigwedge_{T \not\in [\![\, A^{\geq} \,]\!]} \Big(\bigvee \mathtt{Gr}^{-}_{T}(cond(\mathbf{X})) \Big) \tag{9}$$

Proposition 12. *Any regular set atom A satisfies:*
i) set atoms $A^{\geq}$ and $A^{\leq}$ are respectively monotone and antimonotone,

ii) $T \models_{cl} A$ *iff* $T \models_{cl} A^{\geq} \wedge A^{\leq}$
iii) $\Phi[\mathtt{F}[A]] \equiv_s \Phi[\mathtt{F}[A^{\geq}]] \wedge \Phi[\mathtt{F}[A^{\leq}]] \equiv_s \Phi^{\mathtt{D}}[\mathtt{F}[A]]$. □

For $A = f\{\mathbf{X} : cond(\mathbf{X})\} \prec n$, we define $W_{\langle H,T\rangle}(A) \stackrel{\text{def}}{=} \{\ \mathbf{c} \mid \langle H,T\rangle \models cond(\mathbf{c})\ \}$. We also use $W_T(A)$ to stand for $W_{\langle T,T\rangle}(A)$.

Proposition 13. *Any set atom* $A = f\{\mathbf{X} : cond(\mathbf{X})\} \prec c$ *with propositional* $cond(\mathbf{X})$ *satisfies that i)* $\langle H,T\rangle \models \Phi[\mathtt{F}[A]]$ *iff ii)* $H \models_{cl} \mathtt{F}[A]^T$ *iff iii)* $\hat{f}(W_{\langle H,T\rangle}(A)) \prec n$ *and* $\hat{f}(W_T(A)) \prec n$. □

Lemma 1. *Any regular GZ-aggregate* A *satisfies:* $\Phi[\mathtt{F}[A]] \models \Phi[A]$. □

Proof. Suppose, for the sake of contradiction, that there is some HT-interpretation such that $\langle H,T\rangle \models \Phi[\mathtt{F}[A]]$, but $\langle H,T\rangle \not\models \Phi[A]$. Suppose also that $T \models_{cl} A$. Then, from $\langle H,T\rangle \not\models \Phi[A]$, it follows that one of the following conditions must hold

(a) $\langle H,T\rangle \not\models \quad cond(\mathbf{c})$ for some $\mathbf{c} \in \mathcal{C}^{|\mathbf{X}|}$ s.t. $T \models_{cl} cond(\mathbf{c})$,
(b) $\langle H,T\rangle \not\models \neg cond(\mathbf{c})$ for some $\mathbf{c} \in \mathcal{C}^{|\mathbf{X}|}$ s.t. $T \not\models_{cl} cond(\mathbf{c})$

On the one hand, $\langle H,T\rangle \not\models \neg cond(\mathbf{c})$ implies $T \not\models_{cl} \neg cond(\mathbf{c})$ which, in its turn, implies $T \models_{cl} cond(\mathbf{c})$. Thus, (b) is a contradiction. On the other hand, from Proposition 1, it follows that $W_{\langle H,T\rangle} \subseteq W_T$ holds for any HT-interpretation $\langle H,T\rangle$. Then, (a) implies that $W_{\langle H,T\rangle} \subset W_T$ which, since A is regular, implies that either $\hat{f}(W_{\langle H,T\rangle}) \not\prec n$ or $\hat{f}(W_T) \not\prec n$ hold. If the former, then $\langle H,T\rangle \not\models \Phi[\mathtt{F}[A]]$ (Proposition 13) which is a contradiction with the assumption $\langle H,T\rangle \models \Phi[\mathtt{F}[A]]$. If the latter, $T \not\models_{cl} A \equiv_{cl} \Phi[A]$ which is a contradiction with the facts $T \models_{cl} \Phi[\mathtt{F}[A]]$ (because $\langle H,T\rangle \models \Phi[\mathtt{F}[A]]$) and $\Phi[A] \equiv_{cl} \Phi[\mathtt{F}[A]]$. Hence, it must be that $T \not\models A$ and, thus, Proposition 12, implies:

(c) either $T \not\models A^{\geq}$ or $T \not\models A^{\leq}$, and
(d) $\langle H,T\rangle \models \Phi[\mathtt{F}[A]] \equiv_s \Phi^{\mathtt{D}}[\mathtt{F}[A]]$

In its turn, these conditions imply that one of the following two contradictions must hold:

(e) $\langle H,T\rangle \models \neg cond(\mathbf{c})$ for some $\mathbf{c} \in \mathcal{C}^{|\mathbf{X}|}$ s.t. $T \models cond(\mathbf{c})$ (if $T \not\models A^{\leq}$)
(f) $\langle H,T\rangle \models \quad cond(\mathbf{c})$ for some $\mathbf{c} \in \mathcal{C}^{|\mathbf{X}|}$ s.t. $T \not\models cond(\mathbf{c})$ (if $T \not\models A^{\geq}$),

Consequently, $\langle H,T\rangle \models \Phi[\mathtt{F}[A]]$ implies $\langle H,T\rangle \models \Phi[A]$. □

Proof of Proposition 11. From Proposition 10 and Lemma 1, it respectively follows $\Phi[A] \models \Phi[\mathtt{F}[A]]$ and $\Phi[\mathtt{F}[A]] \models \Phi[A]$. Thus, $\Phi[A] \equiv_s \Phi[\mathtt{F}[A]]$. It only remains to note that $\widehat{\mathtt{count}}(W_1) < \widehat{\mathtt{count}}(W_2)$ for all $W_1 \subset W_2$ and, thus, $\mathtt{count}\{\mathbf{X} : cond(\mathbf{X})\} = n$ is regular. Similarly, if there is no $c \in \mathcal{C}$ such that $c \leq 0$ (resp. $c \geq 0$), then $\widehat{\mathtt{sum}}(W_1) < \widehat{\mathtt{sum}}(W_2)$ (resp. $\widehat{\mathtt{sum}}(W_1) > \widehat{\mathtt{sum}}(W_2)$) □

References

1. F. Aguado, P. Cabalar, D. Pearce, G. Pérez, and C. Vidal. A denotational semantics for equilibrium logic. *Theory and Practice of Logic Programming*, 15(4-5):620–634, 2015.
2. M. Alviano and W. Faber. Stable model semantics of abstract dialectical frameworks revisited: A logic programming perspective. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 2684–2690. AAAI Press, 2015.

3. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
4. P. Cabalar. Functional answer set programming. *Theory and Practice of Logic Programming*, 11(2-3):203–233, 2011.
5. F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, and T. Schaub. ASP-Core-2: Input language format. Available at https://www.mat.unical.it/aspcomp2013/ASPStandardization/, 2012.
6. E. Erdem, M. Gelfond, and N. Leone. Applications of ASP. *AI Magazine*, 37(3):53–68, 2016.
7. W. Faber, G. Pfeifer, and N. Leone. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence*, 175(1):278–298, 2011.
8. W. Faber, G. Pfeifer, N. Leone, T. Dell'Armi, and G. Ielpa. Design and implementation of aggregate functions in the DLV system. *Theory and Practice of Logic Programming*, 8(5-6):545–580, 2008.
9. P. Ferraris. Answer sets for propositional theories. In *Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05)*, pages 119–131. Springer-Verlag, 2005.
10. P. Ferraris. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic*, 12(4):25, 2011.
11. P. Ferraris, J. Lee, and V. Lifschitz. A new perspective on stable models. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 372–379. AAAI/MIT Press, 2007.
12. M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188:52–89, 2012.
13. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, pages 1070–1080. MIT Press, 1988.
14. M. Gelfond and Y. Zhang. Vicious circle principle and logic programs with aggregates. *Theory and Practice of Logic Programming*, 14(4-5):587–601, 2014.
15. A. Harrison, V. Lifschitz, and M. Truszczynski. On equivalence of infinitary formulas under the stable model semantics. *Theory and Practice of Logic Programming*, 15(1):18–34, 2015.
16. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.
17. D. Pearce. A new logical characterisation of stable models and answer sets. In *Proceedings of the Sixth International Workshop on Non-Monotonic Extensions of Logic Programming (NMELP'96)*, pages 57–70. Springer-Verlag, 1997.
18. D. Pearce. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):3–41, 2006.
19. N. Pelov, M. Denecker, and M. Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming*, 7(3):301–353, 2007.
20. P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.
21. T. Son and E. Pontelli. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming*, 7(3):355–375, 2007.