Gelfond-Zhang aggregates as propositional formulas

Pedro Cabalar[a], Jorge Fandinno[a,d,2], Torsten Schaub[b,c,4], Sebastian Schellhorn[b]

[a] *University of Corunna, Spain*
[b] *University of Potsdam, Germany*
[c] *INRIA, Rennes, France*
[d] *IRIT, Université de Toulouse, CNRS, Toulouse, France*

Abstract

Answer Set Programming (ASP) has become a popular and widespread paradigm for practical Knowledge Representation thanks to its expressiveness and the available enhancements of its input language. One of such enhancements is the use of aggregates, for which different semantic proposals have been made. In this paper, 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).

Email addresses: `cabalar@udc.es` (Pedro Cabalar), `jorge.fandino@udc.es`, `jorge.fandinno@irit.fr` (Jorge Fandinno), `torsten@cs.uni-potsdam.de` (Torsten Schaub), `seschell@cs.uni-potsdam.de` (Sebastian Schellhorn)

[1]Partially supported by grants GPC-ED431B 2016/035 and 2016-2019 ED431G/01 (Xunta de Galicia, Spain), and TIN 2013-42149-P, TIN 2017-84453-P (MINECO, Spain).

[2]Funded by the Centre International de Mathématiques et d'Informatique de Toulouse through contract ANR-11-LABEX-0040-CIMI within the program ANR-11-IDEX-0002-02.

[3]This work was partially funded by DFG grant SCHA 550/9.

[4]Affiliated with the School of Computing Science at Simon Fraser University, Canada, and the Institute for Integrated and Intelligent Systems at Griffith University, Australia.

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.

Keywords: Aggregates, Answer Set Programming

1. Introduction

Knowledge Representation and Reasoning (KRR) has constituted one of the central areas of Artificial Intelligence since its very beginning and, in particular, logic-based approaches have attracted most of the interest in the literature. Although classical logic copes with many of the usual desiderata for KRR such as simplicity, clear semantics or well-known inference methods and their associated complexity, it was soon detected to fall short for practical purposes in different aspects. For instance, one crucial feature in many KRR scenarios, even in extremely simple ones, is the need to draw conclusions in absence of information, something impossible in classical logic due to its monotonic inference relation. As a result, since the introduction of the three well-known approaches, *circumscription* [1], *default logic* [2] and modal *non-monotonic logics* [3] in the historical special issue of the *Artificial Intelligence* journal in 1980, Non-Monotonic Reasoning (NMR) has centered great part of the discussion in KRR.

Nowadays, *Answer Set Programming* (ASP [4]) has become an established problem-solving paradigm and a prime candidate for practical KRR. The reasons for this success are manifold. The most obvious one is the availability of effective solvers [5, 6] and a growing list of covered application domains [7]. A probably equally important reason is its declarative semantics, having been generalized from the original *stable models* [8] of normal logic programs up to arbitrary first-order [9, 10] and infinitary [11] formulas. Several logical characterizations of ASP have been obtained, among which *Equilibrium Logic* [12] 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 NMR. Finally, a third relevant cause of ASP's success lies in its flexible specification language [13], 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. To understand the utility of this kind of constraints think about representing in first-order logic, for instance, that predicate p is satisfied by at least $n = 3$ different individuals. A typical formalization would look like:

$$\exists x \exists y \exists z\ (p(x) \wedge p(y) \wedge p(z) \wedge x \neq y \wedge y \neq z \wedge x \neq z)$$

This pattern becomes obviously cumbersome when increasing the number n of individuals, as the number of required inequalities combinatorily explodes. Moreover, the formula above cannot be constructed if n is an arbitrary parameter whose value is not known beforehand. On the other hand, this same meaning can be simply captured by a natural ASP aggregate of the form $\texttt{count}\{X : p(X)\} \geq n$.

Unfortunately, there is no clear agreement on the expected behavior of aggregates in ASP, and several alternative semantics have been defined [14, 15, 16, 17, 18, 19], among which perhaps Ferraris' [17] and Faber et al's [18] are the two more consolidated ones due to their respective implementations in the ASP solvers `clingo` [5] and `DLV` [6]. 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 [19] (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 [20], 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:

$$\begin{array}{lll} p(b). & & p(a) \leftarrow p(a) \wedge \neg p(b). \\ p(a) \leftarrow \neg p(a) \wedge p(b). & & p(a) \leftarrow p(a) \wedge p(b). \end{array} \tag{3}$$

Again, it is classically equivalent to the F-translation (2), but quite different in logic programming, where the left rules 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. In Section 5, we lift the fragment in which both semantics coincide to their first-order languages: $\mathcal{A}log$ and `gringo` [21]. Finally, Section 6 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 $\mathit{SM}(\Gamma)$ denote the stable models of Γ. Two theories Γ, Γ' are *strongly equivalent*, written $\Gamma \equiv_s \Gamma'$, iff $\mathit{SM}(\Gamma \cup \Delta) = \mathit{SM}(\Gamma' \cup \Delta)$ 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[5] $\Gamma(\varphi) \subseteq \mathcal{L}$. 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:

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

We use Greek letters φ and ψ and their variants to stand for propositional formulas. We define the derived operators $\neg\varphi \stackrel{\text{def}}{=} (\varphi \to \bot)$, $\varphi \leftrightarrow \psi \stackrel{\text{def}}{=} (\varphi \to \psi) \wedge (\psi \to \varphi)$ and $\top \stackrel{\text{def}}{=} \neg\bot$.

[5]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.

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 $H \subseteq T \subseteq At$.

Definition 1 (HT-satisfaction)**.** *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:*

- $\langle H, T \rangle \not\models \bot$
- $\langle H, T \rangle \models p$ *iff* $p \in H$, *for any atom* p
- $\langle H, T \rangle \models \varphi_1 \wedge \varphi_2$ *iff* $\langle H, T \rangle \models \varphi_1$ *and* $\langle H, T \rangle \models \varphi_2$
- $\langle H, T \rangle \models \varphi_1 \vee \varphi_2$ *iff* $\langle H, T \rangle \models \varphi_1$ *or* $\langle H, T \rangle \models \varphi_2$
- $\langle H, T \rangle \models \varphi_1 \rightarrow \varphi_2$ *iff both:*
  (i) $T \models_{cl} \varphi_1 \rightarrow \varphi_2$ *and*
  (ii) $\langle H, T \rangle \not\models \varphi_1$ *or* $\langle H, T \rangle \models \varphi_2$ $\square$

It is not difficult to see that $\langle T, T \rangle \models \varphi$ iff $T \models_{cl} \varphi$. 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 2 (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.* $\square$

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 $T \models_{cl} \neg\varphi$ $\square$

Definition 3 (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 [22]). *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 [23], that is, $\varphi \equiv_s \psi$ iff $\varphi \equiv \psi$ in that language. The following result follows from Corollary 3 in [24].

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

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 $(p \vee q) \models (\neg p \to q)$. As they are classically equivalent, $SM(p \vee q) \supseteq SM(\neg p \to q)$ which is not such a strong result. However, since HT-entailment is monotonic with respect to conjunction, it follows that $(p \vee q) \wedge \gamma \models (\neg p \to q) \wedge \gamma$ also holds for any γ, and thus, if we replace a disjunctive rule $(p \vee q)$ by $(\neg p \to q)$ 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, $(\neg p \to q) \wedge \gamma$ has a subset of stable models. On the other hand, it occurs negatively in $(p \vee q) \to \gamma$, and so, $(\neg p \to q) \to \gamma$ has a superset of stable models.

3. Aggregates as formulas

To deal with aggregates, we consider a simplified first-order[6] signature $\Sigma = \langle \mathcal{C}, \mathcal{A}, \mathcal{P}\rangle$ formed by three pairwise disjoint sets respectively called *constants*, *aggregate*

[6]An extension to a full first-order language is under development.

symbols and *predicate symbols*. An *arithmetic term* is a combination of numerical constants, variables and arithmetic operators built in the usual way. A *term* is either a constant $c \in \mathcal{C}$, a variable X or an arithmetic term. We use the vector overline to represent tuples of terms, such as $\vec{t}$, and write $|\vec{t}|$ to stand for the tuple's arity. As usual, a *predicate atom* is an expression of the form $p(\vec{t})$ where $\vec{t}$ is a tuple of terms; an *arithmetic atom* is an expression of the form $t \prec t'$ with t and t' arithmetic terms and $\prec \in \{=, \neq, \leq, \geq, <, >\}$ an *arithmetic relation*. A *regular atom* is either a predicate atom or an arithmetic atom. An *(aggregate) formula* φ is recursively defined by the following grammar:

$$\varphi ::= \bot \mid a \mid f\{\vec{X} : \varphi\} \prec t \mid f\{\vec{c}:\varphi, \dots, \vec{c}:\varphi\} \prec t \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi$$

where a is regular atom, $f \in \mathcal{A}$ is an aggregate symbol, $\vec{X}$ is a non-empty tuple of variables, $\vec{c}$ is a non-empty tuple of constants, $\prec \in \{=, \neq, \leq, \geq, <, >\}$ is an arithmetic relation, and t is an arithmetic term. A *(aggregate) theory* is a set of aggregate formulas. As we can see, we distinguish two types of *aggregates*: $f\{\vec{X} : \varphi\} \prec t$ called *GZ-aggregates* (or *set atoms*); and $f\{\vec{c}_1 : \varphi_1, \dots, \vec{c}_m : \varphi_m\} \prec t$, with all $\vec{c}_i$ of same arity, called *F-aggregates*. This syntactic distinction respects the original syntax[7] 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 $\varphi_1, \dots, \varphi_m$ 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 [17], 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 $\vec{X}$. In fact, a formula φ inside $A = f\{\vec{X} : \varphi\} \prec t$ (called the *condition* of A) normally contains occurrences of $\vec{X}$, so we usually write it as $\mathit{cond}(\vec{X})$. Moreover,

[7]Ferraris actually uses $\varphi_i = w$ rather than $\vec{c}:\varphi$, but this is not a substantial difference, assuming w is the first element in tuple $\vec{c}$.

the occurrences of variables $\vec{X}$ in A are said to be *bound* to A. A variable occurrence X in a formula φ is *free* if it is not bound to any GZ-aggregate in φ. An *atom* is either a regular atom (predicate or arithmetic atom) or an aggregate. A *(regular) literal* is either a (regular) atom a (positive literal) or its default negation *not* a (negative literal). A predicate atom $p(\vec{t})$ is said to be *ground* iff all its terms are constants $\vec{t} \subseteq \mathcal{C}^{|\vec{t}|}$; an arithmetic atom $t \prec t'$ is said to be *ground* iff t and t' are numbers; and an aggregate atom $f\{\dots\} \prec t$ is said to be *ground* iff it contains no free variables[8] and t is a number. We write $At(\mathcal{C}, \mathcal{P})$ to stand for the set of ground atoms for predicates $\mathcal{P}$ and constants $\mathcal{C}$. A theory is said to be *ground* iff all atoms occurring in it are ground. We define the grounding of a formula $\varphi(\vec{X})$ with free variables $\vec{X}$ as expected: $\texttt{Gr}(\ \varphi(\vec{X})\ ) \overset{\text{def}}{=} \{\varphi(\vec{c}) \mid \vec{c} \in \mathcal{C}^{|\vec{X}|}\}$ where $\varphi(\vec{c})$ is the result of replacing all occurrences of the variables $\vec{X}$ in $\varphi(\vec{X})$ by the constants in $\vec{c}$ and evaluating all arithmetic terms. Similarly, by $\texttt{Gr}(\Gamma) \overset{\text{def}}{=} \bigcup\{\texttt{Gr}(\varphi(\vec{X})) \mid \varphi(\vec{X}) \in \Gamma\}$ we denote the grounding of any theory Γ. Until Section 5, we will exclusively deal with ground theories. This is not a limitation, since a non-ground formula $\varphi(\vec{X})$ in some theory can be understood as an abbreviation of its grounding $\texttt{Gr}(\varphi(\vec{X}))$, 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 $\texttt{sum}, \texttt{count}, \texttt{max}$, etc. For example, for aggregate symbol $\texttt{sum}$ and arity $m = 1$ the function returns the aggregate addition when the set consists of (1-tuples of) integer numbers. For instance, $\widehat{\texttt{sum}}_1(\{7, 2, -4\}) = 5$ and $\widehat{\texttt{sum}}_1(\emptyset) = 0$ but $\widehat{\texttt{sum}}_1(\{7, a, 3, b\})$ 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, $\widehat{\texttt{sum}}_2(\{\langle 7, a\rangle, \langle 2, b\rangle, \langle 2, a\rangle\}) = 11$. 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 4. *A classical interpretation* T satisfies *a formula* φ*, denoted by*

[8]Note that ground aggregates may still contain variables, but bound to the aggregate.

$T \models_{cl} \varphi$, when the following recursive conditions hold:

- $T \not\models_{cl} \bot$
- $T \models_{cl} p(\vec{c})$ *iff* $p(\vec{c}) \in T$ *for any ground atom* $p(\vec{c})$
- $T \models_{cl} f\{\vec{X} : cond(\vec{X})\} \prec n$ *iff* $\hat{f}_{|\vec{X}|}\big(\{\ \vec{c} \in \mathcal{C}^{|\vec{X}|}\ \mid\ T \models_{cl} cond(\vec{c})\ \}\big)$ *has some value* $k \in \mathbb{Z}$ *and* $k \prec n$ *under the usual meaning of arithmetic relation* $\prec$
- $T \models_{cl} f\{\vec{c}_1 : \varphi_1, \ldots, \vec{c}_m : \varphi_m\} \prec n$ *iff* $\hat{f}_{|\vec{c}_1|}\big(\{\ \vec{c}_i\ \mid\ T \models_{cl} \varphi_i\ \}\big)$ *has some value* $k \in \mathbb{Z}$ *and* $k \prec n$, *again, under its usual meaning*
- $T \models_{cl} \varphi_1 \wedge \varphi_2$ *iff* $T \models_{cl} \varphi_1$ *and* $T \models_{cl} \varphi_2$
- $T \models_{cl} \varphi_1 \vee \varphi_2$ *iff* $T \models_{cl} \varphi_1$ *or* $T \models_{cl} \varphi_2$
- $T \models_{cl} \varphi_1 \to \varphi_2$ *iff* $T \not\models_{cl} \varphi_1$ *or* $T \models_{cl} \varphi_2$

We say that T is a (classical) 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:

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

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, $\texttt{Gr}^+_T(\varphi)$ collects the formulas from $\texttt{Gr}(\varphi)$ satisfied by T.

Definition 5 (reduct)**.** *We will define the* reduct *of a GZ-aggregate formula $A = f\{\vec{X} : cond(\vec{X})\} \prec n$ with respect to a classical interpretation T, denoted as A^T, in the following way:*

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

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

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

The reduct of any other formula is just as in Definition 3. The reduct of a theory is the set of reducts of its formulas. □

Definition 6 (stable model)**.** *A classical interpretation T is a* stable model *of a theory Γ iff T is a $\subseteq$-minimal model of Γ^T.* $\square$

Note that, when restricted to F-formulas, Definition 3 exactly matches the reduct definition for aggregate theories by Ferraris [17]. On the other hand, when restricted to GZ-formulas, it generalises the reduct definition by Gelfond-Zhang [19] allowing arbitrary formulas in $\mathit{cond}(\vec{X})$, including nested aggregates. For this reason, in our setting, the reduct is recursively applied to $(\bigwedge \mathtt{Gr}^+_T(\mathit{cond}(\vec{X})))^T$. In the original case [19], $\mathit{cond}(\vec{X})$ was a conjunction of atoms, but it is straightforward to see that, then, $(\bigwedge \mathtt{Gr}^+_T(\mathit{cond}(\vec{X})))^T = \bigwedge \mathtt{Gr}^+_T(\mathit{cond}(\vec{X}))$. To sum up, the above definitions of stable model and reduct correspond to the original ones for Ferraris [17] and Gelfond-Zhang [19] when restricted to their respective syntactic fragments.

Proposition 5. *Stable models are classical models.* $\square$

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 $p(a) \leftarrow p(a) \wedge p(b)$, their least model being $\{p(b)\}$, 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 $A_1^T = \bot$.* $\square$

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

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

and imagine that $\mathit{owns}(X,Y)$ 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:*

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

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 $(\bigwedge \mathtt{Gr}^+_T(\mathtt{sum}\{Y : \mathit{owns}(X,Y)\} \geq 10))^T$. After grounding free variable X, we obtain:*

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

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: $owns(a,6) \wedge owns(a,8) \wedge owns(c,12)$.* $\square$

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, $\varphi \equiv_s \psi$. $\square$

Proposition 6 generalises results from [17] 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 $B = f\{\vec{c}_1 : \varphi_1, \ldots, \vec{c}_m : \varphi_m\} \prec n$ 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 [17].

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

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. To this aim, we extend translation Φ to be also applicable to any GZ-aggregate $A = f\{\vec{X} : cond(\vec{X})\} \prec n$ with a propositional condition $cond(\vec{X})$, so that $\Phi[A]$ corresponds to the propositional formula:

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

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

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 $A_3 = \texttt{count}\{X : p(X)\} = 1$ 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:*

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

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 $A = f\{\vec{X} : cond(\vec{X})\} \prec n$ we define its corresponding F-aggregate $\texttt{F}[A]$:

$$\texttt{F}[A] \quad \stackrel{\text{def}}{=} \quad f\{\ \vec{c} : cond(\vec{c}) \ \mid\ cond(\vec{c}) \in \texttt{Gr}(cond(\vec{X}))\ \} \prec n \tag{6}$$

This correspondence is analogous to the process of *instantiation* used in [18] to ground aggregates with variables. It is easy to check that A and $\texttt{F}[A]$ are classically equivalent. This can be checked using satisfaction from Definition 4 or classical logic for their propositional representations $\Phi[A] \equiv_{cl} \Phi[\texttt{F}[A]]$. Moreover, it can be observed that these two logical representations are somehow *dual*. Indeed, $\Phi[\texttt{F}[A]]$ eventually amounts to:

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

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 $\texttt{F}[A]$ is that, due to Proposition 1- ii), we can safely replace one by another when negated. In other words:

Proposition 9. *Let φ be a formula with some occurrence of a GZ-aggregate A and let ψ be the result of replacing A by its corresponding F-aggregate $\texttt{F}[A]$ in φ. Then, we have that $\neg\varphi \equiv \neg\psi$.* □

However, as we saw in the introduction examples, replacing some GZ-aggregate A by its F-aggregate version $\texttt{F}[A]$ may change the program semantics. Still, the stable models obtained after such replacement are not arbitrary. As we said, [20] proved that if the GZ-aggregate A occurs in a positive rule body, then the replacement by the F-aggregate $\texttt{F}[A]$ 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 $\Phi[\texttt{F}[A]]$ in HT.

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

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

i) *$SM(\Gamma(A)) \supseteq SM(\Gamma(\texttt{F}[A]))$ for any theory $\Gamma(A)$ where occurrence A is positive;*

ii) *$SM(\Gamma(A)) \subseteq SM(\Gamma(\texttt{F}[A]))$ for any theory $\Gamma(A)$ where occurrence A is negative.* □

Proof. From $\Phi[A] \equiv_{cl} \Phi[\texttt{F}[A]]$ 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) $A = (\mathtt{count}\{\vec{X} : \mathit{cond}(\vec{X})\} = n)$

ii) $A = (\mathtt{sum}\{X, \vec{Y} : \mathit{cond}(X, \vec{Y}) \wedge X > 0\} = n)$

iii) $A = (\mathtt{sum}\{X, \vec{Y} : \mathit{cond}(X, \vec{Y}) \wedge X < 0\} = n)$. □

Note that the result *ii* (resp. *iii*) of Proposition 11 does not apply if the condition $X > 0$ (resp. $X < 0$) is dropped. For instance, the program consisting of the rule

$$p(0) \leftarrow \mathtt{sum}\{X : p(X)\} = 0 \tag{8}$$

has a unique stable model $\{p(0)\}$ under Ferraris' semantic but no stable model under GZ's one.

5. Relation between *𝒜log* and clingo aggregates

In this section, we restrict ourselves to the syntax of logic programming and lift the relation between GZ and F-aggregates to their respective first order languages, studying a syntactic fragment in which the semantics of *𝒜log* [19] and `gringo` [21] coincide. We also show that every *𝒜log* program whose aggregates are all of `count` type can be easily rewritten to this fragment in an human friendly way. The same applies to program that also contain `sum` aggregates provided that 0 does not occur as a constant in the program. It is worth mentioning that a compilation for GZ-aggregates into F-aggregates has already been described in the literature [25]. This compilation has the advantage of covering programs containing any kind of aggregates. On the other hand, it makes uses of new auxiliary atoms that obscure the semantics of the program. In this sense, this compilation is better suited for automatic translation while our rewriting, though less general, preserves human readability.

An $\mathcal{A}log$ *rule* is an expression of the form:

$$Head \leftarrow Pos \wedge Neg \wedge Agg \tag{9}$$

where *Head* is a disjunction of atoms, *Pos* is a conjunction of regular atoms, *Neg* is a conjunction of negative regular literals, and *Agg* is a conjunction of GZ-aggregates. An $\mathcal{A}log$ *program* is a set of $\mathcal{A}log$ rules. Recall that, in this section, we no longer assume that atoms or programs are ground. Note that every $\mathcal{A}log$ program is also a program in the syntax of `gringo`, though their semantic may differ. We also recall the notion of *global variable* from [21]: a variable is said to be *global* in a rule of the form of (9) iff it occurs in any literal in *Pos* or *Neg* or in any term t of any aggregate atom in *Agg* of the form $f\{\vec{X} : cond(\vec{X})\} \prec t$. An *instance* of a rule is obtained by replacing all global variables by constants. The `gringo` *grounding* of logic program Π, denoted $\mathtt{Gr}_{\mathtt{gringo}}(\Pi)$, is obtained by collecting all possible instances of its rules and replace every aggregate atom A by $\mathtt{F}[A]$. A set of atoms T is a `gringo` stable model of a program Π iff it is a stable model of $\mathtt{Gr}_{\mathtt{gringo}}(\Pi)$. Note that notions of global and free variables do not coincide: an occurrence of a variable may be both global and bound. This implies that the $\mathcal{A}log$ and `gringo` grounding of a program may be different and, as a result, the same program may have different stable models. To illustrate this fact, consider the following example from [19]:

Example 5. *Let P_2 consisting of the following rules*

$$\begin{aligned} & r \leftarrow \mathtt{count}\{X : p(X)\} \geq 2 \ \wedge \ q(X) \qquad (10)\\ & p(a)\\ & p(b)\\ & q(a) \end{aligned}$$

Variable X is both global in (10) *and bound in* $\mathtt{count}\{X : p(X)\} \geq 2$*. As a result, the $\mathcal{A}log$ grounding of P_2 is obtained by replacing rule* (10) *by rules*

$$\begin{aligned} & r \leftarrow \mathtt{count}\{X : p(X)\} \geq 2 \ \wedge \ q(a)\\ & r \leftarrow \mathtt{count}\{X : p(X)\} \geq 2 \ \wedge \ q(b) \end{aligned}$$

On the other hand, its clingo grounding is obtained by replacing the same rule by

$$r \leftarrow \texttt{count}\{a : p(a)\} \geq 2 \ \wedge \ q(a)$$
$$r \leftarrow \texttt{count}\{b : p(b)\} \geq 2 \ \wedge \ q(b)$$

Both programs have a unique stable model, but a different one: $\{p(a), p(b), q(a), r\}$ *for the former and* $\{p(a), p(b), q(a)\}$ *for the latter.* $\square$

We say that an aggregate atom is *closed* [26] iff no global variable occurs in it. We say that a rule (resp. program) is *closed* if all its aggregate atoms are closed. Then, from Proposition 11, we immediately get the following result for closed programs:

Proposition 12. *Let* Π *be a closed logic program where all aggregate atoms are of the form* $\texttt{count}\{\vec{X} : \mathit{cond}(\vec{X})\} = t$. *Then, the stable model of* Π *with respect to* $\mathcal{A}\mathit{log}$ *and* `gringo` *coincide.* $\square$

An interesting property of $\mathcal{A}\mathit{log}$ is that we can rewrite any logic program as an equivalent closed program where all aggregate conditions are equalities.

Definition 7. *Given a logic program* Π, *by* $\mathit{tr}_1(\Pi)$ *we denote the result of replacing*

i) every occurrence of a global variable that is also bound to some aggregate A *by a same new fresh variable not occurring anywhere else, and*

ii) every aggregate $f\{\vec{X} : \mathit{cond}(\vec{X}, \vec{Y})\} \prec t$ *with* $\prec$ *different from* $=$, *by the formula* $f\{\vec{X} : \mathit{cond}(\vec{X}, \vec{Y})\} = Z \wedge Z \prec t$ *with* Z *also a new fresh variable not occurring any where else.* $\square$

Proposition 13. *Given a logic program* Π, *the* $\mathcal{A}\mathit{log}$ *stable models of* Π *and* $\mathit{tr}_1(\Pi)$ *coincide.* $\square$

From Proposition 12 and Proposition 13 it immediately follows that we can rewrite any $\mathcal{A}\mathit{log}$ program where all aggregates are of the type `count` into an equivalent one in which its stable models coincide with the `gringo` stable models.

Theorem 3. *Given a logic program* Π *where all aggregate atoms are of the form* $\texttt{count}\{\vec{X} : \mathit{cond}(\vec{X})\} = t$, *then the* $\mathcal{A}\mathit{log}$ *stable models of* Π *coincide with the* `gringo` *stable models of* $\mathit{tr}_1(\Pi)$. $\square$

Example 6 (Ex. 5 continued)**.** *As mentioned above, P_2 is a program whose stable models are different according to $\mathcal{A}log$ and* `gringo` *semantics. On the other hand, we have that $tr_1(P_2)$ is*

$$\begin{aligned} & r \ \leftarrow \ \texttt{count}\{Y : p(Y)\} = Z \ \wedge \ Z \geq 2 \ \wedge \ q(X) \\ & p(a) \\ & p(b) \\ & q(a) \end{aligned} \tag{11}$$

whose unique stable model is $\{p(a), p(b), q, r\}$ according to both semantics. Recall that this is the the unique stable model of P_2 according to $\mathcal{A}log$. As a further example, let P_3 be the logic program consisting of rule (1) with $n = 1$. Recall from the introduction that P_3 has a unique stable model $\{p(a)\}$ according to Ferraris semantics while it does not have any stable model semantics according to Gelfond and Zhang semantics. Note that, since this program is ground, `gringo` *and $\mathcal{A}log$ semantics respectively coincide with Ferraris and Gelfond and Zhang semantics as described in Section 3. Furthermore, we have that $tr_1(P_3)$ is*

$$p(a) \leftarrow \texttt{count}\{X : p(X)\} = Z \wedge Z \geq 0$$

which has no stable model under both semantics. □

Note that, in general, the rewriting $tr_1(\cdot)$ is not safe for other kinds of aggregates with an associated function f for which there exist sets S and S' with $S \subset S'$ such that $f(S) = f(S')$. For instance, the sum of the empty set and the set $\{1, -1\}$ is in both cases 0 and, as a result, we have that programs involving sum over these two sets will have different stable models.

Example 7. *Let P_4 be the logic program consisting of the following rules:*

$$\begin{aligned} p(1) \ &\leftarrow \ \texttt{sum}\{X : p(X)\} = 0 \\ p(1) \ &\leftarrow \ p(-1) \\ p(-1) \ &\leftarrow \ p(1) \end{aligned} \tag{12}$$

It is easy to check that $tr_1(P_4) = P_4$, but this program has no stable model under the $\mathcal{A}log$ semantics and has a unique stable model $\{p(1), p(-1)\}$ under the `gringo` *semantics.* □

6. 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 propositional formula[9] equivalent to Son and Pontelli aggregates was also given in [16]. 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 [20] to our general aggregate theories. Finally, an extension of the current approach to a full first-order language with partial, evaluable functions (as those in [27]) was developed in [28]. This allows treating aggregates as ordinary first-order terms and combine them with arbitrary predicates, not just arithmetic relations.

7. References

[1] J. McCarthy, Circumscription: A form of non-monotonic reasoning, Artificial Intelligence 13 (1–2) (1980) 27–39.

[2] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1–2) (1980) 81–132.

[3] D. V. McDermott, J. Doyle, Non-monotonic logic I, Artificial Intelligence 13 (1–2) (1980) 41–72.

[9]This formula is classically equivalent to the ones for GZ and F-aggregates, but there is no pairwise HT-equivalence among the three of them.

[4] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003.

[5] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, Artificial Intelligence 187-188 (2012) 52–89.

[6] W. Faber, G. Pfeifer, N. Leone, T. Dell'Armi, G. Ielpa, Design and implementation of aggregate functions in the DLV system, Theory and Practice of Logic Programming 8 (5-6) (2008) 545–580.

[7] E. Erdem, M. Gelfond, N. Leone, Applications of ASP, AI Magazine 37 (3) (2016) 53–68.

[8] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, K. Bowen (Eds.), Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88), MIT Press, 1988, pp. 1070–1080.

[9] D. Pearce, Equilibrium logic, Annals of Mathematics and Artificial Intelligence 47 (1-2) (2006) 3–41.

[10] P. Ferraris, J. Lee, V. Lifschitz, A new perspective on stable models, in: M. Veloso (Ed.), Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07), AAAI/MIT Press, 2007, pp. 372–379.

[11] A. Harrison, V. Lifschitz, M. Truszczynski, On equivalence of infinitary formulas under the stable model semantics, Theory and Practice of Logic Programming 15 (1) (2015) 18–34.

[12] D. Pearce, A new logical characterisation of stable models and answer sets, in: J. Dix, L. Pereira, T. Przymusinski (Eds.), Proceedings of the Sixth International Workshop on Non-Monotonic Extensions of Logic Programming (NMELP'96), Vol. 1216 of Lecture Notes in Computer Science, Springer-Verlag, 1997, pp. 57–70.

[13] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, T. Schaub, ASP-Core-2: Input language format, Available at `https://www.mat.unical.it/aspcomp2013/ASPStandardization` (2012).

[14] P. Simons, I. Niemelä, T. Soininen, Extending and implementing the stable model semantics, Artificial Intelligence 138 (1-2) (2002) 181–234.

[15] N. Pelov, M. Denecker, M. Bruynooghe, Well-founded and stable semantics of logic programs with aggregates, Theory and Practice of Logic Programming 7 (3) (2007) 301–353.

[16] T. C. Son, E. Pontelli, A constructive semantic characterization of aggregates in answer set programming, Theory and Practice of Logic Programming 7 (3) (2007) 355–375.

[17] P. Ferraris, Logic programs with propositional connectives and aggregates, ACM Transactions on Computational Logic 12 (4) (2011) 25.

[18] W. Faber, G. Pfeifer, N. Leone, Semantics and complexity of recursive aggregates in answer set programming, Artificial Intelligence 175 (1) (2011) 278–298.

[19] M. Gelfond, Y. Zhang, Vicious circle principle and logic programs with aggregates, Theory and Practice of Logic Programming 14 (4-5) (2014) 587–601.

[20] M. Alviano, W. Faber, Stable model semantics of abstract dialectical frameworks revisited: A logic programming perspective, in: Q. Yang, M. Wooldridge (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, AAAI Press, 2015, pp. 2684–2690.
URL `http://ijcai.org/Abstract/15/380`

[21] M. Gebser, A. Harrison, R. Kaminski, V. Lifschitz, T. Schaub, Abstract Gringo, Theory and Practice of Logic Programming 15 (4-5) (2015) 449–463, available at `http://arxiv.org/abs/1507.06576`.

[22] P. Ferraris, Answer sets for propositional theories, in: C. Baral, G. Greco, N. Leone, G. Terracina (Eds.), Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05), Vol. 3662 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 2005, pp. 119–131.

[23] V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, ACM Transactions on Computational Logic 2 (4) (2001) 526–541.

doi:10.1145/502166.502170.
URL `http://doi.acm.org/10.1145/502166.502170`

[24] F. Aguado, P. Cabalar, D. Pearce, G. Pérez, C. Vidal, A denotational semantics for equilibrium logic, Theory and Practice of Logic Programming 15 (4-5) (2015) 620–634.

[25] M. Alviano, N. Leone, On the properties of GZ-aggregates in answer set programming, in: S. Kambhampati (Ed.), Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, IJCAI/AAAI Press, 2016, pp. 4105–4109.
URL `http://www.ijcai.org/Abstract/16/611`

[26] A. Harrison, V. Lifschitz, F. Yang, The semantics of gringo and infinitary propositional formulas, in: C. Baral, G. De Giacomo, T. Eiter (Eds.), Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'14), AAAI Press, 2014.
URL `http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7966`

[27] P. Cabalar, Functional answer set programming, Theory and Practice of Logic Programming 11 (2-3) (2011) 203–233.

[28] P. Cabalar, J. Fandinno, L. F. del Cerro, D. Pearce, Functional ASP with intensional sets: Application to gelfond-zhang aggregates, Theory and Practice of Logic Programming 18 (3-4) (2018) 390–405.

Appendix A. Proofs

Proof of Proposition 4. First, note that $\varphi \equiv_{cl} \psi$ implies $\Gamma(\varphi) \equiv_{cl} \Gamma(\psi)$ due to substitution of equivalents in classical logic, regardless whether φ occurs positively or negatively in $\Gamma(\varphi)$. On the other hand, HT satisfies the Deduction Theorem, i.e., $\varphi \models \psi$ iff $\varphi \to \psi$ is valid and, from the latter, we can derive the following intuitionistic consequences:

$$\begin{array}{rcl}
(\varphi \wedge \gamma) & \to & (\psi \wedge \gamma) \\
(\varphi \vee \gamma) & \to & (\psi \vee \gamma) \\
(\gamma \to \varphi) & \to & (\gamma \to \psi) \\
(\psi \to \gamma) & \to & (\varphi \to \gamma)
\end{array}$$

which are also consequences in the intermediate logic of HT. As a result, if $\varphi \models \psi$, the above formulas hold and, together with $\varphi \equiv_{cl} \psi$, we can apply Proposition 3 to conclude:

$$\begin{array}{rcl} \mathit{SM}(\varphi \wedge \gamma) & \supseteq & \mathit{SM}(\psi \wedge \gamma) \\ \mathit{SM}(\varphi \vee \gamma) & \supseteq & \mathit{SM}(\psi \vee \gamma) \\ \mathit{SM}(\gamma \rightarrow \varphi) & \supseteq & \mathit{SM}(\gamma \rightarrow \psi) \\ \mathit{SM}(\varphi \rightarrow \gamma) & \subseteq & \mathit{SM}(\psi \rightarrow \gamma) \end{array}$$

Finally, we can apply these relations by bottom-up structural induction on $\Gamma(\varphi)$ having in mind that, as we can see, each time we work with a subformula in an implication antecedent, the inclusion relation is reversed. If this happens an even number of times, the final effect is canceled, and so $\mathit{SM}(\Gamma(\varphi)) \supseteq \mathit{SM}(\Gamma(\psi))$, since we started with $\mathit{SM}(\varphi) \supseteq \mathit{SM}(\psi)$. Otherwise, visiting an odd number of implication antecedents, we get $\mathit{SM}(\Gamma(\varphi)) \subseteq \mathit{SM}(\Gamma(\psi))$. □

Before proving Proposition 6, we introduce first some auxiliary lemmas.

Lemma 1. *Let H and T be two classical interpretations such that $H \subseteq T$ and φ be a formula. Then, $H \models_{cl} \varphi^T$ implies $T \models_{cl} \varphi$.*

Proof. The proof is done by structural induction assuming the statement holds for every subformula of φ. Note that $H \models_{cl} \varphi^T$ implies that the case $\varphi^T = \bot$ is always disregarded.
Case 1: $\varphi = a$ ground atom. Then

$$H \models_{cl} a^T \text{ iff } (H \models_{cl} a \text{ and } a \in T)\text{, and so, } T \models_{cl} a$$

Case 2: $\varphi = A = f\{\vec{X} : \mathit{cond}(\vec{X})\} \prec n$ a GZ-aggregate. Then

$$H \models_{cl} A^T \text{ iff } \left(H \models_{cl} \left(\bigwedge \mathtt{Gr}^{+}_{T}(\mathit{cond}(\vec{X}))\right)^T \text{ with } T \models_{cl} A\right)\text{. Thus, } T \models_{cl} A$$

Case 3: $\varphi = B = f\{\vec{c}_1 : \varphi_1, \ldots, \vec{c}_m : \varphi_m\} \prec n$ an F-aggregate. Then

$$H \models_{cl} B^T \text{ iff } (H \models_{cl} f\{\vec{c}_1 : \varphi_1^T, \ldots, \vec{c}_m : \varphi_m^T\} \prec n \text{ and } T \models_{cl} B).$$

and so, $T \models_{cl} B$.
Case 4: In case that $\varphi = (\varphi_1 \otimes \varphi_2)$. By definition with $\otimes \in \{\wedge, \vee, \rightarrow\}$,

$$\begin{array}{rl} H \models_{cl} \varphi^T & \text{iff } H \models_{cl} (\varphi_1 \otimes \varphi_2)^T \\ & \text{iff } H \models_{cl} (\varphi_1^T \otimes \varphi_2^T) \text{ for } T \models_{cl} (\varphi_1 \otimes \varphi_2) \\ & \text{which implies } T \models_{cl} \varphi \end{array}$$

□

Lemma 2. *Any set of atoms T and formula φ satisfy: $T \models_{cl} \varphi^T$ iff $T \models_{cl} \varphi$.*

Proof. From Lemma 1 with $H = T$, it immediately follows that $T \models_{cl} \varphi^T$ implies $T \models_{cl} \varphi$. We need to prove $T \models_{cl} \varphi$ implies $T \models_{cl} \varphi^T$.

Case 1: $\varphi = a$ ground atom. Then

$$T \models_{cl} a \text{ iff } a \in T \text{ implies } a^T = a \text{ implies } T \models_{cl} a^T$$

Case 2: $\varphi = A = f\{\vec{X} : cond(\vec{X})\} \prec n$ a GZ-aggregate. Then,

$$\begin{aligned} T \models_{cl} A \text{ implies } & \hat{f}_{|\vec{X}|}\big(\{\ \vec{c} \in \mathcal{C}^{|\vec{X}|} \ \mid\ T \models_{cl} cond(\vec{c})\ \}\big) \prec n \\ & \text{and } A^T = \big(\bigwedge \mathtt{Gr}^+_T(cond(\vec{X}))\big)^T \end{aligned}$$

Furthermore, by induction hypothesis, $T \models_{cl} cond(\vec{c})$ implies that $T \models_{cl} cond(\vec{c})^T$ for all $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ and, thus,

$$\begin{aligned} T \models_{cl} \big(\bigwedge \{\ cond(\vec{c})^T \ \mid\ \vec{c} \in \mathcal{C}^{|\vec{X}|} \text{ and } T \models_{cl} cond(\vec{c})\ \}\big) \\ \text{implies } T \models_{cl} \big(\bigwedge \{\ cond(\vec{c}) \ \mid\ \vec{c} \in \mathcal{C}^{|\vec{X}|} \text{ and } T \models_{cl} cond(\vec{c})\ \}\big)^T \\ \text{implies } T \models_{cl} \big(\bigwedge \{\ \psi \in \{\ cond(\vec{c}) \ \mid\ \vec{c} \in \mathcal{C}^{|\vec{X}|}\ \} \ \mid\ T \models_{cl} \psi\ \}\big)^T \\ \text{implies } T \models_{cl} \big(\bigwedge \{\ \psi \in \mathtt{Gr}(cond(\vec{X})) \ \mid\ T \models_{cl} \psi\ \}\big)^T \\ \text{implies } T \models_{cl} \big(\bigwedge \mathtt{Gr}^+_T(cond(\vec{X}))\big)^T \\ \text{implies } T \models_{cl} A^T \end{aligned}$$

Case 3: $\varphi = B = f\{\vec{c}_1 : \varphi_1, \dots, \vec{c}_m : \varphi_m\} \prec n$ an F-aggregate. Then

$$\begin{aligned} T \models_{cl} B \text{ implies } & B^T = f\{\vec{c}_1 : \varphi_1^T, \dots, \vec{c}_m : \varphi_m^T\} \prec n \text{ and } \hat{f}_{|\vec{c}|}\big(\{\ \vec{c}_i \ \mid\ T \models_{cl} \varphi_i\ \}\big) \prec n \\ \text{implies } & T \models_{cl} \hat{f}_{|\vec{c}|}\big(\{\ \vec{c}_i \ \mid\ T \models_{cl} \varphi_i^T\ \}\big) \prec n \\ \text{implies } & T \models_{cl} f\{\vec{c}_1 : \varphi_1^T, \dots, \vec{c}_m : \varphi_m^T\} \prec n \\ \text{implies } & T \models_{cl} B^T \end{aligned}$$

Case 4: In case that $\varphi = (\varphi_1 \otimes \varphi_2)$ with $\otimes \in \{\wedge, \vee\}$. By induction, it follows

$$\begin{aligned} T \models_{cl} (\varphi_1 \otimes \varphi_2) \text{ implies } & (\varphi_1 \otimes \varphi_2)^T = (\varphi_1^T \otimes \varphi_2^T) \text{ and } T \models_{cl} (\varphi_1^T \otimes \varphi_2^T) \\ \text{implies } & T \models_{cl} (\varphi_1 \otimes \varphi_2)^T \\ \text{implies } & T \models_{cl} \varphi^T \end{aligned}$$

Case 5: In case that $\varphi = (\varphi_1 \rightarrow \varphi_2)$. By induction it follows

$$\begin{aligned}
T \models_{cl} (\varphi_1 \rightarrow \varphi_2) &\text{ implies } \varphi^T = (\varphi_1^T \rightarrow \varphi_2^T) \text{ and either } T \not\models_{cl} \varphi_1 \text{ or } T \models_{cl} \varphi_2 \\
&\text{ implies } \varphi^T = (\varphi_1^T \rightarrow \varphi_2^T) \text{ and either } T \not\models_{cl} \varphi_1^T \text{ or } T \models_{cl} \varphi_2^T \\
&\text{ implies } \varphi^T = (\varphi_1^T \rightarrow \varphi_2^T) \text{ and } T \models_{cl} \varphi_1^T \rightarrow \varphi_2^T \\
&\text{ implies } T \models_{cl} \varphi^T \qquad \square
\end{aligned}$$

For formulas φ and ψ, let us write $\varphi \Leftrightarrow \psi$ when the following condition hold: $H \models_{cl} \varphi^T$ iff $H \models_{cl} \psi^T$ for all interpretations $H \subseteq T$.

Lemma 3. *Let ψ and ψ' be a pair of formulas whose unique difference is that some subformula φ of ψ is replaced by another subformula φ' in ψ'. If $\varphi \Leftrightarrow \varphi'$ then $\psi \Leftrightarrow \psi'$.*

Proof. The proof follows by structural induction assuming the statement holds for all subformulas of ψ.
Case 1: $\psi = a$ ground atom, then the only (sub-)formula of ψ is ψ itself. Thus either both $\psi = \varphi$ and $\psi' = \varphi'$ or $\psi = \psi'$. In both cases $\psi \Leftrightarrow \psi'$.

In the following holds, if $\psi = \varphi$ and $\psi' = \varphi'$ then the assertion is trivial.
Case 2: $\psi = f\{\vec{X} : cond(\vec{X})\} \prec n$ is a GZ-aggregate.
Then, $\psi' = f\{\vec{X} : cond'(\vec{X})\} \prec n$ and, by induction hypothesis $cond(X) \Leftrightarrow cond'(X)$ holds. Furthermore, from Lemma 2, this implies that $cond(\vec{c}) \equiv_{cl} cond'(\vec{c})$. Then,

$$\begin{aligned}
H \models_{cl} \psi^T &\text{ iff } H \models_{cl} \bigwedge\{\ cond(\vec{c})^T \ \mid\ T \models_{cl} cond(\vec{c}) \text{ with } \vec{c} \in \mathcal{C}^{\vec{X}}\ \} \text{ and } T \models_{cl} \psi \\
&\text{ iff } H \models_{cl} \bigwedge\{\ cond'(\vec{c})^T \ \mid\ T \models_{cl} cond'(\vec{c}) \text{ with } \vec{c} \in \mathcal{C}^{\vec{X}}\ \} \text{ and } T \models_{cl} \psi' \\
&\text{ iff } H \models_{cl} (\psi')^T
\end{aligned}$$

That is, $\psi \Leftrightarrow \psi'$.
Case 3: $\psi = f\{\vec{c}_1 : \varphi_1, \dots, \vec{c}_m : \varphi_m\} \prec n$ an F-aggregate.
Then, $\psi' = f\{\vec{c}_1 : \varphi'_1, \dots, \vec{c}_m : \varphi'_m\} \prec n$ with $\varphi_i \Leftrightarrow \varphi'_i$ for all i by induction hypothesis. Then,

$$\begin{aligned}
H \models_{cl} \psi^T &\text{ iff } H \models_{cl} f\{\vec{c}_1 : \varphi_1^T, \dots, \vec{c}_m : \varphi_m^T\} \prec n \text{ and } T \models_{cl} \psi \\
&\text{ iff } H \models_{cl} f\{\vec{c}_1 : (\varphi'_1)^T, \dots, \vec{c}_m : (\varphi'_m)^T\} \prec n \text{ and } T \models_{cl} \psi' \\
&\text{ iff } H \models_{cl} (\psi')^T
\end{aligned}$$

That is, $\psi \Leftrightarrow \psi'$.

Case 4: If $\psi = \varphi_1 \otimes \varphi_2$ for $\otimes \in \{\wedge, \vee, \rightarrow\}$, then $\psi' = \varphi_1' \otimes \varphi_2'$ with $\varphi_i' \Leftrightarrow \varphi_i'$ holds, by induction hypothesis. Then,

$$\begin{aligned} H \models_{cl} \psi^T \text{ iff } & H \models_{cl} \varphi_1^T \otimes \varphi_2^T \text{ and } T \models_{cl} \psi \\ \text{iff } & H \models_{cl} (\varphi_1')^T \otimes (\varphi_2')^T \text{ and } T \models_{cl} \psi' \\ \text{iff } & H \models_{cl} (\psi')^T \end{aligned}$$

That is, $\psi \Leftrightarrow \psi'$. □

Proof of Proposition 6. The proof follows directly from the above auxiliary Lemmas: i) from Lemma 1, ii) from Lemma 2 and iii) from Lemma 3. In particular, for iii), note that, if we take a theory $\Gamma(\varphi)$ with a distinguished occurrence of φ, we can separate the formula $\gamma \in \Gamma(\varphi)$ containing that particular occurrence of φ and define all the rest as $\Gamma' = \Gamma(\varphi)\backslash\{\gamma\}$. Then, $\Gamma(\psi) = \Gamma' \cup \{\gamma'\}$ where γ' is obtained from γ by replacing some occurrence of φ by ψ and, from Lemma 3, and the fact $H \models_{cl} \varphi^T$ iff $H \models_{cl} \psi^T$ for all interpretations $H \subseteq T$ it follows that that $H \models_{cl} \gamma^T$ iff $H \models_{cl} (\gamma')^T$. As a result, it is clear that, for any context theory Δ, $H \models_{cl} (\Gamma(\varphi) \cup \Delta)^T$ iff $H \models_{cl} (\Gamma(\psi) \cup \Delta)^T$ for all $H \subseteq T$ which, by definition, implies that $SM(\Gamma(\varphi) \cup \Delta) = SM(\Gamma(\psi) \cup \Delta)$ and $\varphi \equiv_s \psi$. □

Proof of Proposition 7. From Proposition 12 in [17], it follows that $B^T \equiv_{cl} \Phi[B]^T$ for any interpretation T. Thus, the result follows directly from Proposition 6. □

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 vacuously.

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) $H \models_{cl} \quad cond(\vec{c})^T$ for every $\vec{c} \in \mathcal{C}^{\vec{X}}$ s.t. $T \models_{cl} cond(\vec{c})$, and

(b) $H \models_{cl} (\neg cond(\vec{c}))^T$ for every $\vec{c} \in \mathcal{C}^{\vec{X}}$ s.t. $T \not\models_{cl} cond(\vec{c})$

Furthermore, by definition, $T \models_{cl} A$ implies

$$A^T \quad = \quad \bigwedge\{ \ cond(\vec{c})^T \ \mid \ T \models_{cl} cond(\vec{c}) \text{ with } \vec{c} \in \mathcal{C}^{\vec{X}} \ \} \tag{A.1}$$

and, thus, $H \models_{cl} A^T$ implies that (a) holds. Moreover, $T \not\models_{cl} cond(\vec{c})$ implies $cond(\vec{c})^T = \bot$ which, in its turn, implies $H \models_{cl} (\neg cond(\vec{c}))^T$ 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) $H \models_{cl} \quad cond(\vec{c})^T$ for every $\vec{c} \in \mathcal{C}^{\vec{X}}$ such that $I \models_{cl} cond(\vec{c})$, and

(d) $H \models_{cl} (\neg cond(\vec{c}))^T$ for every $\vec{c} \in \mathcal{C}^{\vec{X}}$ such that $I \not\models_{cl} cond(\vec{c})$

From Proposition 6 and the fact that $H \subseteq T$, it follows that $H \models_{cl} cond(\vec{c})^T$ implies that $T \models_{cl} cond(\vec{c})$ and, thus, (c) implies

(c') $T \models_{cl} cond(\vec{c})$ for every $\vec{c} \in \mathcal{C}^{\vec{X}}$ s.t. $I \models_{cl} cond(\vec{c})$

Similarly, $H \models_{cl} (\neg cond(\vec{c}))^T$ implies $T \models_{cl} (\neg cond(\vec{c}))$ which, in its turn, implies that $T \not\models_{cl} cond(\vec{c})$. Thus, (d) implies

(d') $T \not\models_{cl} cond(\vec{c})$ for every $\vec{c} \in \mathcal{C}^{\vec{X}}$ such that $I \not\models_{cl} cond(\vec{c})$

From (c') and (d'), it follows that

(e) $T \models_{cl} cond(\vec{c})$ iff $I \models_{cl} cond(\vec{c})$ holds for every $\vec{c} \in \mathcal{C}^{\vec{X}}$

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 (A.1) 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 $H \models_{cl} cond(\vec{c})^T$ for every $\vec{c} \in \mathcal{C}^{\vec{X}}$ such that $T \models cond(\vec{c})$ which follows from (c) and (e). □

For the following proofs, we need the following notation: given any formula φ, by $[\![\, \varphi \,]\!] \overset{\text{def}}{=} \{ \ I \ \mid \ I \models_{cl} \varphi \ \}$ we denote the set of all propositional interpretation that satisfy φ.

Proof of Proposition 10. Suppose, for the sake of contradiction, that there is some HT-interpretations such that $\langle H, T\rangle \models \Phi[A]$, but $\langle H, T\rangle \not\models \Phi[\mathtt{F}[A]]$. Note that $\langle H, T\rangle \models \Phi[A]$ implies that there is some $I \in [\![\, A \,]\!]$ satisfying the following two conditions:

(a) $\langle H, T\rangle \models \quad cond(\vec{c})$ for all $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ s.t. $I \models cond(\vec{c})$,

(b) $\langle H, T\rangle \models \neg \mathit{cond}(\vec{c})$ for all $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ s.t. $I \not\models \mathit{cond}(\vec{c})$

Furthermore, $\langle H, T\rangle \models \mathit{cond}(\vec{c})$ and $\langle H, T\rangle \models \neg \mathit{cond}(\vec{c})$ respectively imply that $T \models \mathit{cond}(\vec{c})$ and $T \not\models \mathit{cond}(\vec{c})$. In its turn, this implies that $T \models \mathit{cond}(\vec{c})$ iff $I \models \mathit{cond}(\vec{c})$ for all $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ which, since $I \in [\![A]\!]$ holds, implies that $T \in [\![A]\!]$ holds, too. Note that, since $T \in [\![A]\!]$, every $J \notin [\![A]\!]$ must satisfy one of the following condition hold:

(c) $T \not\models \mathit{cond}(\vec{c})$ and $J \models \mathit{cond}(\vec{c})$ for some $\vec{c} \in \mathcal{C}^{|\vec{X}|}$, or

(d) $T \models \mathit{cond}(\vec{c})$ and $J \not\models \mathit{cond}(\vec{c})$ for some $\vec{c} \in \mathcal{C}^{|\vec{X}|}$

Furthermore, $\langle H, T\rangle \not\models \Phi[\mathtt{F}[A]]$ implies that there is some $J \notin [\![A]\!]$ such that $\langle H, T\rangle \not\models \psi$ with $\psi = (\psi_1 \to \psi_2)$ and

$$\psi_1 \stackrel{\text{def}}{=} \Big(\bigwedge \mathtt{Gr}^+_J(A)\Big) \qquad\qquad \psi_2 \stackrel{\text{def}}{=} \Big(\bigvee \mathtt{Gr}^-_J(A)\Big)$$

Hence, either $T \not\models \psi$ or both $\langle H, T\rangle \models \psi_1$ and $\langle H, T\rangle \not\models \psi_2$. If we assume that $T \not\models \psi$, there must be some $J \notin [\![A]\!]$ that satisfies the following conditions:

(e) $T \models \mathit{cond}(\vec{c})$ for all $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ s.t. $J \models \mathit{cond}(\vec{c})$,

(f) $T \not\models \mathit{cond}(\vec{c})$ for all $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ s.t. $J \not\models \mathit{cond}(\vec{c})$

plus (c) and (d), which is a contradiction. Consequently, it must be that both $\langle H, T\rangle \models \psi_1$ and $\langle H, T\rangle \not\models \psi_2$, but this implies that that the following condition hold:

(g) $\langle H, T\rangle \models \mathit{cond}(\vec{c})$ for all $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ s.t. $J \models \mathit{cond}(\vec{c})$,

(h) $\langle H, T\rangle \not\models \mathit{cond}(\vec{c})$ for all $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ s.t. $J \not\models \mathit{cond}(\vec{c})$

Then, since $T \in [\![A]\!]$ and $J \notin [\![A]\!]$, one of the following condition must hold

(i) $I \models \mathit{cond}(\vec{c})$ and $J \not\models \mathit{cond}(\vec{c})$ for some $\vec{c} \in \mathcal{C}^{|\vec{X}|}$

(j) $I \not\models \mathit{cond}(\vec{c})$ and $J \models \mathit{cond}(\vec{c})$ for some $\vec{c} \in \mathcal{C}^{|\vec{X}|}$

If the former, (a) and (h), respectively imply that both, $\langle H, T\rangle \models \mathit{cond}(\vec{c})$ and $\langle H, T\rangle \not\models \mathit{cond}(\vec{c})$, must hold, which is a contradiction. Otherwise, the latter plus (b) and (g) respectively imply that $\langle H, T\rangle \models \neg\mathit{cond}(\vec{c})$ and $\langle H, T\rangle \models \mathit{cond}(\vec{c})$ hold, which is also a contradiction. □

For proving Proposition 11, we need to introduce the following notation and some auxiliary results. Let $\prec$ denote any relation symbol. We say that $A = (f\{\vec{X} : \mathit{cond}(\vec{X})\} \prec n)$ is *monotone* (resp. *antimonotone*) iff $\hat{f}(W_1) \prec n$ implies $\hat{f}(W_2) \prec n$ for all sets of tuples $W_1 \subseteq W_2 \subseteq \mathcal{C}^{|\vec{X}|}$ (resp. $W_2 \subseteq W_1 \subseteq \mathcal{C}^{|\vec{X}|}$).

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 $f^{\geq}\{\vec{X} : \mathit{cond}(\vec{X})\} \geq n$ "testing" the greater or equal relation, that is, its function $\hat{f}^{\geq}$ is defined so that $\hat{f}^{\geq}(W) \geq n$ iff there is some $W' \supseteq W$ such that $\hat{f}(W') \prec n$ Analogously, $A^{\leq}$ stands for $f^{\leq}\{\vec{X} : \mathit{cond}(\vec{X})\} \leq n$ whose function $\hat{f}^{\leq}$ is defined so that $\hat{f}^{\leq}(W) \leq n$ iff there is $W' \subseteq W$ such that $\hat{f}(W) \prec n$. Then, by $\Phi^{\mathtt{D}}[A]$, we denote the following formula:

$$\bigwedge_{T \notin [\![A^{\leq}]\!]} \Big(\neg \bigwedge \mathtt{Gr}^{+}_{T}(\mathit{cond}(\vec{X}))\Big) \wedge \bigwedge_{T \notin [\![A^{\geq}]\!]} \Big(\bigvee \mathtt{Gr}^{-}_{T}(\mathit{cond}(\vec{X}))\ \Big) \tag{A.2}$$

Lemma 4. *Let A be a regular aggregate. Then, $[\![A]\!] = [\![A^{\geq}]\!] \cap [\![A^{\leq}]\!]$.* □

Proof. By definition, it is clear that $[\![A]\!] \subseteq [\![A^{\geq}]\!] \cap [\![A^{\leq}]\!]$. The other way around. Let $I \in [\![A^{\geq}]\!] \cap [\![A^{\leq}]\!]$. Then, there are $W_1 \subseteq W_I \subseteq W_2$ such that $\hat{f}(W_1) \prec n$ and $\hat{f}(W_2) \prec n$. Furthermore, since A is regular, if either $W_1 \subset W_I$ or $W_I \subset W_2$ hold, then either $\hat{f}(W_1) \not\prec n$ and $\hat{f}(W_2) \not\prec n$ must hold too. Therefore, $W_1 = W_I = W_2$ and $I \in [\![A]\!]$ hold. □

Lemma 5. *Any regular F-aggregate B with condition φ satisfies:*

i) aggregates $B^{\geq}$ and $B^{\leq}$ are respectively monotone and antimonotone,

ii) $T \models_{cl} B$ iff $T \models_{cl} B^{\geq} \wedge B^{\leq}$

iii) $\Phi[B] \ \equiv_s\ \Phi[B^{\geq}] \wedge \Phi[B^{\leq}] \ \equiv_s\ \Phi^{\mathtt{D}}[B]$.

Proof. Conditions i) and ii) directly follows from definition and Lemma 4, respectively. Then, since $B^{\geq}$ is monotone and $B^{\leq}$ is antimonotone, from Proposition 13 in [17], it follows that

$$\Phi[B^{\geq}] \ \equiv_s \bigwedge_{I \notin [\![B^{\geq}]\!]} \Big(\bigvee \mathtt{Gr}^{-}_{I}(\varphi)\Big) \quad \Phi[B^{\leq}] \ \equiv_s \bigwedge_{I \notin [\![B^{\leq}]\!]} \Big(\neg \bigwedge \mathtt{Gr}^{+}_{I}(\varphi)\Big)$$

Furthermore, since $[\![B]\!] = [\![B^{\leq}]\!] \cap [\![B^{\geq}]\!]$, it follows that $I \notin [\![A]\!]$ iff either $I \notin [\![B^{\geq}]\!]$ or $I \notin [\![B^{\leq}]\!]$. Hence, we have that $\Phi[B^{\geq}] \wedge \Phi[B^{\leq}] \ \equiv_s\ \Phi^{\mathtt{D}}[B]$

Moreover, we have that

$$\begin{aligned}
\Phi[B] &= \bigwedge_{I \notin [\![B]\!]} \Big(\big(\bigwedge \mathtt{Gr}^+_I(\varphi) \big) \to \big(\bigvee \mathtt{Gr}^-_I(\varphi) \big) \Big) \\
&\equiv \bigwedge_{I \notin [\![B^{\geq}]\!]} \Big(\big(\bigwedge \mathtt{Gr}^+_I(\varphi) \big) \to \big(\bigvee \mathtt{Gr}^-_I(\varphi) \big) \Big) \\
&\wedge \bigwedge_{I \notin [\![B^{\leq}]\!]} \Big(\big(\bigwedge \mathtt{Gr}^+_I(\varphi) \big) \to \big(\bigvee \mathtt{Gr}^-_I(\varphi) \big) \Big) \\
&= \Phi[B^{\geq}] \wedge \Phi[B^{\leq}]
\end{aligned}$$

Consequently, $\Phi[B] \equiv_s \Phi[B^{\geq}] \wedge \Phi[B^{\leq}] \equiv_s \Phi^{\mathtt{D}}[B]$ holds. $\square$

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

Lemma 6. *Any GZ-aggregate $A = f\{\vec{X} : \mathit{cond}(\vec{X})\} \prec c$ with propositional $\mathit{cond}(\vec{X})$ satisfies that the following three conditions are equivalent*

i) $\langle H,T\rangle \models \Phi[\mathtt{F}[A]]$

ii) $H \models_{cl} \mathtt{F}[A]^T$

iii) $\hat{f}(W_{\langle H,T\rangle}(A)) \prec n$ *and* $\hat{f}(W_T(A)) \prec n$. $\square$

Proof. Let us first prove that condition ii) holds iff iii) holds. In case that $T \not\models_{cl} A$, it follows that $A^T = \bot$ and $T \not\models_{cl} \Phi[A]$ which, in their turn, respectively imply that $H \not\models_{cl} \mathtt{F}[A]^T$ and $\hat{f}(W_T(A)) \not\prec n$. Hence, we may assume without loss of generality that $T \models_{cl} A$ and, thus, that $\hat{f}(W_T(A)) \prec n$ holds.

$$\begin{aligned}
H \models_{cl} \mathtt{F}[A]^T \quad &\text{iff} \quad H \models_{cl} f\{\vec{c}_1 : \varphi_1^T, \ldots, \vec{c}_m : \varphi_m^T\} \prec n \\
&\text{iff} \quad \hat{f}\{\ \vec{c}_i\ \mid\ H \models_{cl} \varphi_i^T\ \} \prec n \\
&\text{iff} \quad \hat{f}(\{\ \vec{c}_i\ \mid\ \langle H,T\rangle \models_{cl} \varphi_i\ \}) \prec n \\
&\text{iff} \quad \hat{f}(W_{\langle H,T\rangle}) \prec n
\end{aligned}$$

Recall that, by Proposition 3 in [17], it follows that any propositional formula φ satisfies: $H \models_{cl} \varphi^T$ iff $\langle H,T\rangle \models_{cl} \varphi$. This also implies that conditions i) and ii) are equivalent and, thus, the statement holds. $\square$

Lemma 7. *Any regular GZ-aggregate A satisfies:* $\Phi[\mathtt{F}[A]] \models \Phi[A]$. $\square$

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

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

On the one hand, $\langle H,T\rangle \not\models \neg cond(\vec{c})$ implies $T \not\models_{cl} \neg cond(\vec{c})$ which, in its turn, implies $T \models_{cl} cond(\vec{c})$. 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 $\langle H,T\rangle \not\models \Phi[\mathtt{F}[A]]$ (Lemma 6) which is a contradiction with the assumption $\langle H,T\rangle \models \Phi[\mathtt{F}[A]]$. If the latter, we have that $T \not\models_{cl} A$ and $A \equiv_{cl} \Phi[A]$ and $T \not\models_{cl} \Phi[A]$, 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, Lemma 5, implies:

(c) either $T \not\models A^{\geq}$ or $T \not\models A^{\leq}$, and

(d) $\langle H,T\rangle \models \Phi[\mathtt{F}[A]]$ and $\Phi[\mathtt{F}[A]] \equiv_s \Phi^{\mathtt{D}}[\mathtt{F}[A]]$ and, thus, $\langle H,T\rangle \models \Phi^{\mathtt{D}}[\mathtt{F}[A]]$.

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

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

(f) $\langle H,T\rangle \models \quad cond(\vec{c})$ for some $\vec{c} \in \mathcal{C}^{|\vec{X}|}$ s.t. $T \not\models cond(\vec{c})$ (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. If A is regular, the from Proposition 10 and Lemma 7, it respectively follows $\Phi[A] \models \Phi[\mathtt{F}[A]]$ and $\Phi[\mathtt{F}[A]] \models \Phi[A]$. Thus, $\Phi[A] \equiv_s \Phi[\mathtt{F}[A]]$. Hence, it only remains to be prove that A is regular in these cases. For that, note that $\widehat{\mathtt{count}}(W_1) < \widehat{\mathtt{count}}(W_2)$ for all $W_1 \subset W_2$ and, thus, $\mathtt{count}\{\vec{X} : cond(\vec{X})\} = n$ 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. $\widehat{\mathtt{sum}}(W_1) > \widehat{\mathtt{sum}}(W_2)$), so $A = (\mathtt{sum}\{X,\vec{Y} : cond(X,\vec{Y}) \wedge X > 0\} = n)$ (resp. $A = (\mathtt{sum}\{X,\vec{Y} : cond(X,\vec{Y}) \wedge X < 0\} = n)$) is also regular. Therefore, we conclude that $\Phi[A] \equiv_s \Phi[\mathtt{F}[A]]$ holds. Finally, note that we have $A \equiv_s \Phi[A]$ (Proposition 8) and $\mathtt{F}[A] \equiv_s \Phi[\mathtt{F}[A]]$ (Proposition 7) and, thus, we obtain that the equivalence $A \equiv_s \mathtt{F}[A]$ holds. □

Proof of Proposition 12. Since Π is closed, the set of all instances of any rule r coincide with $\mathtt{Gr}(r)$. Hence, for any ground rule $r' \in \mathtt{Gr}(\Pi)$ there is a rule $r'' \in \mathtt{Gr}_{\mathtt{gringo}}(\Pi)$ which is the result of replacing every aggregate atom A by $\mathtt{F}[A]$, and vice-versa. From Proposition 11, we have that $r' \equiv_s r''$ and, thus, that the stable models of $\mathtt{Gr}(\Pi)$ and $\mathtt{Gr}_{\mathtt{gringo}}(\Pi)$ coincide. $\square$

Lemma 8. *Any GZ-aggregate $A = (f\{\vec{X} : cond(\vec{X})\} \prec n)$ satisfies:*

$$A \;\equiv_s\; \Big(\bigvee_{m \in \mathbb{Z}} (f\{\vec{X} : cond(\vec{X})\} = m) \wedge (m \prec n) \Big)$$

Proof. Let φ the right hand side of the above equivalence. Then, assume that $T \not\models_{cl} A$. This implies that $A^T = \bot$ and, it is clear that $H \not\models_{cl} A^T$ for all set of atoms $H \subseteq T$. Furthermore, $T \not\models_{cl} A$ implies that:

$$\hat{f}_{|\vec{X}|}\big(\{\ \vec{c} \in \mathcal{C}^{|\vec{X}|} \ \mid\ T \models_{cl} cond(\vec{c})\ \}\big) = k$$

with $k \in \mathbb{Z}$, $k \not\prec n$ and the usual meaning of $\prec$. Hence, for all $m \in \mathbb{Z}$, it follows that either $T \not\models_{cl} (f\{\vec{X} : cond(\vec{X})\} = m)$ (case that $m \neq k$) or $T \not\models_{cl} (m \prec n)$ (case that $m = k$). Hence, $T \not\models_{cl} \varphi$ and, thus, it follows that $\varphi^T = \bot$ and $H \not\models_{cl} \varphi^T$ for any set of atoms H. That is, for all set of atoms $H \subseteq T$, we have that both $H \not\models_{cl} A^T$ and $H \not\models_{cl} \varphi^T$ hold. From condition iii) in Proposition 6, this implies that $A \equiv_s \varphi$.

Thus, we may assume without loss of generality that $T \models_{cl} A$. That is, that we have that $\hat{f}_{|\vec{X}|}\big(\{\ \vec{c} \in \mathcal{C}^{|\vec{X}|} \ \mid\ T \models_{cl} cond(\vec{c})\ \}\big) = k$ with $k \in \mathbb{Z}$ and $k \prec n$ and the usual meaning of the relation $\prec$. Furthermore, this implies that $T \models_{cl} \varphi$ and, thus, that

$$\begin{aligned}
\varphi^T \;&\equiv_{cl}\; \bigvee_{m \in \mathbb{Z}} (f\{\vec{X} : cond(\vec{X})\} = m)^T \wedge (m \prec n)^T \\
&\equiv_{cl}\; \bigvee_{m \prec n} (f\{\vec{X} : cond(\vec{X})\} = m)^T \\
&\equiv_{cl}\; \bigvee_{\substack{m \prec n \\ m \neq k}} (f\{\vec{X} : cond(\vec{X})\} = m)^T \vee (f\{\vec{X} : cond(\vec{X})\} = k)^T \\
&\equiv_{cl}\; \bigvee_{\substack{m \prec n \\ m \neq k}} \bot \vee (f\{\vec{X} : cond(\vec{X})\} = k)^T \\
&\equiv_{cl}\; (f\{\vec{X} : cond(\vec{X})\} = k)^T
\end{aligned}$$

Since $T \models_{cl} A$ and $T \models_{cl} (f\{\vec{X}: cond(\vec{X})\} = k)^T$, it follows

$$A^T \;=\; \big(\, \bigwedge \mathtt{Gr}^{+}_{T}(cond(\vec{X}))\,\big)^T \;=\; (f\{\vec{X}: cond(\vec{X})\} = k)^T$$

Then, for all set $H \subseteq At$, it follows that:
$H \models_{cl} \varphi^T$ iff $H \models_{cl} (f\{\vec{X}: cond(\vec{X})\} = k)^T$ iff $H \models_{cl} A^T$. □

Proof of Proposition 13. First note that replacing bound variables does not change the semantics of a logic program with respect to $\mathcal{A}log$. Hence, we may assume without loss of generality that Π is a closed logic program and we will show that step *ii*) of Definition 7 does not change the semantics of the program either.

Let $r(\vec{Y})$ be some rule of the form

$$Head(\vec{Y}) \;\leftarrow\; Pos(\vec{Y}) \wedge Neg(\vec{Y}) \wedge Agg(\vec{Y})$$

with free variables $\vec{Y}$ and let $A = (f\{\vec{X} : cond(\vec{X},\vec{Y})\} \prec t)$ be some aggregate in Agg. Let Agg' be the conjunction of all aggregates in Agg but for A, let $A' = f\{\vec{X} : cond(\vec{X},\vec{Y})\} = Z \wedge Z \prec t$ with with Z a fresh variable not occurring any where else, and let $r'(\vec{Y}, Z)$ be the rule

$$Head(\vec{Y}) \;\leftarrow\; Pos(\vec{Y}) \wedge Neg(\vec{Y}) \wedge Agg'(\vec{Y}) \wedge A'(\vec{Y}, Z)$$

We will show that $\mathtt{Gr}(r(\vec{Y})) \equiv_s \mathtt{Gr}(r'(\vec{Y}, Z))$. First, we note that

$$\mathtt{Gr}(r(\vec{Y})) \;=\; \{r(\vec{c}) \mid \vec{c} \in \mathcal{C}^{|\vec{Y}|}\}$$

and

$$\mathtt{Gr}(r'(\vec{Y}, Z)) \;=\; \{r'(\vec{c}, d) \mid \vec{c} \in \mathcal{C}^{|\vec{Y}|} \text{ and } d \in \mathcal{C}\}$$

Let $S(r(\vec{c})) \stackrel{\text{def}}{=} \{\; r'(\vec{c}, d) \in \mathtt{Gr}(r'(\vec{Y}, Z)) \;\mid\; d \in \mathcal{C} \;\}$ be the set of ground rules of $r'(\vec{Y}, Z)$ corresponding to ground rule $r(\vec{c})$. We just need to show that $r(\vec{c}) \equiv_s \bigwedge S(r(\vec{c}))$. Note that all rules in $S(r(\vec{c}))$ only differ on $A'(\vec{c}, d)$, so we have that $\bigwedge S(r(\vec{c}))$ is strongly equivalent to[10].

$$Head(\vec{c}) \;\leftarrow\; Pos(\vec{c}) \wedge Neg(\vec{c}) \wedge Agg'(\vec{c}) \wedge \bigvee\{\; A'(\vec{c}, d) \;\mid\; d \in \mathcal{C} \;\} \quad \text{(A.3)}$$

[10]Follows by distribution of conjunction over disjunction and the intuitionistic equivalence $\varphi \leftarrow \psi \vee \gamma \equiv (\varphi \leftarrow \psi \wedge \varphi \leftarrow \gamma)$.

Futthermore, from Lemma 8, we have that A and $\bigvee\{\ A'(\vec{c},d) \ \mid\ d \in \mathcal{C}\ \}$ are strongly equivalent. Then, replacing $\bigvee\{\ A'(\vec{c},d) \ \mid\ d \in \mathcal{C}\ \}$ by A in (A.3), we have that

$$\bigwedge S(r(\vec{c})) \ \equiv_s \ \mathit{Head}(\vec{c}) \ \leftarrow \ \mathit{Pos}(\vec{c}) \wedge \mathit{Neg}(\vec{c}) \wedge \mathit{Agg}'(\vec{c}) \wedge A$$

Inductively applying this reasoning to all atoms in $\mathit{Agg}'(\vec{c})$, we obtain that $r(\vec{c}) \ \equiv_s \ \bigwedge S(r(\vec{c}))$ holds and, thus, $\mathtt{Gr}(r(\vec{Y})) \ \equiv_s \ \mathtt{Gr}(r'(\vec{Y}, Z))$ follow. $\square$