Functional ASP with Intensional Sets: Application to Gelfond-Zhang Aggregates*

PEDRO CABALAR
Department of Computer Science, University of Corunna, Corunna, Spain
(e-mail: cabalar@udc.es)

JORGE FANDINNO and LUIS FARIÑAS DEL CERRO
IRIT, Université de Toulouse, CNRS, Toulouse, France
(e-mails: {jorge.fandinno, luis}@irit.fr)

DAVID PEARCE
Universidad Politécnica de Madrid, Madrid, Spain
(e-mail: david.pearce@upm.es)

submitted 28 April 2018; accepted 11 May 2018

Abstract

In this paper, we propose a variant of Answer Set Programming (ASP) with evaluable functions that extends their application to sets of objects, something that allows a fully logical treatment of aggregates. Formally, we start from the syntax of First Order Logic with equality and the semantics of Quantified Equilibrium Logic with evaluable functions (QEL=). Then, we proceed to incorporate a new kind of logical term, intensional set (a construct commonly used to denote the set of objects characterised by a given formula), and to extend QEL= semantics for this new type of expression. In our extended approach, intensional sets can be arbitrarily used as predicate or function arguments or even nested inside other intensional sets, just as regular first-order logical terms. As a result, aggregates can be naturally formed by the application of some evaluable function (count, sum, maximum, etc) to a set of objects expressed as an intensional set. This approach has several advantages. First, while other semantics for aggregates depend on some syntactic transformation (either via a reduct or a formula translation), the QEL= interpretation treats them as regular evaluable functions, providing a compositional semantics and avoiding any kind of syntactic restriction. Second, aggregates can be explicitly defined now within the logical language by the simple addition of formulas that fix their meaning in terms of multiple applications of some (commutative and associative) binary operation. For instance, we can use recursive rules to define sum in terms of integer addition. Last, but not least, we prove that the semantics we obtain for aggregates coincides with the one defined by Gelfond and Zhang for the Alog language, when we restrict to that syntactic fragment.

KEYWORDS: Answer Set Programming, Equilibrium Logic, Partial Functions, Aggregates

* Partially supported by MINECO, Spain, grant TIC2017-84453-P, Xunta de Galicia, Spain (GPC ED431B 2016/035 and 2016-2019 ED431G/01, CITIC). The second author is funded by the Centre International de Mathématiques et d’Informatique de Toulouse (CIMI) through contract ANR-11-LABEX-0040-CIMI within the program ANR-11-IDEX-0002-02. The fourth author is supported by UPM project RP151046021.
Due to its extensive use for practical Knowledge Representation and Reasoning (KRR), the paradigm of Answer Set Programming (ASP; Baral 2003) has been continuously subject to multiple extensions of its input language and, frequently, its formal semantics. One of those possible extensions is the addition of evaluable functions (see Cabalar 2013 for a survey). This extension allows us, for instance, to replace the conjunction $\text{mother}(\text{cain}, X) \land \text{mother}(\text{abel}, X)$ by the equality $\text{mother}(\text{cain}) = \text{mother}(\text{abel})$ so that (i) $\text{mother}$ can be better captured as a function (a person has a unique mother) and (ii) it is not treated as a Herbrand function, since syntactically different terms may refer to the same object. Although several prototypes for functional ASP have been developed (Lin and Wang 2008; Cabalar 2011; Balduccini 2013; Bartholomew and Lee 2014), the use of evaluable functions has not been commonly adopted in the mainstream ASP solvers yet. Still, their logical definition can also be useful for other common ASP extensions, as happened with their application to constraint ASP (Cabalar et al. 2016).

Another ASP extension that can be examined under the functional viewpoint is the use of aggregates. An aggregate is the result of an operation on a set of values, such as their cardinality, their sum, their maximum/minimum value, etc. ASP introduces this feature via so-called aggregate atoms, that allow comparing the result of an aggregate with some fixed value. To put an example, suppose $p(X)$ means that Agatha Christie wrote book $X$. Then, adding the aggregate atom $\text{count}\{X : p(X)\} \geq n$ in a rule body checks that she wrote at least $n$ books. Defining the semantics for these atoms may become tricky, since it is easy to build self-referential rules like:

$$p(a) \leftarrow \text{count}\{X : p(X)\} \geq n.$$  

(1)

to express that Mrs. Christie also writes an autobiography $a$ if she writes at least $n$ books. Different alternative semantics have been proposed for ASP aggregates (Simons et al. 2002; Pelov et al. 2007; Son and Pontelli 2007; Ferraris 2011; Faber et al. 2011; Gelfond and Zhang 2014) but all of them have treated each aggregate atom as a whole, without providing a semantics for its individual components. A different, and perhaps more natural possibility, is to consider inequality as a standard predicate and interpret $\text{count}(S)$ as an evaluable function, whose argument $S$ happens to be a set.

In this paper, we propose an extension of ASP with evaluable functions that allows their application to sets of objects and the treatment of aggregates as functions. To this aim, we start from the first-order logic characterisation of ASP, Quantified Equilibrium Logic (QEL; Pearce and Valverde 2004) plus its extension to evaluable functions (QEL$^F$; Cabalar 2011). Then, we proceed to include a new type of logical term, intensional set, with the form $\{\vec{r}(\vec{x}) : \varphi(\vec{x})\}$ and the expected meaning, that is, the set of tuples $\vec{r}(\vec{c})$ for which the formula $\varphi(\vec{c})$ holds (having $\vec{c}$ and $\vec{x}$ the same arity). Intensional sets constitute a quite common mathematical notation and, in fact, have been already studied in the context of Prolog (Dovier et al. 1991) and Constraint Logic Programming (Dovier et al. 2003). In our case, we will treat them as regular, first-order logical terms, without syntactic restrictions, so they can be arbitrarily nested in other expressions. One interesting feature inherited from QEL$^F$ is that functions can be partial, so we can use them to represent that, say, $\text{mother}(\text{adam})$, $\text{mother}(\text{eve})$ or $\text{division}(3,0)$ are undefined, but also that the maximum value of an empty set $\text{max}(\emptyset)$ is undefined too. The new
extension allows now to define new aggregates within the logical language. It suffices to add formulas to fix their meaning in terms of multiple applications of some (commutative and associative) binary operation. For instance, we show how to define the sum aggregate using recursive rules in terms of integer addition. Finally, we are also able to prove that, when restricted to the the Alog language (Gelfond and Zhang 2014), there is a semantic, one-to-one correspondence.

The rest of the paper is organised as follows. In Section 2 we recall the basic definitions of Functional ASP under the QEL interpretation. Section 3 introduces intensional sets while Section 4 studies their use for aggregates. Section 5 focuses on the correspondence to (Gelfond and Zhang 2014) aggregates. Finally, Section 6 concludes the paper.

2 Background: Quantified Equilibrium Logic with Evaluable Functions

The definition of propositional Equilibrium Logic (Pearce 1996) relied on a selection criterion on models of the intermediate logic of Here-and-There (HT; Heyting 1930). The first order case (Pearce and Valverde 2004) followed similar steps, introducing a quantified version of HT, called SQHT that stands for Quantified HT with static domains1 and equality. In this section we describe the syntax and semantics of a variant of the latter, called SQHT (Cabalar 2011), for dealing with evaluable functions.

We begin by defining a first-order language by its signature, a tuple $\Sigma = \langle C, F, P \rangle$ of disjoint sets where $C$ and $F$ are sets of function names and $P$ a set of predicate names. We assume that each function (resp. predicate) name has the form $f/n$ where $f$ is the function (resp. predicate) symbol, and $n \geq 0$ is an integer denoting the number of arguments (or arity). Elements in $C$ will be called Herbrand functions (or constructors), whereas elements in $F$ will receive the name of evaluable functions (or operations). The sets $C_0$ (Herbrand constants) and $F_0$ (evaluable constants) respectively represent the elements of $C$ and $F$ with arity 0. We assume $C_0$ contains at least one element.

First-order formulas follow the syntax of classical predicate calculus with equality “=”. We assume that default negation $\neg \varphi$ is defined as $\varphi \rightarrow \bot$. We use letters $x, y, z$ and their capital versions to denote variables, $\tau$ to denote terms, and letters $c, d, e$ to denote ground terms. Tuples of variables, terms and ground terms are respectively represented by $\vec{x}, \vec{\tau}, \vec{c}$. By abuse of notation, when a tuple contains a single element, we write just $\tau$ instead of $\langle \tau \rangle$. When writing formulas, we assume that all free variables are implicitly universally quantified. An atom like $\tau = \tau'$ is called an equality atom, whereas an atom like $p(\tau_1, \ldots, \tau_n)$ for any predicate $p/n$ different from equality receives the name of predicate atom. Given any set of functions $S$ we write $\text{Terms}_F(S)$ to stand for the set of ground terms built from functions (and constants) in $S$. In particular, the set of all possible ground terms for the signature $\Sigma = \langle C, F, P \rangle$ would be $\text{Terms}_F(C \cup F)$ whereas the subset $\text{Terms}_F(C)$ will be called the Herbrand Universe of the language $L_F(\Sigma)$.

The Herbrand Base $\mathcal{HB}_F(C, P)$ is the set containing all atoms that can be formed with predicates in $P$ and terms in the Herbrand Universe, $\text{Terms}_F(C)$.

**Definition 1** (SQHT-assignment). An SQHT-assignment $\sigma$ for a signature $\langle C, F, P \rangle$ is a function $\sigma : \text{Terms}_F(C \cup F) \rightarrow \text{Terms}_F(C) \cup \{u\}$ that maps any ground term in the

1 The term static domain means that the universe is shared among all worlds in the Kripke frame.
language to some ground term in the Herbrand Universe or the special value \( u \notin \text{Terms}_F \) (standing for undefined). The function \( \sigma \) must satisfy:

(i) \( \sigma(c) \stackrel{\text{def}}{=} c \) for all \( c \in \text{Terms}_F(C) \).

(ii) \( \sigma(f(\tau_1, \ldots, \tau_n)) \stackrel{\text{def}}{=} \begin{cases} u & \text{if } \sigma(\tau_i) = u \text{ for some } i = 1 \ldots n \\ \sigma(f(\sigma(\tau_1), \ldots, \sigma(\tau_n))) & \text{otherwise} \end{cases} \)

As we can see, the value of any functional term is an element from the Herbrand Universe \( \text{Terms}_F(C) \), excluding the cases in which operations are left undefined (i.e., they are partial functions) – if so, they are assigned the special element \( u \) (outside the universe) instead. Condition (i) asserts, as expected, that any term \( c \) from the Herbrand Universe has the fixed valuation \( \sigma(c) = c \). Condition (ii) asserts that a functional term with an undefined argument becomes undefined in its turn (functions like these are called strict). Otherwise, if all arguments are defined, then functions preserve their interpretation through subterms – for instance, if we have \( \sigma(f(a)) = c \) we expect that \( \sigma(g(f(a))) \) and \( \sigma(g(c)) \) coincide. It is easy to see that (ii) implies that \( \sigma \) is completely determined by the mappings \( f(\bar{c}) = d \) where \( f \) is any operation, \( \bar{c} \) a tuple of elements from \( \text{Terms}_F(C) \), and \( d \) an element in the latter. We call these expressions ground functional facts.

**Definition 2** (Ordering \( \preceq \) among assignments). Given two assignments \( \sigma, \sigma' \) we define \( \sigma \preceq \sigma' \) as the condition: \( \sigma(\tau) = \sigma'(\tau) \) or \( \sigma(\tau) = u \), for all terms \( \tau \in \text{Terms}_F(C \cup F) \).

As usual, we write \( \sigma < \sigma' \) when \( \sigma \preceq \sigma' \) and \( \sigma \neq \sigma' \). The intuitive meaning of \( \sigma \preceq \sigma' \) is that both contain compatible information, but the former contains less information than the latter: any defined function in \( \sigma \) must preserve the same value in \( \sigma' \).

**Definition 3** (SQHT\(^{-}\)-interpretation). An SQHT\(^{-}\)-interpretation \( \mathcal{I} \) for a signature \( \Sigma = (C, F, P) \) is a quadruple \( \langle \sigma^h, \sigma^t, I^h, I^t \rangle \) where \( I^h \subseteq I^t \subseteq \mathcal{H}_B \) are two sets of ground atoms and \( \sigma^h \) and \( \sigma^t \) are two assignments satisfying \( \sigma^h \preceq \sigma^t \).

The superindices \( h, t \) represent two intuitionistic Kripke worlds (respectively standing for here and there) with a reflexive ordering relation satisfying \( h \preceq t \). Accordingly, world \( h \) contains less information than \( t \), as we can see in the conditions \( I^h \subseteq I^t \) and \( \sigma^h \preceq \sigma^t \).

We say that the interpretation \( \mathcal{I} \) is total\(^2\) when both worlds contain the same information, that is, \( I^h = I^t \) and \( \sigma^h = \sigma^t \), and we abbreviate it as the pair \( \langle \sigma^t, I^t \rangle \). Moreover, given \( \mathcal{I} = \langle \sigma^h, \sigma^t, I^h, I^t \rangle \) we define its corresponding total interpretation \( \mathcal{I}^t \) as \( \langle \sigma^t, I^t \rangle \) that is, the one in which all the uncertainty in world \( h \) is “filled” with the information in \( t \).

An interpretation \( \mathcal{I} = \langle \sigma^h, \sigma^t, I^h, I^t \rangle \) satisfies a formula \( \varphi \) at some world \( w \in \{ h, t \} \), written \( \mathcal{I}, w \models_x \varphi \), when any of the following conditions are satisfied:

i) \( \mathcal{I}, w \models_x p(\tau_1, \ldots, \tau_n) \text{ if } p(\sigma^w(\tau_1), \ldots, \sigma^w(\tau_n)) \in I^w \) for any predicate \( p/n \in P \);

ii) \( \mathcal{I}, w \models_x \tau_1 = \tau_2 \text{ if } \sigma^w(\tau_1) = \sigma^w(\tau_2) \neq u \);

iii) \( \mathcal{I}, w \not\models_x \bot; \mathcal{I}, w \models_x \top \);

iv) \( \mathcal{I}, w \models_x \alpha \land \beta \text{ if } \mathcal{I}, w \models_x \alpha \text{ and } \mathcal{I}, w \models_x \beta \);

v) \( \mathcal{I}, w \models_x \alpha \lor \beta \text{ if } \mathcal{I}, w \models_x \alpha \text{ or } \mathcal{I}, w \models_x \beta \);

vi) \( \mathcal{I}, w \models_x \alpha \rightarrow \beta \text{ if for all } w' \geq w: \mathcal{I}, w' \not\models_x \alpha \text{ or } \mathcal{I}, w' \models_x \beta \);

\(^2\) Note that by total we do not mean that functions cannot be left undefined. We may still have some term \( d \) for which \( \sigma^h(d) = \sigma^t(d) = u \).
vii) $\mathcal{I}, w \models_{\mathcal{F}} \forall x \alpha(x)$ if for each $c \in \text{Terms}_{\mathcal{F}}(\mathcal{C})$: $\mathcal{I}, w \models_{\mathcal{F}} \alpha(c)$;

viii) $\mathcal{I}, w \models_{\mathcal{F}} \exists x \alpha(x)$ if for some $c \in \text{Terms}_{\mathcal{F}}(\mathcal{C})$: $\mathcal{I}, w \models_{\mathcal{F}} \alpha(c)$.

The first condition above implies that an atom with an undefined argument will always be evaluated as false since, by definition, $u$ never occurs in ground atoms of $I^h$ or $I^t$. Something similar happens with equality: $\tau_1 = \tau_2$ will be false if any of the two operands, or even both, are undefined. As usual, $I$ is called a model of a theory $\Gamma$, written $I \models_{\mathcal{F}} \Gamma$, when $I, h \models_{\mathcal{F}} \phi$ for all $\phi \in \Gamma$.

**Proposition 1** (From Cabalar 2011). $\mathcal{I}, h \models_{\mathcal{F}} \neg \phi \iff \mathcal{I}, t \models_{\mathcal{F}} \neg \phi \iff \mathcal{I}, t \not\models_{\mathcal{F}} \phi$. □

We define next a particular ordering among $\text{SQHT}_{\mathcal{F}}$-interpretations. We say that $\mathcal{I}_1 = \langle \sigma^1, \tau^1, I^1, I^1 \rangle$ is smaller than $\mathcal{I}_2 = \langle \sigma^2, \tau^2, I^2, I^2 \rangle$, also written $\mathcal{I}_1 \preceq \mathcal{I}_2$ by abuse of notation, when $I^1 = I^2$, $\sigma^1 = \sigma^2$, $I^h_1 \subseteq I^h_2$ and $\sigma^h_1 \preceq \sigma^h_2$. That is, they have the same information at world $t$, but $\mathcal{I}_1$ can have less information than $\mathcal{I}_2$ at world $h$. Again, we write $\mathcal{I}_1 \prec \mathcal{I}_2$ when $\mathcal{I}_1 \preceq \mathcal{I}_2$ and $\mathcal{I}_1 \neq \mathcal{I}_2$. Nonmonotonicity is obtained by the next definition, which introduces the idea of equilibrium models for $\text{SQHT}_{\mathcal{F}}$.

**Definition 4** (Equilibrium model). A total model $\mathcal{I} = \langle \sigma, I \rangle$ of a theory $\Gamma$ is an equilibrium model if there is no strictly smaller interpretation $\mathcal{I}' \prec \mathcal{I}$ that is also a model of $\Gamma$. A set of atoms $\mathcal{I}$ is a stable model\(^3\) of $\Gamma$ if $\mathcal{I}$ is an equilibrium model for some $\sigma$.

\[3\] QEL with Evaluable Functions and Intensional Sets

In this section, we define $\text{SQHT}_{\mathcal{F}}$: a logic that extends $\text{SQHT}_{\mathcal{F}}$ with intensional sets. Intensional sets are terms of the form $\{\vec{x} : \vec{\tau}(\vec{x}) : \phi(\vec{x})\}$ where $\vec{x}$ is a tuple of variables and $\vec{\tau}(\vec{x})$ and $\phi(\vec{x})$ are respectively a tuple of terms and a formula with free variables $\vec{x}$.

Note that, as opposed to terms in $\text{SQHT}_{\mathcal{F}}$ (and also in first order logic), intensional sets are terms whose structure not only depends on other terms, but also on formulas. Hence, we define terms and formulas recursively such that the definition of $i$-terms will depend on the definition of $(i-1)$-formulas while the definition of $i$-formulas will depend on the definition of $i$-terms. We depart from a similar first-order signature $\Sigma = \langle \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ as in $\text{SQHT}_{\mathcal{F}}$, but we build the terms as follows. For any $i \geq 0$, we define an $i$-term as any of the following cases:

i) every constant $c \in C_0 \cup F_0$;

ii) every variable $x$.

iii) $f(\tau_1, \ldots, \tau_n)$, where $f/n \in (C \cup F)$ and $\tau_1, \ldots, \tau_n$ are $i$-terms, in their turn.

iv) the construct $\{\vec{\tau}_1, \ldots, \vec{\tau}_m\}$ (called extensional set), where $m \geq 0$ and $\vec{\tau}_1, \ldots, \vec{\tau}_m$ are $n$-tuples (of the same arity $n \geq 1$) of $i$-terms. If $m = 0$ we write $\emptyset$ instead of $\{\}$.

v) the construct $\{\vec{\tau} : \phi\}$ (called intensional set) if $i > 0$, $\phi$ is an $(i-1)$-formula and $\vec{\tau}$ is a tuple of $i$-terms.

Now, for any $i \geq 0$, $i$-atoms and $i$-formulas are defined over $i$-terms as follows:

vi) if $\tau_1, \ldots, \tau_n$ are $i$-terms and $p/n \in P$, then $p(\tau_1, \ldots, \tau_n)$ is an $i$-atom,

\(^3\) Apart from atoms, we could additionally include ground functional facts $f(\vec{c}) = d$. However, we only consider atoms here, for better comparison to other (non-functional) semantics of aggregates.
Recall that, here, $c$ stands for the unary tuple $\langle c \rangle$. 

\[ 4 \]

ix) if $x$ occurs in $\bar{x}$, then all free occurrences of $x$ in $\psi$ are bound in $\{\bar{x} : \bar{\tau} : \psi\}$ to the outermost intensional set.

iv) In the remaining cases, an occurrence of $x$ is bound (to some connective) in a formula iff it is so in some subformula; otherwise, it is free.

As in the case of SQHT$_{\bar{\tau}}$, when we write standalone formulas, we assume that all free variables are implicitly universally quantified. Similarly, we write $\{\bar{\tau} : \psi\}$ instead of $\{\bar{x} : \bar{\tau} : \psi\}$ when $\bar{x}$ contains exactly all free variables in $\bar{\tau}$. Note that intensional sets play a role similar to quantifiers. As an example, suppose we want to obtain the maximum number of times that the character Poirot is mentioned in an Agatha Christie book $b$, and assume that predicate word$(b, i, w)$ tells us that the $i$-th word of book $b$ is $w$. We
assume by now that we have functions \texttt{count} and \texttt{max} on sets: their meaning will be fixed later on. The set \{i : \textit{word}(b, i, \textit{Poirot})\} collects all occurrences of word \textit{Poirot} in book \textit{b}. Since \textit{i} is the only free variable to the left of ‘:’, the intensional set is an abbreviation of \{i : \textit{word}(b, i, \textit{Poirot})\} revealing that \textit{i} is being varied. On the contrary, variable \textit{b} is left free. Now, take the expression

\[
\text{max}\{\text{count}\{i : \textit{word}(b, i, \textit{Poirot})\} : \textit{author}(\textit{Agatha}, b)\}
\]  

(2)

The left term \text{count}\{i : \textit{word}(b, i, \textit{Poirot})\} contains a free occurrence of \textit{b} while \textit{i} is bound to the inner intensional set. Therefore, (2) actually stands for:

\[
\text{max}\{\text{count}\{i : \textit{word}(b, i, \textit{Poirot})\} : \textit{author}(\textit{Agatha}, b)\}
\]

that is obviously less readable than (2). However, in the general case, we may need to make use of the explicit list of quantified variables. For instance, if we want to parameterise the expression above for some author \textit{x} and character \textit{y} whose values are determined outside the term (as part of a formula), then we would necessarily write:

\[
\text{max}\{\text{count}\{i : \textit{word}(b, i, y)\} : \textit{author}(x, b)\}
\]

(3)

because the free occurrence of \textit{y} in \text{count}\{i : \textit{word}(b, i, y)\} could make us incorrectly assume that it is being varied in the set, as happened with \textit{b}.

### 3.1 Semantics

First, we need to define the domain in which terms are going to be interpreted. Given a set \textit{S}, let us define the set of all possible \textit{n}-tuples of elements from \textit{S}, for any \textit{n} \geq 1, as

\[
\text{Tup}(\textit{S}) \overset{\text{def}}{=} \bigcup_{n \geq 1} \{ \vec{e} \mid \vec{e} \in S^n \}
\]

Our domain \textit{D} is inductively constructed as follows:

\[
\text{D}^0 \overset{\text{def}}{=} \text{Terms}_\Sigma(\textit{C}) \quad \text{D}^{i+1} \overset{\text{def}}{=} \text{D}^i \cup 2^{\text{Tup}(\text{D}^i)}
\]

so that \textit{D} \overset{\text{def}}{=} \bigcup_{0 \leq i} \text{D}^i. We also define the subset of \textit{D} consisting of sets as \textit{S} \overset{\text{def}}{=} \text{D} \setminus \text{D}^0.

Definitions of assignments and interpretations are then straightforward: we just extend the domain of \sigma from \text{Terms}_\Sigma(\textit{C} \cup \textit{F}) to \text{Terms}(\textit{C} \cup \textit{F}) and replace the Herbrand Universe \text{Terms}_\Sigma(\textit{C}) by its corresponding \textit{D}.

**Definition 5 (Assignment).** An assignment \sigma for a signature \Sigma = \langle \textit{C}, \textit{F}, \textit{P} \rangle is a function \sigma : \text{Terms}(\textit{C} \cup \textit{F}) \to \textit{D} \cup \{ \textit{u} \} that maps some ground term in the Herbrand Universe or the special value \textit{u} \notin \text{Terms}(\textit{C} \cup \textit{F}) (standing for undefined) to any ground term in the language. Function \sigma must satisfy:

(i) \sigma(\textit{c}) \overset{\text{def}}{=} \textit{c} for all \textit{c} \in \textit{D}.

(ii) \sigma(\textit{f}(\tau_1, \ldots, \tau_n)) \overset{\text{def}}{=} \begin{cases} \textit{u} & \text{if } \sigma(\tau_i) = \textit{u} \text{ for some } i = 1 \ldots n \\ \{ \sigma(\textit{f}(\sigma(\tau_1), \ldots, \sigma(\tau_n))) \} & \text{otherwise} \end{cases}

(iii) \sigma(\{\tau_1, \ldots, \tau_n\}) \overset{\text{def}}{=} \begin{cases} \textit{u} & \text{if } \sigma(\tau_i) = \textit{u} \text{ for some } i = 1 \ldots n \\ \{ \sigma(\tau_1), \ldots, \sigma(\tau_n) \} & \text{otherwise} \end{cases}

where

\[
\sigma(\langle \tau_1, \ldots, \tau_m \rangle) \overset{\text{def}}{=} \begin{cases} \textit{u} & \text{if } \sigma(\tau_j) = \textit{u} \text{ for some } j = 1 \ldots m \\ \{ \sigma(\tau_1), \ldots, \sigma(\tau_m) \} & \text{otherwise} \end{cases}
\]
Note that we added a third case (iii) for extensional sets, but there is no restriction on the values of intensional sets: their meaning will be fixed later, once we describe the satisfaction of formulas.

**Definition 6** (Ordering ≤ among assignments). Given two assignments \( \sigma, \sigma' \) we define \( \sigma \preceq \sigma' \), as the condition: \( \sigma(\tau) = \sigma'(\tau) \) or \( \sigma(\tau) = u \) for all terms \( \tau \in \text{Terms}(C \cup F) \).

Interpretations \( I = \langle \sigma^h, \sigma^i, I^h, I^i \rangle \) have the same form as before, with \( I^h \subseteq I^i \subseteq HB \) and \( \sigma^h \preceq \sigma^i \), but under the extended definition of assignment and Herbrand Base.

**Definition 7** (S-satisfaction). Given an interpretation \( I = \langle \sigma^h, \sigma^i, I^h, I^i \rangle \), we define when \( I \ S \)-satisfies a formula \( \varphi \) at some world \( w \in \{ h, t \} \), written \( I, w \models_s \varphi \) as follows:

1. \( I, w \models_s p(\tau_1, \ldots, \tau_n) \) if \( p(\sigma^w(\tau_1), \ldots, \sigma^w(\tau_n)) \in I^w \) for any predicate \( p/n \in P \);
2. \( I, w \models_s \tau_1 = \tau_2 \) if \( \sigma^w(\tau_1) = \sigma^w(\tau_2) \neq u \);
3. \( I, w \not\models_s \top \); \( I, w \models_s \bot \);
4. \( I, w \models_s \alpha \land \beta \) if \( I, w \models_s \alpha \) and \( I, w \models_s \beta \);
5. \( I, w \models_s \alpha \lor \beta \) if \( I, w \models_s \alpha \) or \( I, w \models_s \beta \);
6. \( I, w \models_s \alpha \rightarrow \beta \) if for all \( w' \geq w \): \( I, w' \not\models_s \alpha \) or \( I, w' \models_s \beta \);
7. \( I, w \models_s \forall x \alpha(x) \) if for each \( c \in D : I, w \models_s \alpha(c) \);
8. \( I, w \models_s \exists x \alpha(x) \) if for some \( c \in D : I, w \models_s \alpha(c) \).

As usual, we write \( I \models_s \varphi \), when \( I, h \models_s \varphi \). □

It is easy to see that rules i)-vi) for \( \models_s \) are the exactly the same as for \( \models_f \). Rules vii) and viii) are just the result of replacing \( \text{Terms}_S(C) \) by \( D \). From this observation, we can immediately establish a correspondence between \( \models_s \) and \( \models_f \) as follows. Given any interpretation \( I = \langle \sigma^h, \sigma^i, I^h, I^i \rangle \) for a signature \( \Sigma = \langle C, F, P \rangle \), by \( \tilde{I} = \langle \tilde{\sigma}^h, \tilde{\sigma}^i, I^h, I^i \rangle \) we denote a SQHT\( _S \)-interpretation for the signature \( \Sigma' = \langle C \cup S, F, P \rangle \), for each \( w \in \{ h, t \} \), where the assignment \( \tilde{\sigma}^w \) is the restriction of \( \sigma^w \) to \( \text{Terms}_S(C \cup F \cup P) \).

**Proposition 2.** Any interpretation \( I \) and 0-formula \( \varphi \) satisfy: \( I, w \models_s \varphi \) iff \( \tilde{I}, w \models_f \varphi \). □

As, in general, not all SQHT\( _S \) formulas are SQHT\( _F \) formulas, we cannot directly extend this result beyond 0-formulas. However, we may still expect that what can be proved to be a tautology using the SQHT\( _F \) rules, still holds for SQHT\( _S \) formulas. For instance, the SQHT\( _S \) formula \( f = \{ \tilde{\tau} : \varphi \} \rightarrow f = \{ \tilde{\tau} : \varphi \} \) is an obvious tautology which is not a SQHT\( _F \) formula. However, we may replace every occurrence of the intensional set \( \{ \tilde{\tau} : \varphi \} \) by a fresh evaluable constant \( c \) and observe that \( f = c \rightarrow f = c \) is an SQHT\( _F \) tautology, using this to conclude that it is an SQHT\( _S \) tautology too. To formalise this intuition, let \( F_\varphi \) be a set disjoint from \( C \cup F \cup P \) containing a fresh constant per each different intensional set occurring in \( \varphi \) and let \( \kappa \) be a bijection mapping each intensional set in \( \varphi \) to its corresponding constant in \( F_\varphi \). Let us also denote by \( \kappa(\varphi) \) the result of replacing in \( \varphi \) each intensional set by its \( \kappa \) image. By \( \tilde{I} = \langle \tilde{\sigma}^h, \tilde{\sigma}^i, I^h, I^i \rangle \), we denote the SQHT\( _F \)-interpretation for the signature \( \Sigma' = \langle C \cup S, F \cup F_\varphi, P \rangle \) where, for each \( w \in \{ h, t \} \), we have \( \tilde{\sigma}^w(\tau) = \sigma^w(\kappa^{-1}(\tau)) \) if \( \tau \in F_\varphi \) and \( \tilde{\sigma}^w(\tau) = \sigma^w(\tau) \) otherwise.

**Proposition 3.** For any interpretation \( I \) and formula \( \varphi : I, w \models_s \varphi \) iff \( \tilde{I}, w \models_f \kappa(\varphi) \). □

Note that, as illustrated by Proposition 3, intensional sets act just as new fresh evaluable constants with respect to \( S \)-satisfaction. To fix the expected meaning of each
intensional set \( \{ \vec{r}(\vec{x}) : \varphi(\vec{x}) \} \) we still have to relate its value in \( \sigma^w \) to the satisfaction of formula \( \varphi(\vec{x}) \) in \( I \). Given \( \tau = \{ \vec{r}(\vec{x}) : \varphi(\vec{x}) \} \), let us define its extension at \( I, w \) as:

\[
\text{ext}^0(I, w, \tau) \overset{\text{def}}{=} \sigma^w(\vec{r}(\vec{c})) \mid I, w \models_s \varphi(\vec{c}) \text{ for some } \vec{c} \in D^{\vec{x}}
\]

\[
\text{ext}(I, w, \tau) \overset{\text{def}}{=} \begin{cases} 
\text{u} & \text{if } \text{u} \in \text{ext}^0(I, w, \tau) \\
\text{ext}^0(I, w, \tau) & \text{otherwise}
\end{cases}
\]

To put an example, if \( \tau_1 = \{ X \times n/X : p(X) \} \) and \( I_1 \) contains \( I_1^p = \{ p(0), p(1), p(2) \} \) then the set of tuples would be \( \{ 0 \times n/0, 1 \times n/1, 2 \times n/2 \} \). Since the obtained set is finite, its evaluation coincides with the case of extensional sets in Def. 5, item (iii).

In the example, if we have, for instance, \( \sigma^t(n) = 10 \), then \( \text{ext}(I_1, t, \tau_1) = \sigma^t( \{ 0 \times 10/0, 1 \times 10/1, 2 \times 10/2 \} ) = \{ \text{u}, 10, 10 \} = \text{u} \). On the other hand, if \( I_1^h \) consists of the sequence \( p(0), p(s(0)), p(s(s(0))), \ldots \), we similarly obtain the set \( \{ 0 \times n/0, s(0) \times n/s(0), s(s(0)) \times n/s(s(0)), \ldots \} \) which, being infinite, is not covered by Def. 5. With the definition of extension, for \( \sigma^t(n) = 10 \), we get \( \text{ext}(I_1, t, \tau_1) = \{ \sigma^t(0 \times 10/0, \sigma^t(s(0) \times 10/s(s(0)), \ldots \} ) = \{ \text{u}, 10, 10, \ldots \} = \text{u} \).

Now, given any interpretation \( I = (\sigma^h, \sigma^t, I^h, I^t) \), we define the assignments \( \sigma^w \) for \( w \in \{ h, t \} \) as follows. If \( \tau \in \text{Terms}^0(C \cup F) \) (i.e. not an intensional set) we can drop the \( I \) subindex, that is \( \sigma^w_I(\tau) \overset{\text{def}}{=} \sigma^w(\tau) \). If \( \tau \) is an intensional set

\[
\begin{align*}
\sigma^1_I(\tau) & \overset{\text{def}}{=} \text{ext}(I, t, \tau) \\
\sigma^2_I(\tau) & \overset{\text{def}}{=} \begin{cases} 
\text{ext}(I, h, \tau) & \text{if } \text{ext}(I, h, \tau) = \text{ext}(I, t, \tau) \\
\text{ext}(I, t, \tau) & \text{otherwise}
\end{cases}
\end{align*}
\]

As we can see, we have two potential sources of undeterminedness. One may appear because some element in \( \text{ext}(I, t, \tau) \) cannot be evaluated, as we had before with \( 0 \times n/0 \). But a second one may occur if the extension at \( h \) is different from the one at \( t \). For instance, for the same example, the extension of \( \tau_2 = \{ X : p(X) \} \) at \( t \) is \( \sigma^t(\text{ext}(I, t, \tau_2)) = \{ 0, 1, 2 \} \). If we had \( I_1^h = \{ p(0), p(2) \} \), then the extension at \( h \) would be \( \text{ext}(I, h, \tau_2) = \{ 0, 2 \} \neq \text{ext}(I, t, \tau_2) = \{ 0, 1, 2 \} \) and so \( \sigma^h_I(\tau_2) = \text{u} \). We also define the interpretation \( \text{Coh}(I) \overset{\text{def}}{=} (\sigma^2_I, \sigma^1_I, I^h, I^t) \). Note that \( \text{Coh}(I) \) is determined by the interpretation of predicates and terms in \( \text{Terms}^0(C \cup F) \). In this sense, given a SQHT\(_w\)-interpretation \( I \) for the signature \( \Sigma' = (C \cup S, F, P) \), by \( \text{Coh}(I) \), we also denote the interpretation \( \text{Coh}(J) \) for any interpretation \( J \) over the signature \( \Sigma' = (C, F, P) \) such that \( J \models I \).

**Definition 8** (Coherent interpretation). An interpretation \( I = (\sigma^h, \sigma^t, I^h, I^t) \) is said to be coherent (w.r.t. intensional sets) iff \( I = \text{Coh}(I) \).

**Definition 9** (Satisfaction). We say that an interpretation \( I \) satisfies (w.r.t. intensional sets) a formula \( \varphi \) at \( w \in \{ h, t \} \), in symbols \( I, w \models \varphi \), if both \( I \) is coherent and \( I, w \models_s \varphi \). We also write \( I \models \varphi \) when \( I, h \models \varphi \). Given a theory \( \Gamma \), we write \( I \models \Gamma \) if \( I \) is coherent and \( I \models \varphi \) for all formulas \( \varphi \in \Gamma \). We say that a formula \( \varphi \) is a tautology if every coherent interpretation \( I \) satisfies \( I \models \varphi \).

**Proposition 4.** For any SQHT\(_w\)-interp. \( I \) and 0-formula \( \varphi : I \models \varphi \) iff \( \text{Coh}(I) = \varphi \).

**Proposition 5.** Any coherent interpretation \( I \) satisfies:

i) \( I, w \models \varphi \) implies \( I, t \models \varphi \),

ii) \( I, w \models \neg \varphi \) if \( I, t \not\models \varphi \).
Proposition 6. Given a formula $\varphi$, the following statements hold:

i) if $\kappa(\varphi)$ is an SQHT$_{F}$ tautology, then $\varphi$ is an SQHT$_{F}$ tautology,

ii) if $\varphi$ is a 0-formula, then $\varphi$ is an SQHT$_{F}$ tautology iff it is a SQHT$_{F}$ tautology. \(\square\)

Nonmonotonicity is obtained with by the definition of equilibrium models for SQHT$_{F}$.

**Definition 10** (Equilibrium model). A total (coherent) model $I = \langle \sigma, I \rangle$ of a theory $\Gamma$ is an equilibrium model if there is no interpretation $I' \prec I$ which is also model of $\Gamma$. A set of atoms $I$ is a stable model of $\Gamma$ iff $\langle \sigma, I \rangle$ is an equilibrium model of $\Gamma$ for some $\sigma$. \(\square\)

Proposition 7. Let $\Gamma$ be a theory just containing 0-formulas over a signature $\Sigma = \langle C, F, P \rangle$ and let $I$ be a set of ground atoms. Then, $I$ is a stable model of $\Gamma$ according to Definition 10 iff $I$ is a stable model of $\Gamma$ according to Definition 4 with signature $\Sigma' = \langle C \cup S, F, P \rangle$. \(\square\)

Proposition 7 shows that our semantics is a conservative extension of (Cabalar 2011). Furthermore, as we will see later, our approach coincides with (Gelfond and Zhang 2014) and rejects vicious circles as shown by the following example from (Dovier et al. 2003).

**Example 1.** Consider the following logic program $P_1$:

\[
\begin{align*}
r(1). & \quad q(1). & \quad p(Y) \leftarrow Y = \{X : q(X)\} \\
r(2). & \quad q(2) \leftarrow \{X : r(X)\} \land p(Z).
\end{align*}
\]

$P_1$ has a unique equilibrium model $\langle \sigma_1, I_1 \rangle$ with $I_1 = \{q(1), p(\{1\}), r(1), r(2)\}$ and

\[
\sigma_1(\{X : r(X)\}) = \{1, 2\} \quad \sigma_1(\{X : q(X)\}) = \{1\}
\]

Under (Dovier et al. 2003) semantics, there exists a second stable model $I_2 = \{q(1), q(2), p(\{1, 2\}), r(1), r(2)\}$ not corresponding to any equilibrium model. To see why, consider the coherent interpretation $I_2' = \langle \sigma_2', I_2', I_2 \rangle$ with $I_2' = I_2 \setminus \{q(2), p(\{1, 2\})\}$ and:

\[
\begin{align*}
\sigma_2(\{X : r(X)\}) &= \{1, 2\} \quad \sigma_2(\{X : q(X)\}) = \{1, 2\} \\
\sigma_2'(\{X : r(X)\}) &= \{1, 2\} \quad \sigma_2'(\{X : q(X)\}) = u
\end{align*}
\]

Note also that any total, coherent interpretation that agrees with $I_2$ on the true atoms must be of the form $I_2 = \langle \sigma_2, I_2 \rangle$ and that $I_2' \prec I_2$. Hence, $I_2$ is not an equilibrium model. Finally, note that $I_2$ violates the Vicious-Circle Principle (Gelfond and Zhang 2014) because the truth of $q(2)$ depends on $p(\{1, 2\})$ which, in its turn, depends on the fact that $\{X : q(X)\}$ contains element 2. This last fact only holds if $q(2)$ holds. \(\square\)

## 4 Aggregates based on evaluable functions and intensional sets

From now on, we assume that $F$ contains a subset $A$ of function names of arity 1 used to denote aggregate names and that each aggregate name $f/1 \in A$ has an associated predefined function $f : S \rightarrow C \cup \{u\}$ that computes its value as expected (maximum, count, sum, etc). Now we further restrict Definition 8 to fix the meaning of aggregates:

**Definition 11.** An interpretation $I = \langle \sigma^h, \sigma^t, I^h, I^t \rangle$ is said to be coherent (w.r.t. aggregates) if it is coherent w.r.t. intensional sets and, in addition, it satisfies:
We say that an interpretation $\mathcal{I}$ is a model (w.r.t. aggregates) of a formula $\varphi$, in symbols $\mathcal{I} \models \varphi$, if both $\mathcal{I}$ is coherent and $\mathcal{I} \models_s \varphi$. Given a theory $\Gamma$, we write $\mathcal{I} \models \Gamma$ if $\mathcal{I}$ is coherent and $\mathcal{I} \models_s \varphi$ for all formulas $\varphi \in \Gamma$. \hfill $\square$

For the rest of the paper, we assume that the terms ‘coherent’ and ‘model’ are understood w.r.t. aggregates (Definition 11). In particular, we assume that $\mathcal{A}$ contains at least the aggregate names $\text{count}$ and $\text{sum}$ with the following semantics:

1. $\text{count}(S) = d$ if $d$ the number of elements in $S$,
2. $\text{sum}(S) = d$ if $S$ is a set of tuples, each of which has as an integer number as first component, and $d$ is the sum of all first components of all tuples in $S$,

$\hat{f}(S) = u$ with $f \in \{\text{count}, \text{sum}\}$ otherwise. We also assume that $\mathcal{P}$ and $\mathcal{F}$ respectively contain predicates $\leq, \geq, <, >, \neq$ and evaluable functions $+, -, \times, /$ for the arithmetic relations and functions with the standard meanings. Similarly, $\mathcal{P}$ and $\mathcal{F}$ also contain the predicate $\in$ and the evaluable functions $\cup, \cap$ and $\\setminus$ with the standard meanings in set theory. As usual, we use infix notation for arithmetic and set predicates and functions.

We also omit the parentheses around intensional sets, so we write $\text{count}\{X : p(X)\}$ instead of $\text{count}(\{X : p(X)\})$.

**Example 2.** Let $P_2$ be a theory over a signature with set of constants $\mathcal{C} = \{a, b\}$ that contains the rule (1) with $n = 1$ plus the fact $p(b)$. \hfill $\square$

The theory in Example 2 has no stable model. On the one hand, it is clear that every stable model $I$ must satisfy $p(b) \in I$. Furthermore, $\{p(b)\}$ is not a stable model because, every coherent total interpretation $\langle \sigma, \{p(b)\}\rangle$ must satisfy $\sigma(\{X : p(X)\}) = \{b\}$ and $\sigma(\text{count}\{X : p(X)\}) = 1$. Hence, $\langle \sigma, \emptyset \rangle$ does not satisfy (1). On the other hand, the only other alternative is $\{p(a), p(b)\}$ and we have that $I = \langle \sigma, \{p(a), p(b)\}\rangle$, with $\sigma(\{X : p(X)\}) = \{a, b\}$ and $\sigma(\text{count}\{X : p(X)\}) = 2$, satisfies (1). To show that $I$ is not an equilibrium model, let us define $I' = \langle \sigma', \sigma, I', I \rangle$ with $I' = \{p(b)\}$. It is easy to see that $I' \models p(b)$. Furthermore, we have that:

$$\text{ext}(I', t, \{X : p(X)\}) = \{a, b\} \neq \{b\} = \text{ext}(I', h, \{X : p(X)\})$$

(4)

Since these two extensions are different, it follows that $\sigma'(\{X : p(X)\}) = u$ and, consequently, that $I' \not\models \text{count}\{X : p(X)\} \geq 1$. In its turn, this implies that $I'$ is also a model of rule (1) with $n = 1$ and a model of $P_2$. Finally, it easy to check that $I' \not\subset I$ and, therefore, $I$ is not an equilibrium model. As we will see in Section 5, this behaviour agrees with Alog (Gelfond and Zhang 2014), but differs from other approaches like (Son and Pontelli 2007) and (Ferraris 2011) in which $\{p(a), p(b)\}$ is a stable model of $P_2$.

Interestingly, the use of evaluable functions allows defining aggregates within the logical language. First, let us recall the notion of *(directional) assignment* from (Cabalar 2011). By $f(\overline{\gamma}) := \tau'$ we denote the implication$^5$ $(\tau' = \tau') \rightarrow f(\overline{\gamma}) = \tau'$. Then, rather than providing predefined $\text{max}$ and $\text{min}$ functions, we can specify their meaning as aggregates

---

$^5$ Note that $\tau' = \tau'$ can be read as “$\tau'$ is defined.”
max and min by including, instead, the formulas:

\[
\begin{align*}
\text{max}(S) &: = X \land X \in S \land \neg \exists Y(Y \in S \land Y > X) \\
\text{min}(S) &: = X \land X \in S \land \neg \exists Y(Y \in S \land Y < X)
\end{align*}
\]

Clearly, \text{max}(\emptyset) and \text{min}(\emptyset) are always left undefined, because no rule body can satisfy \(X \in \emptyset\). Similarly, \text{count} can be inductively defined in terms of addition as follows:

\[
\begin{align*}
\text{count}(\emptyset) &= 0 \\
\text{count}(S) &= 1 + \text{count}(S\setminus\{Y\}) \quad Y \in S
\end{align*}
\]

That is, the cardinality of the empty set is 0, and the cardinality of any other set is 1 plus the cardinality of any set obtained by removing one of its elements.\(^6\) In general, we can easily define aggregate functions based on any associative and commutative binary function. For instance, to define the \text{sum} aggregate in terms of addition, we can just use:

\[
\begin{align*}
\text{sum}(\emptyset) &= 0 \\
\text{sum}(S) &= \text{sum}(S\setminus\{Y\}) + Y \quad Y \in S
\end{align*}
\]

If we now consider a program \(P_3\) containing these two rules, facts \(p(2)\) and \(p(3)\) plus

\[
q(Y) \leftarrow \text{sum}\{X : p(X)\} = Y
\]

we can check that it has a unique stable model \(I = \{p(2), p(3), q(5)\}\). The stable model \(I\) corresponds to the equilibrium model \(\langle \sigma, I \rangle\) which satisfies the assignments: \(\sigma(\text{sum}(\{2\})) = 2, \sigma(\text{sum}(\{3\})) = 3\) and \(\sigma(\text{sum}(\{2,3\})) = 5\).

### 5 Relation to GZ-aggregates for propositional formulas and Alog

A term \(\tau\) is said to be arithmetic if it only contains variables, numbers and arithmetic functions +, −, ×, etc. A (GZ-)set name is an intensional set of the form \(\vec{x} : \varphi\) where \(\vec{x}\) is a tuple of variables and \(\varphi\) is a 0-formula. A (GZ-)set atom is an expression of the form \(f(\tau) \leq \tau'\) with \(f \in A\) an aggregate function, \(\tau\) a set name, \(\tau'\) an arithmetic term and \(\leq \in \{\text{=}, \leq, \geq, <, >, \neq\}\) an arithmetic relation. A GZ-predicate atom is an expression of the form \(p(\tau_1, \ldots, \tau_n)\) with \(p/n \in \mathcal{P}\) a predicate name and \(\tau_1, \ldots, \tau_n \in D\). A GZ-atom is either a GZ-predicate atom or a set atom. GZ-formulas are the universal closure of formulas built over GZ-atoms using the connectives \(\lor, \land\) and \(\rightarrow\) as usual. A GZ-theory is a set of GZ-formulas. We say that a GZ-formula \(\varphi\) is ground when there are no quantifiers and all arithmetic terms have been evaluated, that is, the only variables occurring in \(\varphi\) are bound to some set name and the only arithmetic terms are numbers. A GZ-theory is said to be ground when all its formulas are ground. The following definitions extend the semantics of Alog (Gelfond and Zhang 2014) to arbitrary propositional formulas (Cabalar et al. 2017):

**Definition 12.** A set of atoms \(T\) satisfies a ground GZ-formula \(\varphi\), denoted by \(T \models_{cl} \varphi\), iff

1) \(T \not\models_{cl} \bot\)

\(^6\) For \text{count} and \text{sum}, we are assuming that set \(S\) is finite. Otherwise, we would need additional formalisation to deal with infinite sets and cardinalities.
\(T \models \varphi\) iff \(\varphi|_{\sim T}\) holds. 

\(\varphi\) is some atom in \(I\) if \(T \models \varphi\) and \(\varphi = f\{\bar{x}:\varphi(\bar{x})\} \subseteq n\) for all \(\varphi\) in \(I\). 

\(\varphi\) is the \(\preceq\)-minimal model of \(\text{Gr}(\Gamma)^T\). 

**Theorem 1.** For any \(GZ\)-theory \(\Gamma\), a set of atoms \(T\) is a stable model of \(\Gamma\) according to Definition 13 iff \(T\) is a stable model of \(\Gamma\) according to Definition 10. 

Let us return to Example 2 and recall we have shown that program \(P_2\) has no stable model according to Definition 10. To illustrate the behaviour of this program with respect to \(Alog\) (Definition 13), note first that the only possible candidate for being a stable model is the set \(I = \{p(a), p(b)\}\); otherwise, the rules would not be satisfied according to Definition 12. Then, we have that \(P_2^I\) corresponds to the normal program
\[
p(a) \leftarrow p(a) \land p(b) \\
p(b)
\]
whose unique \(\preceq\)-minimal model is \(\{p(b)\}\). Hence, \(I\) is not a stable model of \(P_2\). Intuitively, this is explained by the fact that our belief in \(p(a)\) depends on the extension of intensional set \(\{X : p(X)\}\) which, in its turn, depends on our belief in \(p(a)\), forming what Gelfond and Zhang (2014) call a “vicious circle.” According to the **vicious circle principle**, set \(I\) should be rejected as a stable model. In our approach, the “vicious circle” can be easily spotted by observing that evaluation of the set in world \(h\) is left undefined because its extension is different from the one at world \(t\), as shown in (4).

As an example of non-vicious definition, consider the following variation.

**Example 3.** Let \(P_4\) be the following program:
\[
p(a) \leftarrow \text{count}\{X : p(X) \land X \neq a\} \geq 1 \\
p(b)
\]

(10)
Again, the only candidate interpretation that satisfies all rules is \( I = \{ p(a), p(b) \} \), but the reduct corresponds now to:

\[
p(a) \leftarrow p(b) \\
p(b)
\]

whose unique minimal model is \( I \), becoming a stable model under Definition 13. Therefore, the same will happen under Definition 10. Let us put \( \tau = \{ X : p(X) \land X \neq a \} \). It is easy to see that \( \mathcal{I} = \langle \sigma, I \rangle \), with \( \sigma(\tau) = \{ b \} \) and \( \sigma(\text{count}(\tau)) = 1 \), satisfies (10) and obviously \( p(b) \) as well. Now take the smaller interpretation \( \mathcal{I}' = \langle \sigma', \sigma, I', I \rangle \) with \( I' = \{ p(b) \} \). Then, we have:

\[
\text{ext}(\mathcal{I}', t, \tau) = \{ b \} = \text{ext}(\mathcal{I}', h, \tau)
\]

so \( \sigma'(\tau) = \{ b \} \) and, consequently, \( \mathcal{I}' \models \text{count}(\tau) \geq 1 \). In its turn, this implies that \( \mathcal{I}' \) does not satisfy (10) and so is not a model of \( P_4 \). It is easy to see that there is no other smaller interpretation \( \mathcal{I}'' < \mathcal{I} \) that satisfies \( p(b) \), and so \( \mathcal{I} \) is an equilibrium model.

An interesting property of \( \text{Alog} \) is that it is always possible to introduce auxiliary variables for the aggregate value. For instance, we can always replace (1) by:

\[
p(a) \leftarrow \text{count}\{ X : p(X) \} = Y \land Y \geq n
\]

(11)

This transformation is not safe in other semantics such as (Son and Pontelli 2007; Faber et al. 2011; Ferraris 2011). In particular, under these semantics, if we take \( n = 0 \), a program consisting of (1) has a unique stable model \( \{ p(a) \} \) while a program consisting of (11) has no stable model. These two programs are equivalent in \( \text{Alog} \) and have no stable model. We can generalize the safety of this transformation to any context, using \( \text{SQHT}^= \):

**Proposition 8** (Existential variable introduction). Let \( p(\tau_1, \ldots, \tau_i, \ldots, \tau_n) \) be an atom. Then, \( p(\tau_1, \ldots, \tau_i, \ldots, \tau_n) \) is equivalent in \( \text{SQHT}^= \) to \( \exists x [ x = \tau_i \land p(\tau_1, \ldots, x, \ldots, \tau_n) ] \). \( \square \)

From Proposition 8, it immediately follows that rule (1) is equivalent to (11) with an existential quantifier \( \exists Y \) in the body that can be trivially removed.

### 6 Conclusions and Related Work

We have extended Quantified Equilibrium Logic with evaluable, partial functions by introducing *intensional sets*, that is, terms that allow defining elements in a set that satisfy some function or condition. By providing a logical interpretation, we define the semantics of these new expressions without any kind of syntactic restriction, so they can be arbitrarily nested within standard logical terms and formulas. The new extension yields a natural interpretation of an aggregate as an evaluable function applied to a set term. As a result, the semantics of an aggregate can be fixed within the logical language, by the addition of formulas fixing its meaning, rather than relying on an external, predefined function (although we assume that some elementary set predicates are available). This may become a powerful theoretical tool to analyse the fundamental properties of aggregate functions. We have also proved that, when restricted to the
syntactic fragment of language \textit{Alog}, our semantics coincides with that of (Gelfond and Zhang 2014).

Extensions in Logic Programming with sets can be traced back to (Kuper 1990) and (Beeri et al. 1991). The approach of (Dovier et al. 2003) is based on the stable model semantics with a reduct definition, but does not include evaluable functions or allow complex terms (beyond simple variables) to appear as the definition of intensional sets. Still, an important difference for the common syntactic fragment is that (Dovier et al. 2003) does not satisfy the \textit{vicious-circle free principle} as defined in (Gelfond and Zhang 2014) “no object or property can be introduced by the definition referring to the totality of objects satisfying this property” (see Example 1). The approaches (Lee and Meng 2009; Ferraris and Lifschitz 2010) or (Harrison et al. 2017) do not satisfy this principle either, but still share our orientation of defining the semantics of individual components of an aggregate. An important difference, however, is that these formalisations do not deal with general \textit{evaluable} (i.e. non-Herbrand) functions, while we use them as a starting point and then understand aggregates just as one more case whose arguments happen to be sets. This allows us a completely arbitrary use of aggregates as terms and of terms inside aggregates, leading to expressive constructions such as (5)-(6).

Our future work will include the study of non-strict functions and their relation to (Son and Pontelli 2007; Ferraris 2011). We will also study the possible formalisation under Free Logics as in (Cabalar et al. 2014) or the kind of properties that allow functions to be recursively defined as in (5)-(6), and their application to (Gelfond and Zhang 2014).

**Supplementary Material**

To view supplementary material for this article, please visit http://dx.doi.org/10.1017/S1471068418000169.

**References**


Cabalar, P. 2013. Setting the stage for ASP functions. \textit{ALP Newsletter}.


