Answer Sets for Propositional Theories

Paolo Ferraris

University of Texas at Austin, Austin TX 78712, USA otto@cs.utexas.edu

Abstract. Equilibrium logic, introduced by David Pearce, extends the concept of an answer set from logic programs to arbitrary sets of formulas. Logic programs correspond to the special case in which every formula is a "rule" — an implication that has no implications in the antecedent (body) and consequent (head). The semantics of equilibrium logic looks very different from the usual definitions of an answer set in logic programming, as it is based on Kripke models. In this paper we propose a new definition of equilibrium logic which uses the concept of a reduct, as in the standard definition of an answer set. Second, we apply the generalized concept of an answer set to the problem of defining the semantics of aggregates in answer set programming. We propose, in particular, a semantics for weight constraints that covers the problematic case of negative weights. Our semantics of aggregates is an extension of the approach due to Faber, Leone, and Pfeifer to a language with choice rules and, more generally, arbitrary rules with nested expressions.

1 Introduction

Equilibrium logic, introduced by Pearce [1997, 1999], extends the concept of an answer set [Gelfond and Lifschitz, 1988, 1991] from logic programs to arbitrary sets of formulas. Logic programs correspond to the special case in which every formula is a "rule" — an implication that has no implications (or equivalences) in the antecedent (body) and consequent (head).

The semantics of equilibrium logic looks very different from the usual definitions of an answer set in logic programming: it is based on Kripke models. In this note, we propose a new definition of equilibrium logic, equivalent to Pearce's definition, which uses the concept of a reduct, as in the one used in the standard definition of an answer sets.

Second, we apply the generalized concept of an answer set to the problem of defining the semantics of aggregates in answer set programming. The best proposal in this area published so far is due to Faber, Leone, and Pfeifer [2004]. The strong point of that paper is that it is applicable to aggregates that are neither monotone nor antimonotone (such as, for instance, weight constraints in which some some weights are positive and some are negative). It has two defects, however. First, it does not allow negation in aggregate expressions. Second, it does not cover choice rules, which play an important role in answer set programming. [1] Our semantics includes aggregates in the style of [Faber *et al.*,

[1] Both negation within aggregates and choice rules can be defined, in principle, as abbreviations for expressions containing auxiliary atoms.

2004] containing arbitrary formulas and also choice rules. We show also that the existence of an answer set for nondisjunctive program with aggregates of a very simple kind (weight constraints with the weights 1 and -1) is Σ_2^P-hard, as in the case of disjunctive programs.

In the following section we define answer sets for propositional theories, and we relate this definition to equilibrium logic and to the traditional definition of an answer set. In Section 3 we extend several important theorems about logic programs to propostional theories. In Section 4, we propose our semantics of aggregates, discuss its properties, and show, as an example, how it applies to representing a combinatorial auction with negative costs. Comparisons with other formalizations are given in Section 5.

2 Formulas, reducts and answer sets

2.1 Definition

For simplicity, we limit our attention to formulas without strong negation. We consider (propositional) formulas formed from atoms and connectives $\bot$, $\vee$, $\wedge$ and $\supset$. [2] A *theory* is a set of formulas. In the rest of the paper, F and G denote formulas, Γ a theory, X and Y sets of atoms, and $\otimes$ a binary connective.

We identify an interpretation with the set of atoms satisfied by it. We write $X \models F$ ($X \models \Gamma$) if X satisfies F (or Γ) in the sense of classical logic.

The *reduct* F^X of F relative to X is defined recursively:

- if $X \not\models F$ then $F^X = \bot$,
- if $X \models a$ (a is an atom) then $a^X = a$, and
- if $X \models F \otimes G$ then $(F \otimes G)^X = F^X \otimes G^X$.

This definition of a reduct is similar to a transformation proposed in [Osorio *et al.*, 2004, Section 4.2].

The reduct F^X can be alternatively defined as the formula obtained from F by replacing every outermost subformula not satisfied by X with $\bot$ (this alternative definition applies even if we treat $\neg$, $\top$ and $\equiv$ as primitive connectives).

For instance, if X contains p but not q then

$$((p \supset q) \vee (q \supset p))^X = \bot \vee (\bot \supset p).$$

It is easy to see that, for every X, Y, $\otimes$, F and G,

$$Y \models (F \otimes G)^X \text{ iff } X \models F \otimes G \text{ and } Y \models F^X \otimes G^X. \quad (1)$$

The *reduct* Γ^X of Γ relative to X is $\{F^X : F \in \Gamma\}$. A set X is an *answer set* for Γ if X is a minimal set satisfying Γ^X.

For instance, let Γ be $\{(p \supset q) \vee (q \supset p), p\}$. Set $\{p\}$ is an answer set for Γ because $\{p\}$ is a minimal model satisfying the reduct $\{\bot \vee (\bot \supset p), p\}$. It is not difficult to see that no other set of atoms is an answer set for Γ.

[2] $\neg F$ stands for $F \supset \bot$; $\top$ stands for $\bot \supset \bot$; $F \equiv G$ stands for $(F \supset G) \wedge (G \supset F)$.

2.2 Relationship to equilibrium logic

Theorem 1. *For any theory, its models in the sense of equilibrium logic are identical to its answer sets.*

Since in application to programs with nested expressions equilibrium logic is equivalent to the semantics defined in [Lifschitz *et al.*, 1999], Theorem 1 implies that our definition of an answer set extends the corresponding definition from that paper.

In the proof of Theorem 1, we write $\langle X, Y\rangle \models \Gamma$ (with $X \subseteq Y$) if the HT-interpretation [3] $\langle X, Y\rangle$ is a model of Γ.

Lemma 1. *For any X and Y such that $X \subseteq Y$ and any theory Γ,*

$$X \models \Gamma^Y \text{ iff } \langle X, Y\rangle \models \Gamma.$$

The lemma is proven first for the case when Γ is a singleton, by structural induction.

Proof of Theorem 1. According to the semantics of equilibrium logic (reproduced in [Lifschitz *et al.*, 2001, Section 4.4]), Y is a model of Γ iff

$$\langle Y, Y\rangle \models \Gamma \text{ and, for all proper subsets } X \text{ of } Y,\ \langle X, Y\rangle \not\models \Gamma.$$

In view of Lemma 1, this is the same as

$$Y \models \Gamma^Y \text{ and, for all proper subsets } X \text{ of } Y,\ X \not\models \Gamma^Y.$$

which means that Y is an answer set for Γ. □

2.3 Relationship to the traditional definition of reduct

A *nested expressions* is a formula that contains no implications $F \supset G$ with $G \neq \bot$, and no equivalences. [4] A *program with nested expressions* is a set of *rules* $F \leftarrow G$, where F and G are nested expressions. We will identify such a rule with the implication $G \supset F$.

In application to programs with nested expressions, our definition of a reduct is quite different from the traditional definition [Lifschitz *et al.*, 1999]. Consider, for instance, the following program:

$$p \leftarrow not\ q$$
$$q \leftarrow not\ r$$

According to [Lifschitz *et al.*, 1999], its reduct relative to $\{r\}$ is

$$p \leftarrow \top$$
$$q \leftarrow \bot;$$

[3] See [Lifschitz *et al.*, 2001, Section 2.1].

[4] Traditionally, in nested expressions conjunction is denoted by comma, disjunction by semicolon, and negation by "*not*".

with our definition, it is

$$\bot$$
$$q \leftarrow \bot.$$

The first reduct is satisfied, for instance, by $\{p\}$, while the second is unsatisfiable. However, some similarities between these formalisms exist. For instance, it is easy to see that for any formula F, $(\neg F)^X$, according to the new definition, is $\top$ when $X \not\models F$, and $\bot$ otherwise, as with the traditional definition. Indeed, if $X \models F$ then $X \not\models F \supset \bot$ and consequently

$$(\neg F)^X = (F \supset \bot)^X = \bot.$$

Otherwise, $X \models F \supset \bot$, so that

$$(\neg F)^X = (F \supset \bot)^X = F^X \supset \bot = \bot \supset \bot = \top.$$

The following proposition states a more general relationship between the new definition of the reduct and the traditional one. We denote by $F^{\underline{X}}$ the reduct of a nested expression F relative to X according to the definition from [Lifschitz *et al.*, 1999], and similarly for the reduct of a program.

Proposition 1. *For any program Π with nested expressions and any set X of atoms, Π^X is equivalent, in the sense of classical logic,*

- *to $\bot$, if $X \not\models \Pi$, and*
- *to the program obtained from $\Pi^{\underline{X}}$ by replacing all atoms that do not belong to X by $\bot$, otherwise.*

The proof of this proposition is based on the following lemma, proven by structural induction.

Lemma 2. *The reduct F^X of a nested expression F is equivalent, in the sense of classical logic, to the nested expression obtained from $F^{\underline{X}}$ by replacing all atoms that do not belong to X by $\bot$.*

Corollary 1. *Given two sets of atoms X and Y with $Y \subseteq X$ and any program Π, $Y \models \Pi^X$ iff $X \models \Pi$ and $Y \models \Pi^{\underline{X}}$.*

This corollary suggests another way to verify that the definition of an answer set proposed in this paper is equivalent to the usual one in the case of programs with nested expressions. If $X \not\models \Pi$ then X is not an answer set for Π under either semantics. Otherwise, for every subset Y of X, $Y \models \Pi^X$ iff $Y \models \Pi^{\underline{X}}$ by Corollary 1.

3 Properties of propositional theories

Several theorems about answer sets for logic programs can be extended to propositional theories. The proofs are omitted for lack of space.

Two theories Γ_1 and Γ_2 are *strongly equivalent* if, for every theory Γ, $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ have the same answer sets.

Proposition 2. *For any two theories Γ_1 and Γ_2, the following conditions are equivalent:*

(i) Γ_1 *is strongly equivalent to* Γ_2,
(ii) Γ_1 *is equivalent to* Γ_2 *in the logic of here-and-there, and*
(iii) for each set X *of atoms,* Γ_1^X *is equivalent to* Γ_2^X *in classical logic.*

The equivalence between (i) and (ii) is a generalization of the main result of [Lifschitz *et al.*, 2001], and it is an immediate consequence of Lemma 4 from that paper and our Theorem 1. The equivalence between (i) and (iii) is similar to Theorem 1 from [Turner, 2003].

The following claims require some definitions. An occurrence of an atom in a formula is *positive* if it is in the antecedent of an even number of implications. An occurrence is *strictly positive* if such number is 0. An occurrence of an atom in a formula is *negated* if it is in a subformula of the form $F \supset \bot$. For instance, in a formula $(p \supset \bot) \supset q$, the occurrences of p and q are positive, the one of q is strictly positive, and the one of p is negated.

The following proposition is an extension of the property that in each answer set of a program, each atom occurs in the head of a rule of that program [Lifschitz, 1996, Section 3.1].

Proposition 3. *Each answer set of a theory consists of atoms that have a strictly positive occurrence in some formula of that theory.*

The following two propositions were stated in [Ferraris and Lifschitz, 2005] in the case of logic programs.

Proposition 4 (Lemma on Explicit Definitions). *Let Γ be any propositional theory, and Q a set of atoms that do not occur in Γ. For each $q \in Q$, let $Def(q)$ be a formula that doesn't contain any atom from Q. Then $X \mapsto X \setminus Q$ is a 1–1 correspondence between the answer sets of $\Gamma \cup \{Def(q) \supset q : q \in Q\}$ and the answer sets of Γ.*

Proposition 5 (Completion Lemma). *Let Γ be any propositional theory, and Q a set of atoms that do not have positive, nonnegated occurrences in any rule of Γ. For each $q \in Q$, let $Def(q)$ be a formula such that all occurrences of elements of Q in $Def(q)$ are either positive or negated. Then $\Gamma \cup \{Def(q) \supset q : q \in Q\}$ and $\Gamma \cup \{Def(q) \equiv q : q \in Q\}$ have the same answer sets.*

The following proposition is essentially a generalization of the splitting set theorem from [Lifschitz and Turner, 1994] and [Erdoğan and Lifschitz, 2004].

Proposition 6 (Splitting Set Theorem). *Let Γ_1 and Γ_2 be two theories, and S a set of atoms such that all atoms that occur in Γ_1 are elements of S, and no element of S has strictly positive occurrences in Γ_2. Then a set of X is an answer set for $\Gamma_1 \cup \Gamma_2$ iff $X \cap S$ is an answer set for Γ_1 and X is an answer set for $(X \cap S) \cup \Gamma_2$.*

4 Representing aggregates

4.1 Definition

Aggregates are an important extension to logic programs, widely used in answer set programming. We define a *(ground) aggregate* as an expression of the form

$$op\langle\{F_1 = w_1, \ldots, F_n = w_n\}\rangle \prec N \tag{2}$$

where

- op is (a symbol for) a function from multisets of $\mathcal{R}$ (real numbers) to $\mathcal{R} \cup \{-\infty, +\infty\}$ (such as sum, product, min, max, etc.),
- $\{F_1 = w_1, \ldots, F_n = w_n\}$ $(n \geq 0)$ is a multiset where $F_1, \ldots, F_n$ are formulas, and $w_1, \ldots, w_n$ are (symbols for) real numbers ("weights"),
- $\prec$ is (a symbol for) a binary relation between real numbers, such as $\leq$ and $=$, and
- N is a (symbol for) a real number.

As an intuitive explanation of an aggregate, take the multiset W consisting of the weights w_i $(1 \leq i \leq n)$ such that F_i is "true". The aggregate is considered "true" if $op(W) \prec N$. For example,

$$sum\langle\{p = 1, q = 1\}\rangle \neq 1. \tag{3}$$

intuitively expresses the condition that both p are q are "true" or none of them.

To define the semantics of aggregates, we propose to identify (2) with the formula

$$\bigwedge_{I \subseteq \{1,\ldots,n\} \,:\, op(\{w_i \,:\, i \in I\}) \not\prec N} ((\bigwedge_{i \in I} F_i) \supset (\bigvee_{i \in \overline{I}} F_i)), \tag{4}$$

where $\overline{I}$ stands for $\{1, \ldots, n\} \setminus I$, and $\not\prec$ is the negation of $\prec$ [5].

For instance, if we consider aggregate (3), the conjunctive terms in (4) correspond to the ases when the sum of weights is 1, that is, when $I = \{1\}$ and $I = \{2\}$. The two implications are $q \supset p$ and $p \supset q$ respectively, so that (3) is

$$(q \supset p) \wedge (p \supset q). \tag{5}$$

Similarly,

$$sum\langle\{p = 1, q = 1\}\rangle = 1 \tag{6}$$

is

$$(p \vee q) \wedge \neg(p \wedge q). \tag{7}$$

Note that, even if (5) is classically equivalent to (7), they are not equivalent in the logic of here-and-there. This shows that it is generally incorrect to "move" a negation from a binary relation symbol (such as $\neq$) in front of the aggregate as the unary connective $\neg$.

Some properties of aggregates are stated in the following proposition.

[5] This definition, based on the idea of the translation from [Faber *et al.*, 2004], is meant to provide a very general semantics for aggregates, but we do not propose to use it directly for computing answer sets

Proposition 7. *For any aggregate* $op\langle S\rangle \prec N$ *where* S *is*

$$\{F_1 = w_1, \ldots, F_n = w_n\},$$

and any sets X *and* Y *of atoms,*

(a) $X \models op\langle S\rangle \prec N$ *iff* $op(\{w_i \ : \ X \models F_i\}) \prec N$*, and*
(b) $Y \models (op\langle S\rangle \prec N)^X$ *iff* $X \models op\langle S\rangle \prec N$ *and* $Y \models op\langle S^X\rangle \prec N$*,*

where S^X *stands for* $\{F_1^X = w_1, \ldots, F_n^X = w_n\}$*.*

Proposition 7(a) confirms that our proposal to identify (2) with (4) is in agreement with the intuitive meaning of an aggregate. Part (b) is similar to property (1) of binary connectives.

When a theory Γ is described using abbreviation (2) in its formulas, answer sets of Γ can be computed using Proposition 7 instead of a direct reference to (4).

Finally, it can be shown by Proposition 2 that if we want to identify (2) with a formula so that Proposition 7 holds then (4) is the only choice, modulo strong equivalence.

Proposition 7(a) follows from the fact that X satisfies an implication in (4) iff $I \neq \{j \ : \ X \models F_j\}$. The proof of part (b) uses the following lemma that is easily provable.

Lemma 3. *For any formulas* $F_1, \ldots, F_n$ $(n \geq 0)$*, any set* X *of atoms, and any connective* $\otimes \in \{\vee, \wedge\}$*,* $(F_1 \otimes \cdots \otimes F_n)^X$ *is classically equivalent to* $F_1^X \otimes \cdots \otimes F_n^X$*.*

4.2 Monotone aggregates

An aggregate $op\langle\{F_1 = w_1, \ldots, F_n = w_n\}\rangle \prec N$ is *monotone* if, for each pair of multisets W_1, W_2 such that $W_1 \subseteq W_2 \subseteq \{w_1, \ldots, w_n\}$, $op(W_2) \prec N$ is true whenever $op(W_1) \prec N$ is true. The definition of an *antimonotone* aggregate is similar, with $W_1 \subseteq W_2$ replaced by $W_2 \subseteq W_1$.

For instance,

$$sum\langle\{p = 1, q = 1\}\rangle > 1. \tag{8}$$

is monotone, and

$$sum\langle\{p = 1, q = 1\}\rangle < 1. \tag{9}$$

is antimonotone. An example of an aggregate that is neither monotone nor antimonotone is (3).

Proposition 8. *An aggregate* $op\langle\{F_1 = w_1, \ldots, F_n = w_n\}\rangle \prec N$ *is equivalent, in the logic of here-and-there, to*

$$\bigwedge_{I \subseteq \{1,\ldots,n\} \ : \ op(\{w_i \ : \ i \in I\}) \not\prec N} \ (\bigvee_{i \in \overline{I}} F_i)$$

if the aggregate is monotone, and to

$$\bigwedge_{I \subseteq \{1,\ldots,n\} \ : \ op(\{w_i \ : \ i \in I\}) \not\prec N} \ (\neg \bigwedge_{i \in I} F_i)$$

if the aggregate is antimonotone.

In other words, if $op\langle S\rangle \prec N$ is monotone then the antecedents of the implications in (4) can be dropped. Similarly, in case of antimonotone aggregates, the consequents of these implications can be replaced by $\bot$. In both cases, (4) is turned into a nested expression, if $F_1, \ldots, F_n$ are nested expressions.

For instance, the monotone aggregate (8) is

$$(p \vee q) \wedge (p \supset q) \wedge (q \supset p),$$

which is equivalent, in the logic of here and there, to

$$(p \vee q) \wedge q \wedge p$$

and then to $q \wedge p$. In the case of the antimonotone aggregate (9), the formula

$$((p \wedge q) \supset \bot) \wedge (p \supset q) \wedge (q \supset p)$$

is equivalent, in the logic of here-and-there, to

$$(\neg(p \wedge q)) \wedge \neg p \wedge \neg q,$$

and then to $\neg p \wedge \neg q$.

On the other hand, if an aggregate is neither monotone nor antimonotone, it may be not possible to find a nested expression equivalent, in the logic of here-and-there, to (4), even if $F_1, \ldots, F_n$ are nested expressions. This is the case for (3). Indeed, let A denote (3). Considering that this expression stands for (5), it is easy to check that $\langle\{p\}, \{p, q\}\rangle \not\models A$ and $\langle\emptyset, \{p, q\}\rangle \models A$. On the other hand, for any nested expression F, if $\langle\{p\}, \{p, q\}\rangle \not\models F$ then $\langle\emptyset, \{p, q\}\rangle \not\models F$ (easily provable by structural induction.)

Both parts of Proposition 8 can be proven, in the difficult direction, by strong induction on the cardinality of I.

4.3 Example

We consider the following variation of the combinatorial auction problem, which can be naturally formalized using an aggregate that is neither monotone nor antimonotone.

Joe wants to move to another town and has the problem of removing all his bulky furniture from his old place. He has received some bids: each bid may be for one piece or several pieces of furniture, and the amount offered can be negative (if the value of the pieces is lower than the cost of removing them). A junkyard will take any object not sold to bidders, for a price. The goal is to find a collection of bids for which Joe doesn't lose money, if there is any.

Assume that there are n bids, labeled from 1 to n. We express by the formulas

$$b_i \vee \neg b_i \tag{10}$$

$(1 \leq i \leq n)$ that Joe is free to accept any bid or not. Clearly, Joe cannot accept two bids that involve the selling of the same piece of furniture. So, for each pair i, j of such bids, we include the formula

$$\neg(b_i \wedge b_j). \tag{11}$$

Next, we need to express which pieces of the furniture have not been given to bidders. If there are m objects (numbered from 1 through m), we can express that an object i is sold by bid j by adding the rule

$$b_j \supset s_i \tag{12}$$

to our theory.

Finally, we need to express that Joe doesn't lose money by selling his items. This is done by the aggregate

$$sum\langle\{b_1 = w_1, \ldots, b_n = w_n, \neg s_1 = -c_1, \ldots, \neg s_m = -c_m\}\rangle \geq 0, \tag{13}$$

where each w_i is the amount of money (possibly negative) obtained by accepting bid i, and each c_i is the money requested by the junkyard to remove item i. Note that (13) is neither monotone nor antimonotone.

Proposition 9. *$X \mapsto X \cap \{b_1, \ldots, b_n\}$ is a 1–1 correspondence between the answer sets of theory consisting of formulas (10)–(13) and the solutions of this problem.*

5 Other formalisms

5.1 Programs with weight constraints

Weight constraints [Niemelä and Simons, 2000] can be viewed as aggregates of the form $sum\langle S\rangle \geq N$ (traditionally denoted by $N \leq S$) and $sum\langle S\rangle \leq N$ (denoted by $S \leq N$), where each formula in S is a literal. Weight constraints are one of the most commonly used kind of aggregates in logic programs, especially in the case of weights that are equal to 1 (cardinality constraints).

A *program with weight constraints* is a set of formulas of the form

$$W_1 \wedge \cdots \wedge W_n \supset a \tag{14}$$

($n \geq 0$) where a is an atom or $\bot$, and $W_1, ..., W_n$ are weight constraints. [6]

Theorem 2. *For every program with weight constraints, if all the weights are positive, then the answer sets under our semantics are identical to its answer sets in the sense of [Niemelä and Simons, 2000, Section 2.3].*

The proof consists in showing that, in the case of positive weights, $sum\langle S\rangle \geq N$ and $sum\langle S\rangle \leq N$ are equivalent, in the logic of here-and-there, to the nested expressions $[N \leq S]$ and $[S \leq N]$ defined in [Ferraris and Lifschitz, 2005]. Theorem 2 follows from this fact in view of Theorem 1 from [Ferraris and Lifschitz, 2005].

[6] For simplicity, we are considering only part of the syntax allowed in [Niemelä and Simons, 2000]. Every rule in sense of that paper can be equivalently rewritten as a set of rules of the form (14).

Our semantics is not equivalent to the semantics of [Niemelä and Simons, 2000] when the weights can be negative (as discussed in the introduction, our view of negative weights is equivalent to the one proposed in [Faber *et al.*, 2004].) According to [Ferraris and Lifschitz, 2005, Footnote 6], the traditional semantics for weight constraints may lead to some unintuitive results: program

$$(0 \leq \{p = 2, p = -1\}) \supset p \tag{15}$$

according to [Niemelä and Simons, 2000], has no answer sets, while

$$(0 \leq \{p = 1\}) \supset p$$

has one answer set $\{p\}$. Under our semantics, $\{p\}$ is the only answer set for both programs.

While weight constraints with positive weights only are either monotone or antimonotone, this is not the case when negative weights are allowed as in (13). In particular, it may not be possible to represent an aggregate of this kind by a nested expression.

This is the main reason why the translation from programs with weight constraints to programs with nested expressions of [Ferraris and Lifschitz, 2005] was limited to the case of positive weights only.

5.2 Complexity of programs with weight constraints

Under the semantics of [Niemelä and Simons, 2000], the existence of an answer set for programs with weight constraints is a NP-complete problem even in presence of negative weights. On the other hand, under our semantics, the place of this problem in the polynomial hierarchy is different.

Proposition 10. *Under the semantics of this paper, the existence of an answer set for a program with weight constraints is a Σ^2_P-complete problem.*

A similar result has been independently proven by Nicola Leone and Wolfgang Faber (personal communication).

The problem is clearly in Σ^2_P by the definition of an answer set. The Σ^2_P-hardness follows from the Σ^2_P-completeness of the existence of an answer set for disjunctive logic programs [Eiter and Gottlob, 1993, Corollary 3.8], and the following lemma, which provides a polynomial translation from disjunctive programs to programs with weight constraints.

Lemma 4. *Rule*

$$l_1 \wedge \cdots \wedge l_m \supset a_1 \vee \cdots \vee a_n$$

($n > 0, m \geq 0$) where $a_1, \ldots, a_n$ are atoms and $l_1, \ldots, l_m$ are literals, is strongly equivalent to the set of n implications ($i = 1, \ldots, n$)

$$(1 \leq \{l_1 = 1\}) \wedge \cdots \wedge (1 \leq \{l_m = 1\}) \wedge A_{i1} \wedge \cdots \wedge A_{in} \supset a_i,$$

where each A_{ij} stands for $0 \leq \{a_i = 1, a_j = -1\}$.

5.3 FLP-aggregates

We will now show that our semantics of aggregates is an extension of the semantics proposed by Faber, Leone and Pfeifer [2004]. An aggregate of the form (2) is a (ground) *FLP-aggregate* if $F_1, \dots, F_n$ are conjunctions of atoms. A (ground) *FLP-program* is a set of formulas

$$A_1 \wedge \cdots \wedge A_m \supset a_1 \vee \cdots \vee a_n \tag{16}$$

$(n, m \geq 0)$, where $a_1, \dots, a_n$ are atoms and $A_1, \dots, A_m$ are FLP-aggregates. [7]

Theorem 3. *The answer sets for a FLP-program under our semantics are identical to its answer sets in the sense of [Faber* et al.*, 2004].*

To prove this theorem we need to observe, first of all, that the definition of satisfaction of FLP-aggregates and FLP-programs in [Faber *et al.*, 2004] is equivalent to ours. The definition of a reduct is different, however. According to [Faber *et al.*, 2004], the *reduct* of a program Π with FLP-aggregates relative to X (we denote such reduct by $\Pi^{\underline{X}}_{=}$) consists of the rules (16) of Π such that $X \models A_1 \wedge \cdots \wedge A_m$. The definition of an answer set is again similar to ours: a set X of atoms is an *answer set* for a FLP-program Π if X is a minimal set satisfying $\Pi^{\underline{X}}_{=}$.

Lemma 5. *For any nested expression F without negations and any two sets X and Y of atoms such that $Y \subseteq X$, $Y \models F^X$ iff $Y \models F$.*

Lemma 6. *For any FLP-aggregate $op\langle S\rangle \prec N$ and any set X of atoms, if $X \models op\langle S\rangle \prec N$ then*

$$Y \models (op\langle S\rangle \prec N)^X \text{ iff } Y \models op\langle S\rangle \prec N.$$

Lemma 5 can be proven by structural induction. Lemma 6 follows from Lemma 5 and Proposition 7(b).

Proof of Theorem 3. It is easy to see that if $X \not\models \Pi$ then $X \not\models \Pi^X$ and $X \not\models \Pi^{\underline{X}}_{=}$, so that X is not an answer set under either semantics. Now assume that $X \models \Pi$. We will show that the two reducts are satisfied by the same subsets of X. It is sufficient to consider the case in which Π contains only one rule (16). If $X \not\models A_1 \wedge \cdots \wedge A_m$ then $\Pi^{\underline{X}}_{=} = \emptyset$, and Π^X is the tautology

$$\bot \supset (a_1 \vee \cdots \vee a_n)^X.$$

Otherwise, $\Pi^{\underline{X}}_{=}$ is rule (16), and Π^X is

$$A_1^X \wedge \cdots \wedge A_m^X \supset (a_1 \vee \cdots \vee a_n)^X.$$

These two reducts are satisfied by the same subsets of X by Lemmas 5 and 6. □

[7] The syntax of [Faber *et al.*, 2004] is more general in several ways. An expression of the form $\neg(op\langle S\rangle \prec N)$ in such syntax has the same meaning as $op\langle S\rangle \not\prec N$. Also, that paper allows literals as conjunctive terms in the antecedent of the implication (16). However, semantically, an atom a is not different from $sum\langle\{a = 1\}\rangle \geq 1$, and $\neg a$ is not different from $sum\langle\{a = 1\}\rangle \leq 0$.

6 Conclusion

We extended the definition of an answer set to arbitrary propositional theories. This definition of an answer set is equivalent to the definition of a model in equilibrium logic, so that it shares important properties of equilibrium logic such as the characterization of strong equivalence in terms of the logic of here-and-there. The new definition of reduct is different from the traditional definition [Lifschitz *et al.*, 1999] in the case of programs with nested expressions, but it is in some ways similar to it.

Even though propositional theories have a richer syntax, it turns out that any propositional theory can be expressed as a program with nested expression with the same answer sets [Cabalar and Ferraris, 2004]. In view of this fact, the possibility of defining answer sets for arbitrary propositional theories is not so surprising.

Propositional formulas cover both disjunctive rules with FLP-aggregates and choice rules. In the case of weight constraints, if negative weights are allowed then our semantics is not equivalent to the one of [Niemelä and Simons, 2000], but seems to have better properties. We have seen that this difference has consequences from the point of view of computational complexity.

It is possible, by Proposition 7, to view an aggregate $op\langle S\rangle \prec N$ as a primitive construct rather than an abbreviation for an exponentially larger formula. This is what is already happening in the answer set solver DLV [8], which partially supports programs with FLP-aggregates. On the other hand, viewing aggregates as formulas allows us to reason about strong equivalence in terms of the logic of here-and-there.

Acknowledgments

I am grateful to Pedro Cabalar, Selim Erdoğan, Joohyung Lee, David Pearce, Wanwan Ren and Hudson Turner for comments on a previous version of this paper. Special thanks go to Vladimir Lifschitz for many comments and discussions on the topic, and his careful reading of this paper. This research was partially supported by the National Science Foundation under Grant IIS-0412907.

References

[Cabalar and Ferraris, 2004] Pedro Cabalar and Paolo Ferraris. Propositional theories are equivalent to logic programs. In preparation, 2004.

[Eiter and Gottlob, 1993] Thomas Eiter and Georg Gottlob. Complexity results for disjunctive logic programming and application to nonmonotonic logics. In Dale Miller, editor, *Proceedings of International Logic Programming Symposium (ILPS)*, pages 266–278, 1993.

[8] `http://www.dbai.tuwien.ac.at/proj/dlv/ .`

[Erdoğan and Lifschitz, 2004] Selim T. Erdoğan and Vladimir Lifschitz. Definitions in answer set programming. In Vladimir Lifschitz and Ilkka Niemelä, editors, *Proceedings of 7th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-7)*, pages 114–126, 2004.

[Faber *et al.*, 2004] Wolfgang Faber, Nicola Leone, and Gerard Pfeifer. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proc. 9th European Conference on Artificial Intelligence (JELIA'04)*, 2004. Revised version: `http://www.wfaber.com/research/papers/jelia2004.pdf`.

[Ferraris and Lifschitz, 2005] Paolo Ferraris and Vladimir Lifschitz. Weight constraints as nested expressions. *Theory and Practice of Logic Programming*, 5:45–74, 2005.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080, 1988.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Lifschitz and Turner, 1994] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In Pascal Van Hentenryck, editor, *Proceedings of International Conference on Logic Programming (ICLP)*, pages 23–37, 1994.

[Lifschitz *et al.*, 1999] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25:369–389, 1999.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustin Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.

[Lifschitz, 1996] Vladimir Lifschitz. Foundations of logic programming. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, pages 69–128. CSLI Publications, 1996.

[Niemelä and Simons, 2000] Ilkka Niemelä and Patrik Simons. Extending the Smodels system with cardinality and weight constraints. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, pages 491–521. Kluwer, 2000.

[Osorio *et al.*, 2004] Mauricio Osorio, Juan Antonio Navarro, and José Arrazola. Safe beliefs for propositional theories. Accepted to appear at Annals of Pure and Applied Logic, 2004.

[Pearce, 1997] David Pearce. A new logical characterization of stable models and answer sets. In Jürgen Dix, Luis Pereira, and Teodor Przymusinski, editors, *Non-Monotonic Extensions of Logic Programming (Lecture Notes in Artificial Intelligence 1216)*, pages 57–70. Springer-Verlag, 1997.

[Pearce, 1999] David Pearce. From here to there: Stable negation in logic programming. In D. Gabbay and H. Wansing, editors, *What Is Negation?* Kluwer, 1999.

[Turner, 2003] Hudson Turner. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4,5):609–622, 2003.