Defining Answer Sets by a Second-Order Formula

Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz

Department of Computer Sciences, The University of Texas at Austin,
Austin TX 78712, USA {otto,appsmurf,vl}@cs.utexas.edu

Abstract. The answer sets of a logic program are usually defined in terms of reducts; they can be also characterized as the models of the program in the sense of equilibrium logic. In this paper we propose yet another characterization of answer sets, based on a second-order propositional formula that is syntactically very similar to the definition of circumscription. This characterization improves our understanding of the relationship between logic programming and other nonmonotonic formalisms. We apply it to the study of strong equivalence and show that it leads to a generalization of the concept of a loop formula.

1 Introduction

Formal nonmonotonic reasoning comes in two flavors. Among the earliest proposals, default logic [1], the nonmonotonic logic by McDermott and Doyle [2] and autoepistemic logic [3] are "radical," in the sense that they represent radical departures from first-order logic, both syntactically and semantically. Defaults, unlike first-order formulas, are comprised of premises, justifications and conclusions; nonmonotonic logic and autoepistemic logic use modal operators. The semantics of each of these formalisms is based on a fixpoint construction that is not similar to anything in the semantics of first-order logic.

On the other hand, the innovation involved in the definition of circumscription [4; 5] is relatively "moderate." Circumscription is simply a translation that turns a formula F into a stronger formula

$$F \wedge \cdots \tag{1}$$

that represents the new meaning of F. The additional conjunctive term is a minimality condition.[1] It is a second-order formula—an expression of the language that is familiar to the traditional logician almost as well as the language of first-order logic. Second-order formulas are needed whenever we want to talk about arbitrary properties of the objects under consideration (for instance, when we want to express the principle of induction for natural numbers in full generality, or the completeness of the continuum).

In the area of logic programming, the two approaches to the design of a nonmonotonic semantics are represented by the definition of program completion

[1] How that term is written depends on which predicates in F are circumscribed.

[6; 7] and the definition of an answer set (stable model) [8], among others. Syntactically, Prolog rules are similar to first-order formulas of a special form—to implications with a single atom in the consequent. The completion semantics is "moderate"; the completion formula for a logic program F can be thought of as a conjunction of the form (1), where the second conjunctive term is the only-if part of the completion. The similarity (and even a formal relationship) between completion and circumscription was identified soon after their invention [9].

Answer sets, on the other hand, were introduced in connection with the study of the relationship between Prolog and the "radical" autoepistemic logic [10]. The definition of answer sets, based on a fixpoint construction, is of the radical kind.

Further research suggested, however, that the radical theory of answer sets and the moderate theory of circumscription are related to each other more closely than one might initially assume. To begin with, the stable model of a program that does not contain negation is simply its minimal model. In [11, Section 3.4.1] this fact is used to give a characterization of answer sets in terms of circumscription that is applicable even in the presence of negation. On the other hand, the theorem on loop formulas [12] does not explicitly refer to circumscription, but it tells us how to describe the answer sets of a logic program F by a translation—a syntactic operation that turns F into a set of propositional formulas.

Furthermore, equilibrium logic [13] and its reformulation in [14] (reproduced in Section 2.1 below) extended the definition of an answer set to formulas that do not have the special structure of Prolog rules; like the argument F of the circumscription operator, they may combine atoms using arbitrary connectives in any order. This extension is important from the perspective of answer set programming because of its relation to the idea of an aggregate [14, Section 4]. For instance, the rule

$$\{P, Q\}1$$

(a cardinality constraint in the sense of [15]) can be viewed as shorthand for the formula

$$(P \vee \neg P) \wedge (Q \vee \neg Q) \wedge \neg(P \wedge Q)$$

[16].

In this note we show that the answer sets of an arbitrary propositional formula F can be characterized as the models a second-order propositional formula (QBF) of the form (1). The second conjunctive term of that formula does not, generally, express minimality, but syntactically it looks very much like the minimality condition in the definition of circumscription.

This fact improves our understanding of the relationship between logic programming and other nonmonotonic formalisms. It is also interesting for other reasons.

One of them has to do with the strong equivalence relation between logic programs. Its characterizations in terms of the logic of here-and-there [17] and in terms of reducts [18] have been extended to arbitrary propositional formulas [14, Proposition 2]. The new approach to answer sets allows us to do the same

for the characterization of strong equivalence based on classical logic given by Lin [19].

Second, the Lin/Zhao theorem on loop formulas, which was generalized to disjunctive logic programs in [20] and to programs with nested expressions in [21], can be now further extended to arbitrary propositional formulas.

Finally, we expect that this work will serve as the basis for a new approach to answer sets of programs with variables, which, like the theory of program completion, will be direct—it will not rely on grounding.

After reviewing the definitions of answer sets and circumscription in Section 2, we define the "stable model transformation" SM, similar to circumscription, and prove that the formula SM$[F]$ characterizes the answer sets of F (Section 3). Applications to strong equivalence and to loop formulas are discussed in Section 4.

2 Preliminaries

2.1 Review of Answer Sets

(Propositional) formulas are formed from propositional atoms (denoted in this paper by $P, Q, \dots$) and the 0-place connective $\bot$ using the binary connectives $\wedge$, $\vee$ and $\rightarrow$. We understand $\neg F$ as shorthand for $F \rightarrow \bot$, and $\top$ as shorthand for $\bot \rightarrow \bot$. A set X of atoms *satisfies* a formula F (symbolically, $X \models F$) if F is satisfied (in the sense of classical logic) by the truth assignment that makes the elements of X true and all other atoms false.

The *reduct* F^X of F relative to X is the formula obtained from F by replacing each maximal subformula that is not satisfied by X with $\bot$ [14]. We say that X is an *answer set* of F if X is a minimal (w.r.t. set inclusion) set satisfying F^X.

This definition is equivalent to the usual definition of an answer set for logic programs without strong (classical) negation, including disjunctive programs and even programs with nested expressions, provided that $\neg$ is understood as negation as failure [14, Section 2.2]. For instance, the disjunctive program

$$P\,;Q \leftarrow R$$
$$R \leftarrow \mathit{not}\ S$$

corresponds to the formula

$$(R \rightarrow (P \vee Q)) \wedge (\neg S \rightarrow R). \tag{2}$$

Let us check that $\{P, R\}$ is an answer set of this formula. The reduct of (2) relative to $\{P, R\}$ is

$$(R \rightarrow (P \vee \bot)) \wedge (\neg\bot \rightarrow R),$$

which is equivalent to $P \wedge R$. The set $\{P, R\}$ is indeed minimal among the sets satisfying this conjunction.

2.2 Review of Circumscription

The review of circumscription below is limited to the propositional case of parallel circumscription.

Second-order propositional formulas are formed from propositional atoms, an infinite supply of propositional variables (denoted in this paper by $p, q, \dots$) and $\bot$ using the connectives $\wedge$, $\vee$, $\rightarrow$ and the quantifiers $\forall$, $\exists$. The usual recursive definition of satisfaction for propositional formulas is extended to second-order propositional formulas without free variables as follows: a truth assignment (or a set of atoms) satisfies $\forall pF(p)$ if it satisfies both $F(\bot)$ and $F(\top)$; it satisfies $\exists pF(p)$ if it satisfies at least one of these two formulas. A second-order propositional formula is *logically valid* if its universal closure is satisfied by all truth assignments.

Quantifiers can be eliminated from any second-order propositional formula by repeatedly replacing parts of the form $\forall pF(p)$ with $F(\bot) \wedge F(\top)$, and parts of the form $\exists pF(p)$ with $F(\bot) \vee F(\top)$. This transformation turns logically valid formulas without free variables into tautologies.

Let $\mathbf{P}$ be a tuple of distinct atoms $P_1, \dots, P_n$, and $F(\mathbf{P})$ a propositional formula. The *circumscription of* $\mathbf{P}$ *in* $F(\mathbf{P})$, denoted by $\mathrm{CIRC}[F(\mathbf{P}); \mathbf{P}]$, is the second-order propositional formula

$$F(\mathbf{P}) \wedge \neg\exists \mathbf{p}(\mathbf{p} < \mathbf{P} \wedge F(\mathbf{p})),$$

where $\mathbf{p}$ is a tuple of n distinct propositional variables $p_1, \dots, p_n$, and $\mathbf{p} < \mathbf{P}$ stands for

$$(p_1 \rightarrow P_1) \wedge \cdots \wedge (p_n \rightarrow P_n) \wedge \neg((P_1 \rightarrow p_1) \wedge \cdots \wedge (P_n \rightarrow p_n)).$$

For instance,

$$\begin{aligned}
\mathrm{CIRC}[P \vee Q; P] &= (P \vee Q) \wedge \neg\exists p(p < P \wedge (p \vee Q)) \\
&\leftrightarrow (P \vee Q) \wedge \neg((\bot < P \wedge (\bot \vee Q)) \vee (\top < P \wedge (\top \vee Q))) \\
&\leftrightarrow (P \vee Q) \wedge \neg((P \wedge Q)) \vee (\bot \wedge \top)) \\
&\leftrightarrow (P \vee Q) \wedge \neg(P \wedge Q).
\end{aligned}$$

3 New Characterization of Answer Sets

3.1 Definition of the Operator SM

Let $P_1, \dots, P_n$ be all atoms occurring in a propositional formula F. By $\mathrm{SM}[F]$ we denote the second-order propositional formula

$$F \wedge \neg\exists \mathbf{p}(\mathbf{p} < \mathbf{P} \wedge F^*(\mathbf{p})),$$

where $\mathbf{P}$ stands for $P_1, \dots, P_n$, $\mathbf{p}$ is a tuple of n distinct propositional variables $p_1, \dots, p_n$, and $F^*(\mathbf{p})$ is defined recursively, as follows:

- $(P_i)^* = p_i$;

- $\bot^* = \bot$;
- $(F \wedge G)^* = F^* \wedge G^*$;
- $(F \vee G)^* = F^* \vee G^*$;
- $(F \to G)^* = (F^* \to G^*) \wedge (F \to G)$.

(There is no clause for $(\neg F)^*$ in this definition because we view $\neg F$ as an abbreviation for $F \to \bot$. Applying the operator SM to formulas written using this abbreviation is discussed below in connection with Proposition 1.)

The reader familiar with Kripke models will notice the similarity between the definition of $F^*(\mathbf{p})$ and Kripke's definition of satisfaction. We will return to this question in Section 3.2.

Example 1 Let F be the formula $P \wedge (P \to (Q \vee R))$, corresponding to the disjunctive program

$$P$$
$$Q\,;R \leftarrow P.$$

Then

$$\begin{aligned} F^* &= P^* \wedge (P \to (Q \vee R))^* \\ &= P^* \wedge (P^* \to (Q \vee R)^*) \wedge (P \to (Q \vee R)) \\ &= P^* \wedge (P^* \to (Q^* \vee R^*)) \wedge (P \to (Q \vee R)) \\ &= p \wedge (p \to (q \vee r)) \wedge (P \to (Q \vee R)) \\ &\leftrightarrow p \wedge (q \vee r) \wedge (P \to (Q \vee R)) \end{aligned}$$

and

$$\begin{aligned} \mathrm{SM}[F] &\leftrightarrow P \wedge (P \to (Q \vee R)) \\ &\quad \wedge \neg\exists pqr((p,q,r) < (P,Q,R) \wedge p \wedge (q \vee r) \wedge (P \to (Q \vee R))) \\ &\leftrightarrow P \wedge (Q \vee R) \wedge \neg\exists pqr((p,q,r) < (P,Q,R) \wedge p \wedge (q \vee r)) \\ &\leftrightarrow P \wedge (Q \vee R) \wedge \neg\exists qr((\top,q,r) < (P,Q,R) \wedge (q \vee r)) \\ &\leftrightarrow P \wedge (Q \vee R) \wedge \neg\exists qr((q,r) < (Q,R) \wedge P \wedge (q \vee r)) \\ &\leftrightarrow P \wedge (Q \vee R) \wedge \neg\exists qr((q,r) < (Q,R) \wedge (q \vee r)) \\ &\leftrightarrow P \wedge (Q \vee R) \wedge \neg\exists qr((\neg q \wedge r \wedge Q \wedge R) \vee (q \wedge \neg r \wedge Q \wedge R)) \\ &\leftrightarrow P \wedge (Q \vee R) \wedge \neg((Q \wedge R) \wedge \exists qr((\neg q \wedge r) \vee (q \wedge \neg r)) \\ &\leftrightarrow P \wedge (Q \vee R) \wedge \neg((Q \wedge R) \wedge \top) \\ &\leftrightarrow P \wedge (Q \vee R) \wedge \neg(Q \wedge R). \end{aligned}$$

The sets satisfying $\mathrm{SM}[F]$ are $\{P, Q\}$ and $\{P, R\}$, which are the two answer sets of F. This is an instance of the general property of the transformation SM stated below as Proposition 2.

Note that the operation $F \mapsto F^*(\mathbf{p})$ replaces the atoms from $\mathbf{P}$ with the corresponding variables from $\mathbf{p}$, and that it commutes with all connectives except implication. If we drop the second conjunctive term from the clause for implication in the definition of F^* then F^* will turn into the result of substituting $\mathbf{p}$ for $\mathbf{P}$ in F, and $\mathrm{SM}[F]$ will turn into $\mathrm{CIRC}[F;\mathbf{P}]$.

In one way, however, the operation $F \mapsto F^*(\mathbf{p})$ is essentially different from the substitution of $\mathbf{p}$ for $\mathbf{P}$: for two equivalent formulas F and G, $F^*(\mathbf{p})$ is not

necessarily equivalent to $G^*(\mathbf{p})$. Here is an example:

$$\begin{aligned}(P \to Q)^* &= (p \to q) \wedge (P \to Q),\\ (\neg P \vee Q)^* &= (\neg P)^* \vee Q^*\\ &= (\neg p \wedge \neg P) \vee q\\ &\leftrightarrow (p \to q) \wedge (P \to q).\end{aligned}$$

Applying the circumscription operator to each of two equivalent formulas gives two equivalent results; the operator SM does not have this property. This is not surprising: two (classically) equivalent formulas can have different answer sets.

In the following proposition, $\mathbf{p} \leq \mathbf{P}$ stands for $(p_1 \to P_1) \wedge \cdots \wedge (p_n \to P_n)$.

Proposition 1 *For any propositional formula formed from the atoms* $\mathbf{P}$*, the formula*

$$\mathbf{p} \leq \mathbf{P} \to ((\neg F)^* \leftrightarrow \neg F)$$

is logically valid.

This proposition is useful when we want to apply SM to a formula written using the abbreviation $\neg F$. It shows that when $(\neg F)^*$ occurs in a conjunction of the form $\mathbf{p} < \mathbf{P} \wedge \ldots$, it can be equivalently replaced with $\neg F$. The proposition is immediate from the following lemma, which can be easily verified by induction:

Lemma 1 *For any propositional formula formed from the atoms* $\mathbf{P}$*, the formula*

$$\mathbf{p} \leq \mathbf{P} \to (F^* \to F)$$

is logically valid.

Example 2 Let F be the formula $\neg P \to Q$, corresponding to the one-rule program

$$Q \leftarrow \mathit{not}\ P.$$

The calculation of SM$[F]$ below uses two methods for eliminating quantifiers over propositional variables different from the general procedure described in Section 2.2. First, $\exists pF(p)$ can be equivalently replaced with $F(\bot)$ if all occurrences of p in $F(p)$ are negative. Second, if F and G are formulas that don't contain p then

$$\exists p((F \to p) \wedge (p \to G))$$

can be equivalently replaced with $F \to G$.

$$\begin{aligned}\text{SM}[F] &\leftrightarrow (\neg P \to Q) \wedge \neg\exists pq((p,q) < (P,Q) \wedge (\neg P \to Q)^*)\\ &\leftrightarrow (\neg P \to Q) \wedge \neg\exists pq((p,q) < (P,Q) \wedge ((\neg P)^* \to Q^*) \wedge (\neg P \to Q))\\ &\leftrightarrow (P \vee Q) \wedge \neg\exists pq((p,q) < (P,Q) \wedge ((\neg P)^* \to Q^*))\\ &\leftrightarrow (P \vee Q) \wedge \neg\exists pq((p,q) < (P,Q) \wedge (\neg P \to q))\\ &\leftrightarrow (P \vee Q) \wedge \neg\exists q(\exists p((p,q) < (P,Q)) \wedge (\neg P \to q))\\ &\leftrightarrow (P \vee Q) \wedge \neg\exists q((\bot,q) < (P,Q) \wedge (\neg P \to q))\\ &\leftrightarrow (P \vee Q) \wedge \neg\exists q((P \vee Q) \wedge (q \to (P \wedge Q)) \wedge (\neg P \to q))\\ &\leftrightarrow (P \vee Q) \wedge \neg\exists q((q \to (P \wedge Q)) \wedge (\neg P \to q))\\ &\leftrightarrow (P \vee Q) \wedge \neg(\neg P \to (P \wedge Q))\\ &\leftrightarrow (P \vee Q) \wedge \neg P\\ &\leftrightarrow \neg P \wedge Q.\end{aligned}$$

This formula is satisfied by $\{Q\}$, which is the only answer set of F.

Example 3 Let F be the formula $P \vee \neg P$, corresponding to the choice rule $\{P\}$. Then

$$\begin{aligned} \mathrm{SM}[F] &\leftrightarrow (P \vee \neg P) \wedge \neg\exists p(p < P \wedge (p \vee \neg P)) \\ &\leftrightarrow \neg\exists p(p < P \wedge (p \vee \neg P)) \\ &\leftrightarrow \neg(P \wedge (\bot \vee \neg P)) \\ &\leftrightarrow \top. \end{aligned}$$

This formula is satisfied by $\emptyset$ and $\{P\}$, which are the two answer sets of F.

3.2 Main Theorem

The following proposition generalizes the observations regarding the relationship between SM[F] and the answer sets of F that we made in Examples 1–3. It is the main theorem of this paper.

Proposition 2 *A set X of atoms is an answer set of F iff X satisfies* SM[F].

For any set $Y \subseteq \mathbf{P}$, by $\overrightarrow{Y}$ we denote the tuple $(Y_1, \ldots, Y_n)$ where

$$Y_i = \begin{cases} \top, & \text{if } P_i \in Y; \\ \bot, & \text{otherwise.} \end{cases}$$

Lemma 2 *For any set X of atoms and any subset Y of X, $Y \models F^X$ iff $X \models F^*(\overrightarrow{Y})$.*

Proof. by induction on F. *Case 1:* F is an atom P_i, so that $F^*(\mathbf{p})$ is p_i. If $P_i \in X$ then F^X is P_i; $F^*(\overrightarrow{Y})$ is $\top$ or $\bot$ depending on whether or not $P_i \in Y$, that is, depending on whether or not Y satisfies F^X. Otherwise F^X is $\bot$; since $Y \subseteq X$, $P_i \notin Y$, so that $F^*(\overrightarrow{Y})$ is $\bot$ too.

Case 2: F is $\bot$. Each of the formulas F^X, $F^*(\overrightarrow{Y})$ is $\bot$.

Case 3: F is $G \wedge H$, so that $F^*(\overrightarrow{Y})$ is $G^*(\overrightarrow{Y}) \wedge H^*(\overrightarrow{Y})$. If X satisfies $G \wedge H$ then F^X is $G^X \wedge H^X$, and we use the induction hypothesis. Otherwise F^X is $\bot$, and X doesn't satisfy at least one of the formulas G, H. Assume, for instance, that $X \not\models G$. Then G^X is $\bot$, and, by the induction hypothesis, $X \not\models G^*(\overrightarrow{Y})$. It follows that $X \not\models F^*(\overrightarrow{Y})$.

Case 4: F is $G \vee H$, so that $F^*(\overrightarrow{Y})$ is $G^*(\overrightarrow{Y}) \vee H^*(\overrightarrow{Y})$. If X satisfies $G \vee H$ then F^X is $G^X \vee H^X$, and we use the induction hypothesis. Otherwise F^X is $\bot$, and X satisfies neither G nor H. Then each of the formulas G^X, H^X is $\bot$, and, by the induction hypothesis, X satisfies neither $G^*(\overrightarrow{Y})$ nor $H^*(\overrightarrow{Y})$. It follows that $X \not\models F^*(\overrightarrow{Y})$.

Case 5: F is $G \to H$, so that $F^*(\overrightarrow{Y})$ is

$$(G^*(\overrightarrow{Y}) \to H^*(\overrightarrow{Y})) \wedge (G \to H). \tag{3}$$

If X satisfies the second term $G \to H$ of (3) then F^X is $G^X \to H^X$; from the induction hypothesis we conclude that X satisfies this formula iff it satisfies the first term of (3). Otherwise F^X is $\bot$; X doesn't satisfies (3) because it doesn't satisfy the second conjunctive term.

In view of Lemma 1 from [14], our Lemma 2 can be equivalently stated as follows: For any set X of atoms and any subset Y of X, $X \models F^*(\overrightarrow{Y})$ iff (Y, X) satisfies F in the logic of here-and-there. This version of Lemma 2 can be proved by induction on F as well, and the proof exploits the analogy between the definition of $F^*(\mathbf{p})$ and the definition of satisfaction in the logic of here-and-there.

Proof of Proposition 2. SM[F] is equivalent to

$$F \wedge \forall \mathbf{p}(\mathbf{p} < \mathbf{P} \to \neg F^*(\mathbf{p})),$$

and consequently to

$$F \wedge \bigwedge_{Y \subseteq \mathbf{P}} (\overrightarrow{Y} < \mathbf{P} \to \neg F^*(\overrightarrow{Y})).$$

It is clear that X satisfies $\overrightarrow{Y} < \mathbf{P}$ iff Y is a proper subset of X. We conclude that $X \models \mathrm{SM}[F]$ iff

(i) $X \models F$, and
(ii) for every proper subset Y of X, $X \not\models F^*(\overrightarrow{Y})$.

It is easy to check by induction on F that $X \models F^X$ iff $X \models F$. Using this fact and Lemma 2, we can restate conditions (i) and (ii) as follows:

(i′) $X \models F^X$, and
(ii′) for every proper subset Y of X, $Y \not\models F^X$.

This is equivalent to saying that X is an answer set of F.

Theorem 4 from [11], describing a relation between answer sets of "traditional" programs and circumscription, can be derived from Proposition 2 as a special case.

4 Applications

4.1 Strong Equivalence

Recall that propositional formulas F and G are *strongly equivalent* to each other if, for every formula H, $F \wedge H$ and $G \wedge H$ have the same answer sets [17], [14]. In the following theorem, $\mathbf{P}$ is a tuple $P_1, \ldots, P_n$ of distinct atoms containing all atoms occurring in F, G; $\mathbf{P}'$ is a tuple $P'_1, \ldots, P'_n$ of n distinct atoms disjoint from $\mathbf{P}$.

Proposition 3 *Formulas F and G are strongly equivalent iff*

$$\mathbf{P}' \leq \mathbf{P} \to (F^*(\mathbf{P}') \leftrightarrow G^*(\mathbf{P}')) \tag{4}$$

is a tautology.

Proof. By Proposition 2 from [14], F is strongly equivalent to G iff $F^X \leftrightarrow G^X$ is a tautology. It is easy to see that all atoms occurring in F^X, G^X belong to X. Using this fact and Lemma 2, we conclude:

$F^X \leftrightarrow G^X$ is a tautology
iff for all X and all $Y \subseteq X$, $Y \models F^X$ iff $Y \models G^X$
iff for all X and all $Y \subseteq X$, $X \models F^*(\overrightarrow{Y})$ iff $X \models G^*(\overrightarrow{Y})$
iff for all X and all $Y \subseteq X$, $X \models F^*(\overrightarrow{Y}) \leftrightarrow G^*(\overrightarrow{Y})$
iff for all X and Y, $X \models \overrightarrow{Y} \leq \mathbf{P} \to (F^*(\overrightarrow{Y}) \leftrightarrow G^*(\overrightarrow{Y}))$
iff for all Y, $\overrightarrow{Y} \leq \mathbf{P} \to (F^*(\overrightarrow{Y}) \leftrightarrow G^*(\overrightarrow{Y}))$ is a tautology
iff $\mathbf{P}' \leq \mathbf{P} \to (F^*(\mathbf{P}') \leftrightarrow G^*(\mathbf{P}'))$ is a tautology.

Proposition 3 is a generalization of Theorem 1 from [19] to arbitrary propositional formulas. We can use it, for instance, to prove that the formula

$$(P \to Q) \vee R \tag{5}$$

is strongly equivalent to

$$(P \to (Q \vee R)) \wedge (\neg Q \to (\neg P \vee R)).$$

In view of Proposition 3 and Lemma 1, it is sufficient to derive, in classical propositional logic, the equivalence between

$$((P' \to Q') \wedge (P \to Q)) \vee R'$$

and

$$(P' \to (Q' \vee R')) \wedge (P \to (Q \vee R)) \wedge (\neg Q \to (\neg P \vee R')) \wedge (\neg Q \to (\neg P \vee R))$$

from

$$(P' \to P) \wedge (Q' \to Q) \wedge (R' \to R).$$

This can be easily done by considering the cases R', $\neg R'$.

Formula (4) in the statement of Proposition 3 can be replaced with

$$F^*(\mathbf{P} \wedge \mathbf{P}') \leftrightarrow G^*(\mathbf{P} \wedge \mathbf{P}'), \tag{6}$$

where $\mathbf{P} \wedge \mathbf{P}'$ stands for the tuple $P_1 \wedge P_1', \ldots, P_n \wedge P_n'$. Indeed, it is easy to check that (4) is a tautology iff (6) is a tautology.

4.2 Loop Formulas

The definition of a loop in [12] is based on the definition of a (positive) dependency graph of a logic program. A traditional program, written as a formula, is a conjunction of implications of the form

$$\bigwedge_i Q_i \wedge \bigwedge_j \neg R_j \to P, \tag{7}$$

where P, Q_i and R_j are atoms; the edges of its dependency graph lead from P to all atoms Q_i. Our goal here is to define loops for an arbitrary propositional formula, and we will start by introducing dependency graphs for arbitrary formulas.

As usual, about (an occurrence of) a subformula in a formula F we say that it is *positive* if the number of implications in F containing it in the antecedent is even, and that it is *strictly positive* if that number is 0. An occurrence of an atom in F is *negated* if it belongs to a subformula of the form $\neg G$ (that is, $G \to \bot$).

The *dependency graph* of a formula F is the directed graph such that

- its vertices are the atoms that occur in F, and
- it has an edge from a vertex P to a vertex Q if F has a strictly positive subformula $G \to H$ such that P has a strictly positive occurrence in H, and Q has a positive, non-negated occurrence in G.

For example, the dependency graph of (5) has one edge, from Q to P. The dependency graph of the formula

$$((P \to Q) \to R) \to S$$

has two edges—from S to R and from S to P. The dependency graph of

$$(\neg P \to R) \to S$$

has only one edge, from S to R.

It is clear that any edge of the dependency graph of F starts at an atom that has a strictly positive occurrence in F.

A nonempty set L of atoms from F is a *loop* of F if, for any $P, Q \in L$ there exists a path (possibly of length 0) from P to Q in the dependency graph of F whose vertices belong to L. It is clear that every set consisting of a single atom is a loop.

In the following theorem, F is any propositional formula formed from the atoms $\mathbf{P}$. For any $Y \subseteq \mathbf{P}$, by $\bigwedge Y$ and $\bigvee Y$ we denote the conjunction and, respectively, disjunction of the elements of Y. By $\mathbf{P}^Y_\bot$ we denote the tuple obtained from $\mathbf{P}$ by substituting $\bot$ for all atoms that belong to Y.

Proposition 4 *For any subset X of $\mathbf{P}$, the following conditions are equivalent:*

(a) X is an answer set of F;

(b) X satisfies F and the formulas

$$\bigwedge Y \to \neg F^*(\mathbf{P}_\bot^Y) \tag{8}$$

for all nonempty subsets Y of $\mathbf{P}$;

(c) X satisfies F and the formulas (8) for all loops Y of F;

(d) X satisfies F and the formulas

$$\bigvee Y \to \neg F^*(\mathbf{P}_\bot^Y) \tag{9}$$

for all subsets Y of $\mathbf{P}$;

(e) X satisfies F and the formulas (9) for all loops Y of F.

The equivalence between (a) and (b) is essentially a generalization of results of [22] and [23]; see [21, Section 4.1]. The equivalence between (a) and (e) is essentially a generalization of the theorem on loop formulas from [12].

To illustrate Proposition 4, let's apply it to formula (5). In this case $F^*(p, q, r)$ is

$$((p \to q) \wedge (P \to Q)) \vee r.$$

The loops of (5) are the singletons $\{P\}$, $\{Q\}$, $\{R\}$, and formulas (8) for these sets Y are

$$\begin{array}{l} P \to \neg((\bot \to Q) \wedge (P \to Q)) \vee R), \\ Q \to \neg((P \to \bot) \wedge (P \to Q)) \vee R), \\ R \to \neg((P \to Q) \wedge (P \to Q)) \vee \bot). \end{array} \tag{10}$$

(Formula (9) is identical to (8) whenever Y is a singleton.) Proposition 4 tells us that the answer sets of (5) can be characterized as the sets that satisfy both (5) and (10). Formulas (10) can be simplified as follows:

$$\begin{array}{ll} P \to \neg Q, & P \to \neg R, \\ Q \to P, & Q \to \neg R, \\ R \to P, & R \to \neg Q. \end{array}$$

Consequently, the conjunction of these formulas is equivalent to $\neg Q \wedge \neg R$, and the conjunction of (5) with (10) is equivalent to $\neg P \wedge \neg Q \wedge \neg R$. The empty set is the only answer set of (5).

The proof of Proposition 4 consists of two parts. First we will show that conditions (a), (b) and (d)—those not referring to the concept of a loop—are equivalent to each other. Then we will outline a proof of the fact that they are also equivalent to (c) and (e).

Note first that, by Proposition 2,

$$\begin{aligned} X \text{ is an answer set of } F \quad &\text{iff} \quad X \models \mathrm{SM}[F] \\ &\text{iff} \quad X \models F \wedge \forall \mathbf{p}(\mathbf{p} < \mathbf{P} \to \neg F^*(\mathbf{p})) \\ &\text{iff} \quad X \models F \wedge \bigwedge_{Y \subseteq \mathbf{P}} \left(\overrightarrow{Y} < \mathbf{P} \to \neg F^*(\overrightarrow{Y}) \right) \\ &\text{iff} \quad X \models F \wedge \bigwedge_{Y \subseteq \mathbf{P}} \left(\overrightarrow{Y} < \overrightarrow{X} \to \neg F^*(\overrightarrow{Y}) \right) \\ &\text{iff} \quad X \models F \wedge \bigwedge_{Y \subseteq \mathbf{P}:\ Y \subset X} \neg F^*(\overrightarrow{Y}). \end{aligned}$$

It follows that condition (a) from the statement of Proposition 4 can be equivalently reformulated as follows:

(a′) X satisfies F and the conjunction

$$\bigwedge_{Y \subseteq \mathbf{P}:\ Y \subset X} \neg F^*(\overrightarrow{Y}).$$

On the other hand, Lemma 3 below shows that conditions (b) and (d) can be equivalently restated as follows:

(b′) X satisfies F and the formulas

$$\bigwedge Y \to \neg F^*(\overrightarrow{X \setminus Y})$$

for all nonempty subsets Y of $\mathbf{P}$;

(d′) X satisfies F and the formulas

$$\bigvee Y \to \neg F^*(\overrightarrow{X \setminus Y})$$

for all subsets Y of $\mathbf{P}$.

Lemma 3 *For any sets X, Y of atoms, $X \models F^*(\mathbf{P}^Y_\perp)$ iff $X \models F^*(\overrightarrow{X \setminus Y})$.*

Proof. by induction on F. *Case 1:* F is an atom. If $F \in Y$ then each of the formulas $F^*(\mathbf{P}^Y_\perp)$, $F^*(\overrightarrow{X \setminus Y})$ is $\perp$. Otherwise $F^*(\mathbf{P}^Y_\perp)$ is F, while $F^*(\overrightarrow{X \setminus Y})$ is $\top$ or $\perp$ depending on whether $X \models F$.

Case 2: F is $\perp$. Each of the formulas $F^*(\mathbf{P}^Y_\perp)$, $F^*(\overrightarrow{X \setminus Y})$ is $\perp$.

Case 3: F is $G \wedge H$. Then, using the induction hypothesis,

$$\begin{aligned} X \models F^*(\overrightarrow{X \setminus Y}) \quad &\text{iff} \quad X \models G^*(\overrightarrow{X \setminus Y}) \wedge H^*(\overrightarrow{X \setminus Y}) \\ &\text{iff} \quad X \models G^*(\mathbf{P}^Y_\perp) \wedge H^*(\mathbf{P}^Y_\perp)\ . \end{aligned}$$

Case 4: F is $G \vee H$. Similar to Case 3.

Case 5: F is $G \to H$. Then, using the induction hypothesis,

$$\begin{aligned} X \models F^*(\overrightarrow{X \setminus Y}) \\ \text{iff} \quad & X \models (G^*(\overrightarrow{X \setminus Y}) \to H^*(\overrightarrow{X \setminus Y})) \wedge (G \to H) \\ \text{iff} \quad & X \models (G^*(\mathbf{P}_\perp^Y) \to H^*(\mathbf{P}_\perp^Y)) \wedge (G \to H) \\ \text{iff} \quad & X \models F^*(\mathbf{P}_\perp^Y). \end{aligned}$$

Now we are ready to complete the first part of the proof of Proposition 4. Condition (a′) is equivalent to (b′):

$$\begin{aligned} X \models \bigwedge_{Y \subseteq \mathbf{P}:\ Y \subset X} \neg F^*(\overrightarrow{Y}) \text{ iff } \quad & X \models \bigwedge_{Z \subseteq \mathbf{P}:\ Z \subseteq X,\ Z \neq \emptyset} \neg F^*(\overrightarrow{X \setminus Z}) \\ \text{iff} \quad & X \models \bigwedge_{Z \subseteq \mathbf{P}:\ Z \neq \emptyset} (\bigwedge Z \to \neg F^*(\overrightarrow{X \setminus Z})). \end{aligned}$$

It is also equivalent to (d′):

$$\begin{aligned} X \models \bigwedge_{Y \subseteq \mathbf{P}:\ Y \subset X} \neg F^*(\overrightarrow{Y}) \text{ iff } \quad & X \models \bigwedge_{Z \subseteq \mathbf{P}:\ Z \cap X \neq \emptyset} \neg F^*(\overrightarrow{X \setminus Z}) \\ \text{iff} \quad & X \models \bigwedge_{Z \subseteq \mathbf{P}} (\bigvee Z \to \neg F^*(\overrightarrow{X \setminus Z})). \end{aligned}$$

The second part of the proof is based on the following lemma:

Lemma 4 *Let X be a set of atoms that satisfies F, and let Y be any nonempty set of atoms. If $X \models \neg F^*(\mathbf{P}_\perp^L)$ for every loop $L \subseteq Y$ then $X \models \neg F^*(\mathbf{P}_\perp^Y)$.*

The proof of the lemma is omitted for lack of space. Lemma 4 implies that condition (c) in the statement of Proposition 4 is equivalent to condition (b). Finally, to prove that condition (e) is equivalent to the other conditions, observe that (d) implies (e), (e) implies (c), and (c) is equivalent to (d).

5 Conclusion

We have seen that answer sets can be characterized by a second-order propositional formula that is syntactically similar to the definition of circumscription. This idea has led us to new results on strong equivalence and on loop formulas, more general than those available in the literature. These theorems may be useful in connection with the fact that formulas more general than programs with nested expressions have applications to the problem of aggregates.

One of the attractive features of translational nonmonotonic formalisms, such as program completion and circumscription, is the ease of handling variables, both free and bound. Fixpoint definitions, on the other hand, such as the definition of answer sets in terms of reducts or the definition of an extension in default

logic, usually start with a grounding procedure of some kind whenever variables are involved. We hope that a predicate logic counterpart of transformation SM will help us formulate a theory of answer sets for programs with variables that does not rely on grounding and is, in this sense, more direct. Avoiding the use of grounding in the study of answer sets is a topic for future research.

Acknowledgements

We are grateful to Selim Erdoğan, Fangzhen Lin, Wanwan Ren and Hudson Turner for comments related to the topic of this note. This research was partially supported by the National Science Foundation under Grant IIS-0412907.

References

1. Reiter, R.: A logic for default reasoning. Artificial Intelligence **13** (1980) 81–132
2. McDermott, D., Doyle, J.: Nonmonotonic logic I. Artificial Intelligence **13** (1980) 41–72
3. Moore, R.: Semantical considerations on nonmonotonic logic. Artificial Intelligence **25** (1985) 75–94
4. McCarthy, J.: Circumscription—a form of non-monotonic reasoning. Artificial Intelligence **13** (1980) 27–39,171–172 Reproduced in [24].
5. McCarthy, J.: Applications of circumscription to formalizing common sense knowledge. Artificial Intelligence **26** (1986) 89–116 Reproduced in [24].
6. Clark, K.: Negation as failure. In Gallaire, H., Minker, J., eds.: Logic and Data Bases. Plenum Press, New York (1978) 293–322
7. Lloyd, J., Topor, R.: Making Prolog more expressive. Journal of Logic Programming **3** (1984) 225–240
8. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R., Bowen, K., eds.: Proceedings of International Logic Programming Conference and Symposium. (1988) 1070–1080
9. Reiter, R.: Circumscription implies predicate completion (sometimes). In: Proc. IJCAI-82. (1982) 418–420
10. Gelfond, M.: On stratified autoepistemic theories. In: Proc. AAAI-87. (1987) 207–211
11. Lin, F.: A Study of Nonmonotonic Reasoning. PhD thesis, Stanford University (1991)
12. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence **157** (2004) 115–137
13. Pearce, D.: A new logical characterization of stable models and answer sets. In Dix, J., Pereira, L., Przymusinski, T., eds.: Non-Monotonic Extensions of Logic Programming (Lecture Notes in Artificial Intelligence 1216), Springer-Verlag (1997) 57–70
14. Ferraris, P.: Answer sets for propositional theories.[2] Unpublished draft (2005)
15. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138** (2002) 181–234

[2] `http://www.cs.utexas.edu/users/otto/papers/proptheories.ps` .

16. Ferraris, P., Lifschitz, V.: Weight constraints as nested expressions. Theory and Practice of Logic Programming **5** (2005) 45–74
17. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2** (2001) 526–541
18. Turner, H.: Strong equivalence made easy: nested expressions and weight constraints. Theory and Practice of Logic Programming **3(4,5)** (2003) 609–622
19. Lin, F.: Reducing strong equivalence of logic programs to entailment in classical propositional logic. In: International Conference on Knowledge Representation and Reasoning (KR). (2002) 170–176
20. Lee, J., Lifschitz, V.: Loop formulas for disjunctive logic programs. In: Proc. ICLP-03. (2003) 451–465
21. Lee, J.: A model-theoretic counterpart of loop formulas. In: Proc. IJCAI. (2005) To appear.
22. Saccá, D., Zaniolo, C.: Stable models and non-determinism in logic programs with negation. In: Proceedings of Symposium on Principles of Database Systems (PODS). (1990) 205–217
23. Leone, N., Rullo, P., Scarcello, F.: Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. Information and Computation **135(2)** (1997) 69–112
24. McCarthy, J.: Formalizing Common Sense: Papers by John McCarthy. Ablex, Norwood, NJ (1990)