A New Form of Circumscription for Logic Programs

Allen Van Gelder
University of California at Santa Cruz

September 1, 1990

Abstract

Various semantics for logic programs with negation are described in terms of a dualized program together with additional axioms, some of which are second order formulas. A new axiom to represent "common sense" reasoning is proposed for logic programs. It is shown that the well-founded semantics and stable models are definable with this axiom.

1 Introduction

Although the semantics of Horn logic programs is standard, as given in the seminal work of van Emden and Kowalski [VEK76], there is presently no universally accepted semantics for logic programs with negation. The purposes of this paper are to describe various existing proposals using a common framework of classical (two-valued) logic, to thereby more clearly delineate their differences, and to introduce a new form of circumscription for logic programs and show its relationship to the recently proposed semantics based on stable models and well-founded models. We shall discuss and compare the earlier semantics proposed by Clark [Cla78], Fitting [Fit85], and Kunen [Kun87], as well as the more recent ones identified with stable models [GL88] and well-founded models [VGRS90]. We avoid the use of procedural definitions and nonstandard logics, but in some cases use second order formulas.

It has long been accepted that a logic program, particularly one with negative subgoals, carries a certain amount of implicit information. Clark was the first to formalize a method to make this implicit content explicit, by defining a *completed logic program* to be associated with the original [Cla78]. Reiter's *closed world assumption* is somewhat related [Rei78]. Independently, McCarthy proposed the concept of *circumscription* to capture the implicit "common sense" element in any first order sentence (not necessarily a logic program) that was intended roughly as a "knowledge about the world" [McC80]. Whereas Clark remained (mostly) within a first order framework, McCarthy used a second order formula. To a large extent, subsequent research has refined one of these fundamental approaches in an attempt to cure various perceived problems in circumscription [Lif85, Lif88, Lif86, KP88b, Kri88] or program completion [She85, Fit85, Kun87, Llo87, She88, Kun88]. Stratified semantics [ABW88, VG89b] and its generalizations, stable models [GL88], and well-founded semantics [VGRS90], constitute a vein that has seen considerable recent activity [Prz88, Cho88, Kol90, Prz89, Ros89, Sch88, VG89a, PP90, SZ90, Ros90, Sch90, YY90].

The study of logic programs without function symbols, usually with models required to be finite, has also seen much recent activity [CH85, GS86, Imm86, AV88, KP88a, CGKV88, Lei89, Kol90]. Such programs are often called deductive databases. This paper is not closely related to these studies, as infinite models are permitted.

We propose a new second order axiom that formalizes "common sense" differently (Section 5). Rather than minimality, it addresses lack of support. One form of this axiom leads to stable models; another leads to well-founded models. However, obtaining either stable or well-founded models, as presented in previous literature, may require an additional axiom to force the domain of interpretation to be (an isomorphic copy of) the Herbrand universe. An intriguing open question is what kind of semantics result when this domain closure axiom is dropped.

2 Preliminaries and Terminology

We assume familiarity with the common terminology of logic programming, but review some specifics, and describe our notation and basic definitions.

A *Horn* logic program may be thought of as a finite set of *rules*, exactly one for each predicate symbol, in the form:

$$q_i(\vec{x}) \leftarrow B_i(\vec{x}) \qquad i = 1, \ldots, n$$

where $\vec{x}$ is a vector of distinct variables, $q_i(\vec{x})$, called the *head* of the rule, is an atomic formula, and $B_i(\vec{x})$, called the *body* of the rule, is a positive, existential, formula of first order logic with equality. That is, the rule body is built from atomic formulas (atoms) connected by "and", "or", and existential quantifiers. The backwards implication symbol "$\leftarrow$" is conveniently read as "if". The propositional constants **true** and **false** may also be used for rule bodies. The reason for this slightly unorthodox description will soon become evident.

There are several natural ways to extend the Horn rule format to introduce nonmonotonicity. We are primarily concerned with the extension to *normal* logic programs, in the terminology of Lloyd [Llo87]. This extension permits rule bodies to be built from positive and negative literals (where Horn rules are limited to atoms, or positive literals), connected by "and", "or", and existential quantifiers; essentially, only universal quantifiers are absent. A closely related extension, *general* logic programs, permits rule bodies to be any first order formula. (Lloyd and Topor used the terms "generalized clause" and "extended clause" where we use "normal" and "general" [LT84].) A quite different road is taken with *disjunctive* logic programs, which permit the head of a rule to be a disjunction of atoms [Min82].

Some additional notational conventions: generic formulas are denoted by A, B, C, while ground (variable-free) terms are written a, b, c; generic terms are s, t, u; variables are $v, \ldots, z$. In examples, any lowercase letter may represent a predicate or a function symbol, when the syntax dictates this meaning; generic program predicates are denoted with q, p, and e. Predicate *variables* of second order formulas are written using uppercase letters R, ..., Z. The notation $\tilde{q}$ is discussed in the next section.

To streamline notation when no confusion will arise, we may drop subscripts and omit arrows over vectors. For example, the generic program rule may be abbreviated to

$$q(x) \leftarrow B(x)$$

By doing so we do not intend to limit the discussion to programs with a single unary predicate. Similarly, R and $\tilde{R}$ normally denote vectors of predicate variables, with each R_i and $\tilde{R}_i$ corresponding to q_i in arity. The "pair" $(R, \tilde{R})$ is really a $2n$-vector.

The notation $B[q/R, \tilde{q}/\tilde{R}](x)$ means the formula $B(x)$ with each q_i atom replaced by the corresponding R_i atom, and each $\tilde{q}_i$ atom replaced by the corresponding $\tilde{R}_i$ atom. In addition, in second order formulas we will use set operations like $R \subseteq B$ as shorthand for $\forall x[R(x) \rightarrow B(x)]$, which in turn is really an abbreviation for

$$\bigwedge_{i=1}^{n} \forall \vec{x_i}[R_i(\vec{x_i}) \rightarrow B_i(\vec{x_i})]$$

3 Dualized Programs

Clark introduced the idea of replacing an "if" rule by its "if and only if" counterpart. We introduce *dualized* rules and programs to achieve a somewhat weaker effect. The dualized program serves as a common point of departure for various semantics.

The first step of dualization is replace each negative literal $\neg q(t)$ in the rule body by $\tilde{q}(t)$; $\neg(t = u)$ is replaced by $t \neq u$. Since $\tilde{q}$ and $\neq$ are regarded as new predicate symbols, this actually removes all negation from the program. We call q and $=$ *positive* predicates and call $\tilde{q}$ and $\neq$ *tilde* predicates. The rules that result, denoted as a set by $\mathbf{P}$, are called the *positive* rules.

Definition 3.1: For any formula A in which negation is absent, the *dual* of A, denoted by $\tilde{A}$, is defined inductively:

1. Atoms $q(t)$ and $\tilde{q}(t)$ are duals; $(t = u)$ and $(t \neq u)$ are duals; **true** and **false** are duals.
2. The dual of $(A \wedge B)$ is $(\tilde{A} \vee \tilde{B})$; the dual of $(A \vee B)$ is $(\tilde{A} \wedge \tilde{B})$.
3. The dual of $\exists x\, A(x)$ is $\forall x\, \tilde{A}(x)$; the dual of $\forall x\, A(x)$ is $\exists x\, \tilde{A}(x)$.

The definition extends to all formulas by first pushing negations down to the atoms, then replacing $\neg q$ by $\tilde{q}$, and finally applying the above definition to construct the dual.

The *tilde rule* corresponding to positive rule $q(x) \leftarrow B(x)$ is $\tilde{q}(x) \leftarrow \tilde{B}(x)$, where $\tilde{B}(x)$ is the dual formula of $B(x)$. The set of tilde rules corresponding to $\mathbf{P}$ is denoted $\tilde{\mathbf{P}}$. $\square$

The tilde rule $\tilde{q}(x) \leftarrow \tilde{B}(x)$ can be thought of as the "only if" version of $q(x) \leftarrow B(x)$. The initial form of the "only if" rule is $q(x) \rightarrow B(x)$. However, its contrapositive form is $\neg q(x) \leftarrow \neg B(x)$, which leads to the tilde rule by pushing down negations in the rule body and replacing negative literals by tilde atoms.

Due to the introduction of universal quantifiers, it is important to rule out models in which syntactically different terms collapse to the same interpreted object. The standard way to do this is to append the Clark Equality Theory (CET), which forces syntactically different variable-free terms to be interpreted as different objects [Cla78, AVE82, Kun87, Llo87]. This is an infinite recursive set of first order sentences, often called the *equality freeness axioms*, that depend on the program. Among other things they force functions to be 1-1 with disjoint ranges ($f(x) \neq g(y)$); constants are 0-ary functions, so $a \neq f(x)$ as well. They also prevent any term being equal to a proper subterm of itself.

Definition 3.2: The *dualized program* associated with a given logic program consists of the positive and tilde rules, together with the CET axioms: $(\mathbf{P}, \tilde{\mathbf{P}}, \mathrm{CET})$. $\square$

Example 3.1: The original rules are:

$$\begin{aligned} p(x) &\leftarrow e(x) \wedge \exists y\, d(x,y) \\ d(x,y) &\leftarrow \neg(x = y) \\ e(x) &\leftarrow (x = a) \end{aligned}$$

Intuitively, d intends to express the property that x and y are distinct. The dualized program is:

$$\begin{aligned} p(x) &\leftarrow e(x) \wedge \exists y\, d(x,y) \\ d(x,y) &\leftarrow (x \neq y) \\ e(x) &\leftarrow (x = a) \end{aligned}$$

$$\begin{aligned} \tilde{p}(x) &\leftarrow \tilde{e}(x) \vee \forall y\, \tilde{d}(x,y) \\ \tilde{d}(x,y) &\leftarrow (x = y) \\ \tilde{e}(x) &\leftarrow (x \neq a) \end{aligned}$$

If the dualized program is interpreted in a universe of one element, then $\tilde{p}(a)$ must be true; whereas if it is interpreted in a larger universe, then $p(a)$ must be true. Note that neither $p(a)$ nor $\tilde{p}(a)$ is a logical consequence of the dualized program. This program is discussed further in later examples. □

Although all rule bodies are positive formulas in the dualized program, it is not generally a Horn program (aside from CET) due to the presence of universal quantification. However, it *is* an inductive system as studied by Moschovakis [Mos74]. The familiar "immediate consequence" operator

$$\phi(R, \tilde{R})(x) = B[q/R, \tilde{q}/\tilde{R}](x)$$

associated with the dualized program is monotonic and has a least fixpoint in any structure.

4 Program Completion Semantics

From the dualized program we can re-establish the connection between duals and negation with the aid of two additional first order axioms.

Definition 4.1: The *disjointness* axiom is

$$\mathit{disjoint} \stackrel{\text{def}}{=} \neg\exists x\, [q(x) \wedge \tilde{q}(x)] \equiv (q \cap \tilde{q} = \emptyset)$$

The *totality* axiom is

$$\mathit{total} \stackrel{\text{def}}{=} \forall x\, [q(x) \vee \tilde{q}(x)]$$

We remind the reader the axioms are presented in abbreviated form for programs with several predicates; the unabbreviated forms would have the appropriate conjunctions and subscripts. □

Clark proposed a semantics based on the *completed program*, which in our notation takes the form:

$$\mathit{comp}(\mathbf{P}) \stackrel{\text{def}}{=} (\mathbf{P}, \tilde{\mathbf{P}}, \text{CET}, \mathit{disjoint}, \mathit{total})$$

Sentences that are *logical consequences* of the completed program are regarded as true. For query-answering purposes, associate $\tilde{q}$ with $\neg q$.

One problem with the completed program is that it might be inconsistent. While the examples of this phenomenon, such as $p \leftarrow \neg p$, might appear silly as programs, inconsistency can arise in quite reasonable programs. For this single rule, the dualized rules are $p \leftarrow \tilde{p}$ and $\tilde{p} \leftarrow p$, which cannot satisfy *total* and *disjoint*.

However, it is clear that the disjointness axiom can cause inconsistency only in conjunction with the totality axiom. Fitting and Kunen considered variations of program completion that essentially discarded the totality axiom, although their work was presented in terms of three-valued logic. With the totality axiom gone, a little study shows that the disjointness axiom does not constrain the logical consequences, so it too may be ignored, and we are back to the dualized program $(\mathbf{P}, \tilde{\mathbf{P}}, \mathrm{CET})$. There are two important differences between the Fitting and Kunen semantics, one of which is quite subtle.

The Fitting semantics essentially limits the domain of interpretation to the Herbrand universe; it specifies that the true facts are precisely those in the least fixpoint of ϕ, the immediate consequence operator, on that universe. We now describe this semantics in terms of logical consequences of a second order formula.

Membership in the Herbrand universe can be defined inductively. For each finite set of function symbols (with constants treated as 0-ary functions) there is a rule $h(x) \leftarrow B_h(x)$ whose least fixpoint defines the Herbrand universe. The construction of B_h depends on the set of symbols, but is purely mechanical. For example, if the program has just a constant a and binary function f,

$$B_h(x) \stackrel{\text{def}}{=} (x = a) \vee \exists yz\, [(x = f(y, z)) \wedge h(y) \wedge h(z)]$$

As is well-known, the inductive closure of this rule has no first order expression, but *is* expressed by the second order formula

$$(h(x) \leftrightarrow B_h(x)) \wedge \forall R\, [(B_h[h/R] \subseteq R) \rightarrow h \subseteq R]$$

The second order part of the formula constrains h to be the least fixpoint of (the operator defined by) B_h. This leads to a domain closure axiom, which effectively restricts attention to Herbrand models.

Definition 4.2: With B_h as defined above, the *domain closure axiom* is:

$$dca \stackrel{\text{def}}{=} \forall x\, [B_h[h/true](x)] \wedge \forall R\, [(B_h[h/R] \subseteq R) \rightarrow \forall y\, R(y)]$$

□

It follows that the Fitting semantics can be defined as the logical consequences of the dualized program conjoined the above domain closure axiom: $(\mathbf{P}, \tilde{\mathbf{P}}, \mathrm{CET}, dca)$.

The well-recognized difference between the Fitting and Kunen semantics is that Fitting uses ϕ^∞, the least fixpoint, whereas Kunen uses ϕ^ω. There is another subtle difference: Kunen requires the logic program to be expressed in a language with a countably infinite set of function symbols of each arity. This has the effect of preventing the definition of domain closure! The next example shows that the Fitting and Kunen semantics are actually incomparable; neither is contained in the other.

Example 4.1: Recall the program and dualized program of Example 3.1. The atom $\tilde{p}(a)$ is true in all Herbrand models, hence is true in the Fitting semantics. However, $p(a)$ is true in all models on infinite universes, hence is true in Kunen semantics. □

It is not always clear whether requiring the program to be interpreted in a universe with infinitely many "unknown" objects agrees with the user's intentions. This question, called the *universal query problem*, is discussed by Przymusinska and Przymusinski [PP90]. In the context of normal software, it seems clear that the domain closure axiom has no place, and the existence of many unknown objects should be assumed. We want procedures to be as independent as possible of their environments; certainly, we do not want their behavior to change when unrelated procedures, containing new symbols, are added to the system.

A simple way to incorporate "unknown" objects is to form the augmented program [VGRS90]:

Definition 4.3: Given a dualized program, the associated *augmented program* is obtained by adding the rule:

$$aug \stackrel{\text{def}}{=} p\$(f\$(a\$)) \leftarrow p\$(f\$(a\$))$$

to the program, where $p\$$, $f\$$, and $a\$$, are new symbols ($p\$$ is considered neither a positive nor a tilde predicate). CET axioms are extended to include the new symbols. If domain closure (Definition 4.2) is to be used, the augmented program is formed *before* forming dca. □

It appears that the following is true, but no proof has been offered:

Conjecture 4.1: An atom in the Herbrand base of the dualized program is true in the Kunen semantics if and only if it is a logical consequence of the augmented program. □

None of the program completion semantics capture the complement of an inductive closure in a natural way. The well worn example is the complement of transitive closure on a finite directed graph [VG89b, Kun88, VGRS90]. This fact is perhaps the primary motivation for exploring other semantics. We examine some recent proposals in the next section.

5 A "Common Sense" Axiom

McCarthy observed that in everyday "common sense" reasoning, people treated a statement as false if there was no foundation for believing it to be true [McC80]. Thus from the rule "healthy birds can fly, except penguins" and the fact "Tweety is a healthy bird", the common sense conclusion is that Tweety can fly, because there is *no reason to believe* that Tweety is a penguin. He formalized this practice by adding an axiom requiring models to be minimal with respect to certain predicates.

For logic programs, we propose a different axiom, one that goes more directly to the point that there is "no reason to believe" something. Informally, a set of facts that we have "no reason to believe" are true are called *unsupported*. We intend to accept models only if all sets of unsupported facts about the *original* predicates of the program have dual facts that are true in the model. That is, if $p(a)$ is in an unsupported set, then $\tilde{p}(a)$ must be in the model. This is very close to the concept of *unfounded set* [VGRS90], but not identical. We now proceed with the formal definitions.

Definition 5.1: Let $\mathbf{P}$ be a given logic program with generic rule $q(x) \leftarrow B(x)$. The *unsupported set axiom* for U with respect to S and $\tilde{S}$ is:

$$unsup(U, S, \tilde{S}) \stackrel{\text{def}}{=} U \subseteq \tilde{B}[q/S, \tilde{q}/(\tilde{S} \cup U))]$$

In words, if we currently believe $(S, \tilde{S})$ and add U to the tilde relations, then all of U is rederivable as immediate tilde consequences. □

Unfounded sets were defined as part of the well-founded semantics [VGRS90]. They are defined within the Herbrand universe. The definition is given here in terms of the dualized program. The cited paper should be consulted for additional details and motivation. The relationship to unsupported sets is illustrated in the lemma that follows the definitions.

Definition 5.2: Without loss of generality, we assume that each rule body is in the form

$$B_i(x) \stackrel{\text{def}}{=} \bigvee_j \exists y_j \bigwedge_k L_{jk}(x, y_j)$$

where the L_{jk} are atoms. A vector of relations U_i, $1 \leq i \leq n$, is an *unfounded set* with respect to $(S, \tilde{S})$ if for each ground Herbrand rule instantiation

$$B_i(a) = \bigvee_j \bigwedge_k L_{jk}(a, b_j)$$

such that $a \in U_i$, for each disjunct j there is a so-called *witness of unusability* for some $L_{jk}(a, b_j)$ with one of these properties:

1. If $L_{jk}(a, b_j) = q_m(c)$, a positive atom, then $\tilde{q}_m(c) \in \tilde{S}$ is a witness of unusability; similarly, if $L_{jk}(a, b_j) = \tilde{q}_m(c)$, a tilde atom, then $q_m(c) \in S$ is a witness of unusability.
2. If $L_{jk}(a, b_j) = q_m(c)$, a positive atom, then $c \in U_m$ is a witness of unusability.

In words, for rule body $B_i(a)$ to be true (interpreted in the Herbrand universe), it would be necessary that some pair of atoms $(q_m(c), \tilde{q}_m(c))$ be true (condition 1), or that some atom of U be true (condition 2).

The mapping $\tilde{\mathbf{U}}(S, \tilde{S})$ is defined by

$$\begin{aligned} \tilde{\mathbf{U}}(S, \tilde{S}) \stackrel{\text{def}}{=} \{\tilde{q}_i(a) \mid a \in U_i \text{ such that} \\ U \text{ is unfounded w.r.t. } (S, \tilde{S})\} \end{aligned}$$

Note that this maps a pair of vectors of relations into a vector of tilde relations.

The *well-founded transformation* $\mathbf{W}(S, \tilde{S})$ combines the positive immediate consequences with $\tilde{\mathbf{U}}$:

$$\mathbf{W}(S, \tilde{S}) \stackrel{\text{def}}{=} (B[q/S, \tilde{q}/\tilde{S}],\ \tilde{\mathbf{U}}(S, \tilde{S}))$$

and its least fixpoint gives the *well-founded dualized model* [VGRS90]. □

Observe that, if U is unfounded w.r.t. $(S, \tilde{S})$, then no element of U can be the *first* element of U to be derived in a positive relation as long as the positive and tilde relations that represent the current "set of beliefs" remain disjoint and are supersets of $(S, \tilde{S})$. Furthermore, this remains true even after the tuples of U are added to the tilde relations. In this sense, it is "safe" to put U's tuples in the tilde relations.

What is called the "well-founded partial model" in the original nomenclature is called the "well-founded dualized model" here because it is actually a *model* of the dualized program, but may not satisfy the totality axiom, *total*. One theorem we shall use is that stable models are precisely the fixpoints of $\mathbf{W}$ that satisfy *total* [VGRS90].

The next lemma shows that unsupported sets are essentially a generalization of unfounded sets to arbitrary domains.

Lemma 5.1: Let *dca*, the domain closure axiom, hold. Then U is an unfounded set with respect to $(S, \tilde{S})$ if and only if U is an unsupported set with respect to $(S, \tilde{S})$.

Proof: This is a matter of checking the definitions. Let a rule body be

$$B_i(x) = \bigvee_j \exists y_j \bigwedge_k L_{jk}(x, y_j)$$

Its dual is

$$\tilde{B}_i(x) = \bigwedge_j \forall y_j \bigvee_k \tilde{L}_{jk}(x, y_j)$$

Suppose U is unfounded. The domain closure axiom forces $\forall y_j$ in $\tilde{B}_i$ to range only over ground Herbrand terms. For $x = a$ and $y_j = b_j$, the witness of unusability causes some $\tilde{L}_{jk}[q/S, \tilde{q}/(\tilde{S} \cup U)]$ to evaluate to true, and there is one witness for each conjunct j.

Suppose U is unsupported. Then for each $q_i(a) \in U$, the body $\tilde{B}_i(a)$ must be true, so each conjunct j must be true, so for all terms b_j some literal $\tilde{L}_{jk}[q/S, \tilde{q}/(\tilde{S} \cup U)](a, b_j)$ must be true. Whichever literal is true is the witness of unusability. For example, if the literal is $\tilde{q}_m(c)$ and $c \in U_m$, it meets condition 2; other cases meet condition 1. ∎

Now we can formulate an axiom to require that all unsupported sets are contained in corresponding tilde relations. This is the "common sense" axiom that we believe captures the intention of circumscription more accurately than the usual relation-minimization axioms.

Definition 5.3: Let **P** be a given logic program with generic rule $q(x) \leftarrow B(x)$. The *well-supported set axiom* is:

$$\mathit{wellsup}(R, \tilde{R}) \stackrel{\text{def}}{=} \forall U\, [\mathit{unsup}(U, R, \tilde{R})) \rightarrow (U \subseteq \tilde{R})]$$

In words, for a "set of beliefs" $(R, \tilde{R})$ to be well-supported, any unsupported set U with respect to $(R, \tilde{R})$ must be contained in $\tilde{R}$. □

We can now state certain relationships between the *wellsup* axiom and stable and well-founded models.

Theorem 5.2: Let a dualized (possibly augmented) program be given, and append the domain closure axiom, and the axiom $\mathit{wellsup}(q, \tilde{q})$. The logical consequences of the resulting set of axioms, which is $(\mathbf{P}, \tilde{\mathbf{P}}, \mathrm{CET}, \mathit{wellsup}(q, \tilde{q}), \mathit{dca})$, agree with the well-founded dualized model in the sense that a ground atom is in the well-founded dualized model if and only if it is a logical consequence of the axioms.

Proof: The domain closure axiom allows us to restrict attention to Herbrand models. Suppose $(R, \tilde{R})$ is the well-founded dualized model. By its definition [VGRS90], $\tilde{R}$ is the union of all sets U that are unfounded with respect to $(R, \tilde{R})$. By Lemma 5.1 all such U are unsupported, so the axiom $\mathit{wellsup}(R, \tilde{R})$ holds. Thus the well-founded dualized model is an upper bound on the set of atoms that are logical consequences. Let $(S, \tilde{S})$ be any Herbrand model that satisfies $\mathit{wellsup}(q, \tilde{q})$. Being a model, we have $B(S, \tilde{S}) \subseteq S$, and by the *wellsup* axiom $\tilde{\mathbf{U}}(S, \tilde{S}) \subseteq \tilde{S}$ (recall Definition 5.2 for $\tilde{\mathbf{U}}$ and $\mathbf{W}$). Thus $(S, \tilde{S})$ is a prefixpoint of $\mathbf{W}$, so is a superset of the well-founded dualized model. ∎

Theorem 5.3: Let a dualized program be given, and append the domain closure axiom, the totality axiom, the disjointness axiom, and the axiom $\mathit{wellsup}(q, \tilde{q})$. The models of the resulting set of axioms, which is

$(\mathbf{P}, \tilde{\mathbf{P}}, \mathrm{CET}, \textit{disjoint}, \textit{total}, \textit{wellsup}(q, \tilde{q}), \textit{dca})$, are precisely the stable models of the original program (with $\tilde{q}$ corresponding to $\neg q$).

Proof: It is sufficient to prove that all models are fixpoints of $\mathbf{W}$, and all fixpoints of $\mathbf{W}$ that satisfy totality and disjointness are models. If $(R, \tilde{R})$ is a fixpoint of $\mathbf{W}$, the *wellsup* axiom holds, as argued in the previous theorem. It easily follows that $(R, \tilde{R})$ is a model of the whole formula.

Now suppose $(R, \tilde{R})$ is a model, implying that $B[q/R, \tilde{q}/\tilde{R}] \subseteq R$. If $U \stackrel{\text{def}}{=} R - B[q/R, \tilde{q}/\tilde{R}]$ is nonempty, then we claim it is an unsupported set w.r.t. $(S, \tilde{S})$. By totality and disjointness, $\tilde{B}$ is true just where B is false (and *vice versa*), so $U \subseteq \tilde{B}[q/R, \tilde{q}/\tilde{R}]$. So Definition 5.1 holds by monotonicity of $\tilde{B}$, and the claims follows. Consequently, $B[q/R, \tilde{q}/\tilde{R}] = R$ (or disjointness would be violated). Similarly, $\textit{wellsup}(R, \tilde{R})$ implies that $\tilde{\mathbf{U}}(R, \tilde{R}) \subseteq \tilde{R}$. If it is a proper subset, there is a nonempty set $S \subseteq \tilde{R}$ such that $S \subseteq B[q/R, \tilde{q}/\tilde{R}] = R$ as well, again violating disjointness. So $\tilde{\mathbf{U}}(R, \tilde{R}) = \tilde{R}$, and $(R, \tilde{R})$ is a fixpoint of $\mathbf{W}$. ▮

Perhaps the most important observation about these theorems is that they add the domain closure axiom *dca* to ensure the connection between the logical consequences and the existing model-based definitions in the literature. However, *dca* seems to be unrelated to the common sense and autoepistemic properties that the *wellsup* axiom is attempting to enforce, so it is natural to wonder if it is necessary or even desirable. It was included in the hypotheses of the theorems to facilitate the proofs.

The next examples demonstrate that the logical consequences of $(\mathbf{P}, \tilde{\mathbf{P}}, \mathrm{CET}, \textit{wellsup}(q, \tilde{q}), \textit{aug})$ (without *dca*) often characterize the well-founded dualized model. Here q denotes the vector of predicates that actually occur in the example.

Example 5.1: Consider the dualized program

$$\begin{aligned}
p(x) &\leftarrow (x = s(a)) \vee \exists y\,[(x = s(y) \wedge p(y)] \\
n(x) &\leftarrow \tilde{p}(x) \\
d &\leftarrow \exists y\,[p(y) \wedge n(s(y))] \\
\tilde{p}(x) &\leftarrow (x \neq s(a)) \wedge \forall y\,[(x \neq s(y) \vee \tilde{p}(y)] \\
\tilde{n}(x) &\leftarrow p(x) \\
\tilde{d} &\leftarrow \forall y\,[\tilde{p}(y) \vee \tilde{n}(s(y))]
\end{aligned}$$

Intuitively, p identifies successors of a, where s denotes immediate successor; n identifies the complement of p; and d "asks" whether there is a y that is a successor of a such that $s(y)$ is not a successor of a; we expect the answer to be "no", but let us see.

Clearly, $p(s^k(a))$ are logical consequences, for $k \geq 1$, so $\tilde{n}(s^k(a))$ are, as well. Also, CET implies $\forall y\,[a \neq s(y)]$, so $\tilde{p}(a)$ and $n(a)$ are logical consequences. (Augmenting the program has no significant effect here, as $\tilde{p}(t)$ is derived for any term t containing augmentation symbols.) The minimum Herbrand model excludes d and other ground atoms for p, $\tilde{p}$, n, $\tilde{n}$, besides those mentioned. Only the status of $\tilde{d}$ remains in doubt: it holds in all Herbrand models, but what about others?

If we add a copy of the integers (positive and negative) to the Herbrand universe, and interpret s as normal successor on them, we can construct a model in which $\tilde{d}$ is false. We make $p(x)$ and $\tilde{p}(x)$ false for all the integers (possible because every integer is the successor of some integer).

Kunen's theorem [Kun87, Theorem 6.3] also tells us that $\tilde{d}$ is not a logical consequence: for each finite stage k there are some y's for which $\tilde{n}(s(y))$ has not been derived yet, so $\tilde{d}$ is not in ϕ^{ω}.

If we add the axiom $wellsup(q, \tilde{q})$, then $\tilde{d}$ is a logical consequence. To see this, let $U = (U_p, U_n, U_d)$, where $U_p(x)$ holds for all "non-Herbrand" elements, U_n and U_d are empty. Then U_p is unsupported w.r.t. $(\emptyset, \emptyset)$, as can be verified by substituting U_p for $\tilde{p}$ in the body of the rule for $\tilde{p}$. Thus $\tilde{p}$ and n must hold for all "non-Herbrand" elements of any model, which is sufficient to force $\tilde{d}$ to hold.

This development follows a standard pattern: p is defined by a positive existential induction, so $wellsup(q, \tilde{q})$ forces everything not in the inductive closure to be in $\tilde{p}$, *including non-Herbrand elements.*

To summarize, $\tilde{d}$ does not hold in the Kunen semantics, but does hold in the well-founded semantics without requiring *dca*. It holds in the Fitting semantics just because of *dca*, and it holds in the Clark semantics because of *total*. The program is stratified, so the well-founded dualized model is the unique stable model [VGRS90]. □

Example 5.2: This example examines an induction that is positive, but contains universal quantification. To introduce $\forall z$ it is necessary in our syntax to "transfer" the rule for w through $\tilde{u}$.

$$\begin{aligned}
w(x) &\leftarrow \tilde{u}(x) \\
u(x) &\leftarrow \exists z\, [e(z, x) \wedge \tilde{w}(z)] \\
p(x) &\leftarrow (x = s(a)) \vee \exists y\, [(x = s(y) \wedge p(y)] \\
e(x, y) &\leftarrow p(x) \wedge ((y = s(x)) \vee (y = a)) \\
\tilde{w}(x) &\leftarrow u(x) \\
\tilde{u}(x) &\leftarrow \forall z\, [\tilde{e}(z, x) \vee w(z)] \\
\tilde{p}(x) &\leftarrow (x \neq s(a)) \wedge \forall y\, [(x \neq s(y) \vee \tilde{p}(y)] \\
\tilde{e}(x, y) &\leftarrow \tilde{p}(x) \vee ((y \neq s(x)) \wedge (y \neq a))
\end{aligned}$$

Intuitively, e is an infinite directed graph in which every node $s^k(a)$, $k \geq 1$, has an edge to $s^{k+1}(a)$ and an edge to a. The rule for p is the same as in the previous example, identifying the nodes that have out-edges. The predicate w is intended to define the well-founded nodes of this graph, that is, the nodes from which there is no infinite descending chain.

The question is whether $w(a)$ is a logical consequence of this dualized program. By stages, we "discover" $w(s(a))$, $w(s^2(a))$, …, but at each finite stage there is some $z = s^k(a)$ such that $w(z)$ has not been discovered yet. Clearly, neither does $\tilde{e}(z, a)$ hold for this z, so $\tilde{u}(a)$ and $w(a)$ are not derived at any finite stage. It follows by Kunen's theorem that $w(a)$ is not a logical consequence.

Again, a counter model is constructed by adding the integers to the Herbrand universe, with s the normal immediate successor function. As in the previous example, it is consistent to make $\tilde{p}$ false on all non-Herbrand elements. This done, $\tilde{e}(x, a)$ and $\tilde{e}(x, s(x))$ may be false for all non-Herbrand x. So $\tilde{u}$ and w may be made false for all non-Herbrand elements and for a. It is possible for this model to satisfy *total* by just adding the necessary positive atoms.

Just as in the previous example, the addition of the axiom $wellsup(q, \tilde{q})$ forces $\tilde{p}(x)$ to hold for non-Herbrand x in all models, and so $\tilde{e}(x, a)$ holds for all such x. It follows in a few steps that $w(a)$ is a logical consequence. As before, augmentation makes no material difference.

To summarize, $w(a)$ is not supported by Kunen's semantics or Clark's semantics, as non-Herbrand models exist in which it is false. The addition of *dca* makes it true in Fitting's semantics, and $wellsup(q, \tilde{q})$ without *dca* also does the job. The program is not stratified, but the well-founded dualized model satisfies *total*, so it is the unique stable model. □

Example 5.3: The next program is one in which the augmentation rule *does* make a difference; it is a more elaborate version of the dualized program of Example 3.1:

$$\begin{aligned}
p(x) &\leftarrow e(x) \wedge \exists v\, d(v,x) \\
d(v,x) &\leftarrow \tilde{g}(v,x) \\
g(v,x) &\leftarrow (v = x) \vee g(v, s(x)) \\
e(x) &\leftarrow (x = a)
\end{aligned}$$

$$\begin{aligned}
\tilde{p}(x) &\leftarrow \tilde{e}(x) \vee \forall v\, \tilde{d}(v,x) \\
\tilde{d}(v,x) &\leftarrow g(v,x) \\
\tilde{g}(v,x) &\leftarrow (v \neq x) \wedge \tilde{g}(v, s(x)) \\
\tilde{e}(x) &\leftarrow (x \neq a)
\end{aligned}$$

Intuitively, $g(v,x)$ means $v = s^k(x)$ for some $k \geq 0$. Thus $g(v,a)$ and $\tilde{d}(v,a)$ are true for all v in the Herbrand universe of the unaugmented program. As before, neither $p(a)$ nor $\tilde{p}(a)$ is a logical consequence of just $(\mathbf{P}, \tilde{\mathbf{P}}, \mathrm{CET})$. Addition of $\mathit{wellsup}(q, \tilde{q})$ does not change things: $p(a)$ holds in non-Herbrand models and $\tilde{p}(a)$ holds in Herbrand models. However, if *dca* is added (with or without $\mathit{wellsup}(q, \tilde{q})$), then $\tilde{p}(a)$ holds in all models. This demonstrates that *dca* can make a difference to an *unaugmented* program with the *wellsup* axiom. We shall see that augmentation nullifies this difference.

In the augmented program, it turns out that one extra symbol is enough; let us denote it \$ for simplicity. (Alternatively, think of \$ as representing $a\$$ or any ground term whose principal functor is $f\$$.)

With *aug* added, $\tilde{p}(a)$ is not a logical consequence, even if *dca* is also added. But what about $p(a)$? We observe that the rule for $\tilde{g}$ is recursive with no base case, so no atoms for $\tilde{g}$ or d will ever be derived (even transfinitely), and neither will $p(a)$.

This is another case where the complement of the inductive closure does not show up in the tilde relation without the "encouragement" of $\mathit{wellsup}(q, \tilde{q})$. The unsupported set of interest is $U_g(\$, s^k(a))$ for $k \geq 0$. Substituting this for $\tilde{g}$ in its own rule body verifies that the resulting formula is true for all tuples $(\mathbf{S}, s^k(a))$, as required. So $\tilde{g}(\$, s^k(a))$ for $k \geq 0$ is required to hold in all models that satisfy $\mathit{wellsup}(q, \tilde{q})$; in particular, $\tilde{g}(\$, a)$, $d(\$, a)$, and $p(a)$ hold. This outcome did not require *dca*.

This time the semantics of Clark, Fitting, and Kunen agree that $p(a)$ is not supported in the augmented program. Only the well-founded semantics finds $p(a)$ to be true. The program is stratified, so the well-founded dualized model is the unique stable model. $\square$

In all the examples we have studied, if *aug* and $\mathit{wellsup}(q, \tilde{q})$ are added to the dualized program, then *dca* has no effect on which Herbrand atoms are logical consequences. We conjecture that this is always the case.

Conjecture 5.4: Let a dualized, augmented program be given, and append the axiom $\mathit{wellsup}(q, \tilde{q})$. The logical consequences of the resulting set of axioms, which is $(\mathbf{P}, \tilde{\mathbf{P}}, \mathrm{CET}, \mathit{wellsup}(q, \tilde{q}), \mathit{aug})$, agree with the well-founded dualized model in the sense that a ground atom is in the well-founded dualized model if and only if it is a logical consequence of the axioms. $\square$

6 Future Work

The natural (and open) questions concern variations of well-founded and stable semantics that are less closely tied to the Herbrand universe:

1. What sort of semantics are given by the logical consequences of the dualized program plus $\mathit{wellsup}(q, \tilde{q})$ without the additional restriction enforced by *dca*? In most practical cases, we expect this to be the same as the well-founded semantics.

2. Besides removing *dca*, suppose we augment the program or require an infinite "unknown" domain in analogy with the difference between Fitting and Kunen semantics? Again, we expect this to be usually the same as the well-founded semantics of the augmented program. In fact, we have found no example where *dca* makes a difference on an augmented program.

3. The corresponding questions about stable semantics.

4. In general, semantics with second order axioms are not recursively enumerable. Can reasonable classes of programs be identified that allow the second order axiom to be replaced by a first order axiom, and thereby drop the complexity to r.e.? This question has been studied in connection with traditional circumscription [Lif85, Kri88, KP88b].

5. Procedurally, it seems necessary to label atoms "undefined" in the course of query answering in the well-founded semantics [Ros89]. Should the language of logic programming be extended to allow the user to state $\mathit{undefined}(p(x))$ explicitly as a goal?

Acknowledgements

This attempt to express logic programming semantics in terms of classical logic was motivated by Robert Kowalski's expressed reservations about nonstandard logics and procedural definitions. (However, he has in no way endorsed this effort.) Discussions with Phokion Kolaitis have been helpful to the author. This research was supported in part by NSF grants CCR-8958590 and IRI-8902287.

References

[ABW88] K. R. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, Los Altos, CA, 1988.

[AV88] S. Abiteboul and V. Vianu. Procedural and declarative database update languages. In *ACM Symposium on Principles of Database Systems*, pages 240–250, 1988.

[AVE82] K. R. Apt and M. H. Van Emden. Contributions to the theory of logic programming. *JACM*, 29(3):841–862, 1982.

[CGKV88] S. S. Cosmadakis, H. Gaifman, P. C. Kanellakis, and M. Y. Vardi. Decidable optimization problems for database logic programs. In *ACM Symposium on Theory of Computing*, 1988.

[CH85] A. Chandra and D. Harel. Horn clause queries and generalizations. *Journal of Logic Programming*, 2(1):1–15, 1985.

[Cho88] P. Cholak. Post correspondence problem and Prolog programs. Technical report, Univ. of Wisc., Madison, 1988. (manuscript).

[Cla78] K. L. Clark. Negation as failure. In Gallaire and Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, New York, 1978.

[Fit85] M. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Fifth Int'l Conf. Symp. on Logic Programming*, pages 1070–1080, Seattle, 1988.

[GS86] Y. Gurevich and S. Shelah. Fixed-point extensions of first order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.

[Imm86] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68(1):86–104, 1986.

[Kol90] P. G. Kolaitis. The expressive power of stratified programs. *Information and Computation*, 1990. (to appear; also available as UCSC-CRL-89-14 from UC Santa Cruz).

[KP88a] P. G. Kolaitis and C. H. Papadimitriou. Why not negation by fixpoint? In *ACM Symposium on Principles of Database Systems*, pages 231–239, 1988.

[KP88b] Phokion G. Kolaitis and Christos H. Papadimitriou. Some computational aspects of circumscription. In *AAAI Conference on Artificial Intelligence*, pages 465–469, 1988.

[Kri88] T. Krishnaprasad. On the computability of circumscription. *Information Processing Letters*, 27:237–243, 1988.

[Kun87] K. Kunen. Negation in logic programming. *Journal of Logic Programming*, 4(4):289–308, 1987.

[Kun88] K. Kunen. Some remarks on the completed database. Technical Report 775, Univ. of Wisconsin, Madison, WI 53706, 1988. (Abstract appeared in 5th Int'l Conf. Symp. on Logic Programming, Seattle, Aug. 1988).

[Lei89] D. Leivant. Inductive definitions over finite structures. *Information and Computation*, 1989. (to appear). Also available from author as CMU–CS–89–153.

[Lif85] V. Lifschitz. Computing circumscription. In *Ninth Int'l Joint Conference on AI*, pages 121–127, 1985.

[Lif86] V. Lifschitz. Pointwise circumscription: prelininary report. In *AAAI Conference on Artificial Intelligence*, pages 406–410, 1986.

[Lif88] V. Lifschitz. On the declarative semantics of logic programs with negation. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 177–192. Morgan Kaufmann, Los Altos, CA, 1988.

[Llo87] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, New York, 2nd edition, 1987.

[LT84] J. W. Lloyd and R. W. Topor. Making Prolog more expressive. *Journal of Logic Programming*, 1(3):225–240, 1984.

[McC80] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1):27–39, 1980.

[Min82] J. Minker. On indefinite databases and the closed world assumption. In *Sixth Conference on Automated Deduction*, pages 292–308, New York, 1982. Springer-Verlag.

[Mos74] Y. N. Moschovakis. *Elementary Induction on Abstract Structures.* North-Holland, New York, 1974.

[PP90] H. Przymusinska and T. Przymusinski. Semantic issues in deductive databases and logic programs. In R. Banerji, editor, *Formal Approaches to Artificial Intelligence: A Sourcebook.* North-Holland, New York, 1990.

[Prz88] T. C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, Los Altos, CA, 1988.

[Prz89] T. C. Przymusinski. Every logic program has a natural stratification and an iterated fixed point model. In *Eighth ACM Symposium on Principles of Database Systems*, pages 11–21, 1989.

[Rei78] R. Reiter. On closed world databases. In Gallaire and Minker, editors, *Logic and Databases*, pages 55–76. Plenum Press, New York, 1978.

[Ros89] K. A. Ross. A procedural semantics for well-founded negation in logic programs. In *Eighth ACM Symposium on Principles of Database Systems*, pages 22–33, 1989.

[Ros90] K. A. Ross. Modular stratification and magic sets for Datalog programs with negation. In *Ninth ACM Symposium on Principles of Database Systems*, pages 161–171, 1990.

[Sch88] J. S. Schlipf. Choosing inference rules for logic programming. Technical report, Univ. of Cincinnati, 1988. (manuscript).

[Sch90] J. S. Schlipf. The expressive powers of the logic programming semantics. In *Ninth ACM Symposium on Principles of Database Systems*, pages 196–204, 1990.

[She85] J. C. Shepherdson. Negation as failure, II. *Journal of Logic Programming*, 2(3):185–202, 1985.

[She88] J. C. Shepherdson. Negation in logic programming. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 19–88. Morgan Kaufmann, Los Altos, CA, 1988.

[SZ90] D. Saccà and C. Zaniolo. Partial models, stable models and non-determinism in logic programs with negation. Technical report, MCC, Austin, TX, January 1990. (Extended abstract appeared in Ninth ACM Symposium on Principles of Database Systems, 1990.).

[VEK76] M. H. Van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *JACM*, 23(4):733–742, 1976.

[VG89a] A. Van Gelder. The alternating fixpoint of logic programs with negation. In *Eighth ACM Symposium on Principles of Database Systems*, pages 1–10, 1989. Full version available from UC Santa Cruz as UCSC-CRL-89-39.

[VG89b] A. Van Gelder. Negation as failure using tight derivations for general logic programs. *Journal of Logic Programming*, 6(1):109–133, 1989. Preliminary versions appeared in *Third IEEE Symp. on Logic Programming* (1986), and *Foundations of Deductive Databases and Logic Programming*, J. Minker, ed., Morgan Kaufmann, 1988.

[VGRS90] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 1990. (to appear). Available from first author as UCSC-CRL-89-38. Preliminary abstract appeared in Seventh ACM Symposium on Principles of Database Systems, 1988.

[YY90] J.-H. You and L. Y. Yuan. Three-valued formalization of logic programming: Is it needed? In *Ninth ACM Symposium on Principles of Database Systems*, pages 196–204, 1990.