A KRIPKE-KLEENE SEMANTICS FOR LOGIC PROGRAMS*

MELVIN FITTING

1. INTRODUCTION

The use of conventional classical logic is misleading for characterizing the behavior of logic programs because a logic program, when queried, will do one of *three* things: succeed with the query, fail with it, or not respond because it has fallen into infinite backtracking. In [7] Kleene proposed a three-valued logic for use in recursive function theory. The so-called third truth value was really *undefined*: truth value not determined. This logic is a useful tool in logic-program specification, and in particular, for describing models. (See [11].)

Tarski showed that formal languages, like arithmetic, cannot contain their own truth predicate because one could then construct a paradoxical sentence that effectively asserts its own falsehood. Natural languages do allow the use of "is true", so by Tarski's argument a semantics for natural language must leave truth-value gaps: some sentences must fail to have a truth value. In [8] Kripke showed how a model having truth-value gaps, using Kleene's three-valued logic, could be specified. The mechanism he used is a familiar one in program semantics: consider the least fixed point of a certain monotone operator. But that operator must be defined on a space involving three-valued logic, and for Kripke's application it will not be continuous.

We apply techniques similar to Kripke's to logic programs. We associate with each program a monotone operator on a space of three-valued logic interpretations, or better *partial interpretations*. This space is not a complete lattice, and the operators are not, in general, continuous. But least and other fixed points do exist. These fixed points are shown to provide suitable three-valued program models. They relate closely to the least and greatest fixed points of the operators used in [1].

Because of the extra machinery involved, our treatment allows for a natural consideration of negation, and indeed, of the other propositional connectives as well. And because of the elaborate structure of fixed points available, we are able to

Address correspondence to Melvin Fitting, Department of Mathematics and Computer Science, Herbert H. Lehman College, Bedford Park Boulevard West, Bronx, NY 10468.

*Work partially supported by NSF grant DCR-8304959.

THE JOURNAL OF LOGIC PROGRAMMING

52 Vanderbilt Ave., New York, NY 10017 0743-1066/85/$03.30

clearly differentiate between programs that "behave" the same but that we "feel" are different.

Finally, we show the result is far too powerful. We can now write logic programs semantically characterizing the Π^1_1 relations, not just the recursively enumerable ones. Thus semantic behavior is not generally machine realizable. We take this as an argument for imposing restrictions on logic programs, to weed out the "too powerful" ones.

2. KLEENE'S THREE-VALUED LOGIC

We use a language L in which we have the propositional connectives $\wedge$ (and), $\vee$ (or), and $\neg$ (not) as primitive. Alternatively we could take some as primitive and define others via the usual definitions, which work even in Kleene's three-valued logic. We also allow quantifiers $\forall$ and $\exists$, taking both as primitive. Quantifiers will not be used in the actual writing of programs, only in their analysis.

We systematically use *statement* for formula with no free variables. For a formula P, and a substitution θ, $P\theta$ is a *closed instance* of P if it is a statement. We sometimes write $P(x)$ to indicate that x is the only free variable of P, and $P(t)$ for the closed instance resulting from substituting t for x.

The idea behind Kleene's logic is simple. Say P is a statement with the informal meaning $f(3) = 5$, where f is a function we have a Turing machine to compute. Similarly say Q means $g(4) = 7$, where g is another Turing calculable function. By running the Turing machines we may determine the truth or falsity of P and Q, or we may never learn anything if the machines fail to halt. So, use a logic with truth values **t** (true), **f** (false), and **u** (undefined or undetermined) to mirror the situation.

In this logic, how should values be assigned to $P \wedge Q$, say? Certainly in cases not involving **u**, truth values should behave classically. Now say P has value **f** but Q is **u**. Still Q "has" a truth value; we just don't know what it is. Since P is **f**, $P \wedge Q$ will be **f** no matter whether Q turns out to be **t** or **f**. Consequently $P \wedge Q$ is given the value **f** in this case. On the other hand, say P has value **t** but Q is **u**. If we eventually discover that Q is **t**, $P \wedge Q$ will turn out to have value **t**. But if we discover Q is **f**, $P \wedge Q$ will be **f**. Given no further information then, we must say the value of $P \wedge Q$ is **u**.

Then a table for the connective $\wedge$ is as follows:

$\wedge$	**t**	**f**	**u**
t	**t**	**f**	**u**
f	**f**	**f**	**f**
u	**u**	**f**	**u**

Tables for the other connectives can be easily constructed (see [7]). But we prefer a somewhat different approach.

3. SATURATED SETS

Rather than working with truth functions, Hintikka and others have popularized the use of sets of statements, model sets. Being in the set corresponds to being true; being out, to false. We extend this to the three-valued case by using Smullyan's

device of *prefixed* formulas [13]. In effect we treat both known-to-be-true and known-to-be-false as positive information, and undefined as absence of information. What follows is essentially taken from [6].

We introduce two new symbols, T and F. If X is a formula, TX and FX are *signed* formulas. If S is a set of signed statements, we informally think about it in the following way. $TX \in S$ means S says X is true. $FX \in S$ means S says X is false. If neither $TX \in S$ nor $FX \in S$, S says nothing about the truth value of X, or X has truth value **u**. Of course we do not want both TX and FX in S then. We introduce the following terminology.

Definitions. Let S be a set of signed statements.

(1) S is *downward saturated* if

(a) $TX \wedge Y \in S \Rightarrow TX \in S$ and $TY \in S$,
(b) $FX \wedge Y \in S \Rightarrow FX \in S$ or $FY \in S$,
(c) $TX \vee Y \in S \Rightarrow TX \in S$ or $TY \in S$,
(d) $FX \vee Y \in S \Rightarrow FX \in S$ and $FY \in S$,
(e) $T\neg X \in S \Rightarrow FX \in S$,
(f) $F\neg X \in S \Rightarrow TX \in S$,
(g) $T(\forall x)P(x) \in S \Rightarrow TP(t) \in S$ for every closed term t,
(h) $F(\forall x)P(x) \in S \Rightarrow FP(t) \in S$ for some closed term t,
(i) $T(\exists x)P(x) \in S \Rightarrow TP(t) \in S$ for some closed term t,
(j) $F(\exists x)P(x) \in S \Rightarrow FP(t) \in S$ for every closed term t.

(2) S is *upward saturated* if

(a) $TX \in S$ and $TY \in S \Rightarrow TX \wedge Y \in S$,
(b) $FX \in S$ or $FY \in S \Rightarrow FX \wedge Y \in S$,
(c) $TX \in S$ or $TY \in S \Rightarrow TX \vee Y \in S$,
(d) $FX \in S$ and $FY \in S \Rightarrow FX \vee Y \in S$,
(e) $FX \in S \Rightarrow T\neg X \in S$,
(f) $TX \in S \Rightarrow F\neg X \in S$,
(g) $TP(t) \in S$ for every closed term $t \Rightarrow T(\forall x)P(x) \in S$,
(h) $FP(t) \in S$ for some closed term $t \Rightarrow F(\forall x)P(x) \in S$,
(i) $TP(t) \in S$ for some closed term $t \Rightarrow T(\exists x)P(x) \in S$,
(j) $FP(t) \in S$ for every closed term $t \Rightarrow F(\exists x)P(x) \in S$.

(3) S is *saturated* if S is both downward and upward saturated.

(4) S is *consistent* if not both TX, FX are in S, for any statement S.

(5) S is *atomically consistent* if not both TA, FA are in S for any atomic statement A.

(6) S is *complete* if either TX or FX is in S for every statement X.

(7) S is *atomically complete* if either TA or FA is in S for every atomic statement A.

(8) S is a *model set* if S is saturated, consistent and complete.

Saturated, consistent sets correspond exactly to valuations in Kleene's three-valued logic, in the following sense. Suppose S is a saturated, consistent set of signed

statements. Define a map v from statements to $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ as follows.

$$v(X) = \begin{cases} \mathbf{t} & \text{if } TX \in S, \\ \mathbf{f} & \text{if } FX \in S, \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

v is a valuation in the Kleene three valued sense. Conversely, if we start with such a valuation v, we can easily construct the corresponding saturated, consistent set S. In the rest of this paper we will work exclusively with saturated, consistent sets, rather than with three-valued mappings.

Note that adding a completeness requirement, that is, restricting consideration to model sets, gives us the classical two-valued mappings.

Proposition 3.1. Let S be a set of signed statements. There is a smallest upward saturated set extending S.

PROOF. Let $\mathbf{C}$ be the collection of upward saturated sets extending S. $\mathbf{C}$ is not empty, since it contains the set of all signed statements. So $\bigcap\mathbf{C}$ exists. It is easy to check that it is upward saturated. It fulfills the other considerations, by definition. □

Definition. For a set S of signed statements, the smallest upward saturated set extending S is called the *upward saturated closure* of S, and is denoted S^U.

Proposition 3.2. Let A and B be sets of signed statements. Then

$$A \subseteq B \quad \Rightarrow \quad A^U \subseteq B^U.$$

PROOF. Let $\mathbf{C}_A$ be the collection of upward saturated sets extending A, and similarly for $\mathbf{C}_B$. If $A \subseteq B$ then $\mathbf{C}_B \subseteq \mathbf{C}_A$; hence $\bigcap\mathbf{C}_A \subseteq \bigcap\mathbf{C}_B$. But $A^U = \bigcap\mathbf{C}_A$ and $B^U = \bigcap\mathbf{C}_B$. □

Proposition 3.3. If S is downward saturated, S^U is saturated.

PROOF. It is enough to show S^U is downward saturated. Say TX is not in S^U; we show $TX \wedge Y$ is not in S^U. The other cases are similar.

If TX is not in S^U, we cannot have $TX \wedge Y \in S$, because S is downward saturated, so we would have $TX \in S$, but $S \subseteq S^U$. Hence $S \subseteq S^U - \{TX \wedge Y\}$. But $S^U - \{TX \wedge Y\}$ is still upward saturated. So $S^U \subseteq S^U - \{TX \wedge Y\}$. Then $TX \wedge Y$ is not in S^U. □

Proposition 3.4. If S is downward saturated and atomically consistent, then both S and S^U are consistent.

PROOF. Suppose S is downward saturated. Then it is easy to check that, if S contains TX and FX, S must contain TY and FY for some subformula Y of X. Consequently if S is atomically consistent, it must be consistent.

Further, S and S^U contain the same signed atomic statements, so if S is atomically consistent, so is S^U. And by Proposition 3.3, if S is downward saturated, so is S^U. The consistency of S^U now follows by the same argument as that for S. □

Completeness issues play a minor role here, but we include the following, without proof, for completeness' sake.

Proposition 3.5

(a) If S is atomically complete, then S^U is complete.

(b) If S is downward saturated, atomically consistent, and atomically complete, then S^U is a model set.

(c) If S is downward saturated and atomically consistent, then S can be extended to a model set.

4. LOGIC PROGRAMS—SYNTAX

We will only consider clauses with a single atomic formula in the conclusion, but we wish to allow arbitrary propositional connectives in the hypothesis. Accordingly we define clause somewhat differently than usual. We assume the language L has some fixed set of function, constant, and relation symbols, and formulas of L are defined in the usual way. Recall we take $\wedge$, $\vee$, $\neg$, $\forall$, and $\exists$ as primitive. We also assume that $=$ is a relation symbol of L, though we do not allow its use in logic programs. Rather, like quantifiers, it comes up in their analysis.

Definition. A *program* is a possibly infinite set of clauses. A *clause* is a pair consisting of an atomic formula A of L and a finite set $\{B_1, \ldots, B_n\}$ $(n \geq 0)$ of quantifier free formulas of L, neither A nor $\{B_1, \ldots, B_n\}$ containing the symbol $=$, written

$$A \leftarrow B_1, \ldots, B_n.$$

A is the *conclusion* and $\{B_1, \ldots, B_n\}$ is the *premise* of this clause.

5. SEMANTICS

We define the notion of a three-valued model (or better, a partial model) for a finite logic program (or better, for the IFF formula associated with it). Such a model will be a consistent set of signed atomic statements of L.

Definition. Let S be a consistent set of signed atomic statements of L. For a statement A we say S makes A true (false) if $TA \in S^U$ (if $FA \in S^U$). For a finite set $\{B_1, \ldots, B_n\}$ of statements, S makes $\{B_1, \ldots, B_n\}$ true (false) if S makes $B_1 \wedge \cdots \wedge B_n$ true (false). Here we assume some arbitrary parenthesizing.

Since a consistent set of signed atomic statements S need not be complete, S need not make every statement true or false. On the other hand, since members of S are signed *atomic* statements, S is trivially downward saturated, hence S^U is both saturated and consistent, by Propositions 3.3 and 3.4.

The equality relation plays a somewhat special role, so we give it a special treatment.

Definition. By a *full set* we mean a consistent set S of signed atomic statements of L such that, for closed terms t and u, if $t = u$ then $T(t = u) \in S$, and if $t \neq u$ then $F(t = u) \in S$. By A *basic set* we mean a consistent set of signed atomic statements not involving $=$.

It is trivial that any basic set S can be extended to a unique full set. We call it the *full set associated with* S. In giving examples, we usually present a basic set that can be extended to a full set, rather than specifying the full set in detail.

The connective $\equiv$ can be defined in the usual ways; for example, $X \equiv Y$ means $\neg(X \wedge \neg Y) \wedge \neg(Y \wedge \neg X)$. But it is not strong enough for our purposes. For instance, if S is a consistent set of signed atomic statements, S will make $P \equiv P$ neither true nor false if S assigns no truth value to P itself. We introduce a connective $\cong$, also from Kleene, which we only use in a restricted way. $\cong$ is a connective whose behavior cannot be defined from those introduced so far. Intuitively, $X \cong Y$ is to mean X and Y have the same truth value from the set $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$.

Definition. For a consistent set S of signed atomic statements, we say:

S satisfies $X \cong Y$, where X and Y are statements (not containing $\cong$), provided S makes both X and Y true, or S makes both X and Y false, or S assigns neither X nor Y a truth value.

S satisfies $X \cong Y$, where X and Y are formulas, provided S satisfies every closed instance of $X \cong Y$.

S satisfies a conjunction of formulas of the form $X \cong Y$ provided S satisfies each one of them.

Now let P be a finite logic program. It is turned into a single formula in the following familiar way. First, each clause

$$R(t_1, \ldots, t_n) \leftarrow B_1, \ldots, B_m$$

is replaced by

$$R(x_1, \ldots, x_n) \leftarrow (\exists y_1, \ldots, y_k)[x_1 = t_1 \wedge \cdots \wedge x_n = t_n \wedge B_1 \cdots \wedge B_m],$$

where $x_1, \ldots, x_n$ are new variables, and $y_1, \ldots, y_k$ are all the variables of $t_1, \ldots, t_n, B_1, \ldots, B_m$. Next, all rewritten clauses with the same conclusion

$$R \leftarrow D_1,$$
$$R \leftarrow D_2,$$
$$\vdots$$
$$R \leftarrow D_s$$

are replaced by the single expression $R \leftarrow D_1 \vee D_2 \vee \cdots \vee D_s$. If R is a relation symbol occurring in P with no clause in P of the form $R(t_1, \ldots, t_n) \leftarrow B_1, \ldots, B_m$, add the expression $\neg R(x_1, \ldots, x_n)$. Then the resulting set of expressions is replaced by its conjunction. Finally all occurrences of $\leftarrow$ are replaced by $\cong$. The resulting formula is denoted $D(P)$.

Finally, the main notion of this section.

Definition. A full set S is a *partial model* for logic program P provided S satisfies $D(P)$. A basic set S is a *partial model* for P provided the full set associated with S is a partial model for P.

Convention. In giving examples in this and later sections, we assume the language L has the constant, function and relation symbols that actually occur in the

program under consideration, *and no others*. This is not necessary, but it is convenient. Of course, when discussing the behavior of a program we may move to a larger language involving $=$. It will be clear when this happens.

Example. Let P be the program

$\text{even}(a) \leftarrow$

$\text{even}(s(x)) \leftarrow \neg\text{even}(x)$.

Then $D(P)$ is the formula

$\text{even}(y) \cong [y = a \vee (\exists x)(y = s(x) \wedge \neg\text{even}(x))]$.

Let S be the full set associated with $\{T\text{even}(a), F\text{even}(s(a)), T\text{even}(s(s(a))), F\text{even}(s(s(s(a)))), \ldots\}$. S is a partial model for P. That is, S satisfies $D(P)$. We check one case; consider the closed instance of $D(P)$:

$\text{even}(s(a)) \cong [s(a) = a \vee (\exists x)(s(a) = s(x) \wedge \neg\text{even}(x))]$.

Since $s(a)$ and a are different terms, $F(s(a) = a) \in S \subseteq S^U$. If t is any closed term other than a, then $s(a)$ and $s(t)$ are different; hence $F(s(a) = s(t)) \in S$, and hence by upward closure, $F(s(a) = s(t) \wedge \neg\text{even}(t)) \in S^U$. And if t is a, then $T\text{even}(t) \in S$, so $F\neg\text{even}(t) \in S^U$, $F(s(a) = s(t) \wedge \neg\text{even}(t)) \in S^U$. It follows that $F(\exists x)(s(a) = s(x) \wedge \neg\text{even}(x)) \in S^U$. Then finally, $F[s(a) = a \vee (\exists x)(s(a) = s(x) \wedge \neg\text{even}(x))] \in S^U$, so S makes $[s(a) = a \vee (\exists x)(s(a) = s(x) \wedge \neg\text{even}(x))]$ false. Trivially S makes $\text{even}(s(a))$ false, so S satisfies $\text{even}(s(a)) \cong [s(a) = a \vee (\exists x(s(a) = s(x) \wedge \neg\text{even}(x))]$.

As another example, this time let P be the program

$\text{even}(a) \leftarrow$

$\text{even}(s(s(x))) \leftarrow \text{even}(x)$

$\text{even}(x) \leftarrow \text{even}(s(s(x)))$

The basic set $\{T\text{even}(a), F\text{even}(s(a)), T\text{even}(s(s(a))), F\text{even}(s(s(s(a)))), \ldots\}$ is a partial model, but so is $\{T\text{even}(a), T\text{even}(s(s(a))), \ldots\}$.

6. MONOTONE MAPPINGS

We want to associate partial models for logic programs with fixed points of operators. Now the intersection of any family of consistent sets of signed statements is again a consistent set of signed statements, but the same is not true for union. We do have that the union of a chain, and more generally, of a directed family of consistent sets is a consistent set. This means we do not have a complete lattice, but a weaker structure, called a complete semilattice here.

In a complete lattice, [14] guarantees that every monotone function has a smallest and a greatest fixed point, and the collection of its fixed points itself constitutes a complete lattice. In a complete semilattice things are a little more complicated. A monotone function always has a smallest fixed point, though it may not have a greatest, but several maximal ones instead. Among fixed points certain *intrinsic* or *optimal* ones are singled out (definition below). There is a smallest and a greatest intrinsic fixed point, and the intrinsic fixed points constitute a complete lattice.

Interest in the fixed-point structure of monotone maps on complete semilattices was clearly rather broad in the mid 70s. [8] presented the essential facts (without

proof) and applied them to philosophical problems. [9] state and prove the same results, in different terminology, and apply them to issues in computer science. See also [10]. We use Kripke's term *intrinsic* rather than Manna and Shamir's *optimal*. And we present proofs of what we need, based on the proofs in [6].

Definition. $\langle \mathbf{C}, \leq \rangle$ is a *complete semilattice* if

(1) $\mathbf{C}$ is partially ordered by $\leq$,

(2) every nonempty subset $\mathbf{D}$ of $\mathbf{C}$ has an inf, denoted $\bigcap\mathbf{D}$ (in particular, $\mathbf{C}$ has a smallest member),

(3) every nonempty directed subset $\mathbf{D}$ of $\mathbf{C}$ has a sup, denoted $\bigcup\mathbf{D}$. (A set S is directed if, for any $A, B \in S$, there is some $C \in S$ with $A \leq C$ and $B \leq C$.)

Lemma 6.1. Let $\langle \mathbf{C}, \leq \rangle$ be a complete semilattice. Any nonempty subset of $\mathbf{C}$ having an upper bound has a least upper bound.

PROOF. Let $\mathbf{D} \subseteq \mathbf{C}$, and suppose $\mathbf{D}$ has an upper bound. Set $\mathbf{E} = \{ X \in \mathbf{C} \mid X$ is an upper bound for $\mathbf{D}\}$. By hypothesis $\mathbf{E}$ is not empty, so $\bigcap\mathbf{E} \in \mathbf{C}$. It is straightforward to check that $\bigcap\mathbf{E}$ is the least upper bound for $\mathbf{D}$.

More definitions. Let $\langle \mathbf{C}, \leq \rangle$ be a partial ordering. $\Phi : \mathbf{C} \to \mathbf{C}$ is *monotone* if $A \leq B \Rightarrow \Phi(A) \leq \Phi(B)$.
If $\Phi(A) = A$, then A is a *fixed point* of Φ.
$A, B \in \mathbf{C}$ are *compatible* if there is some $C \in \mathbf{C}$ with $A \leq C$ and $B \leq C$.
I is an *intrinsic* fixed point of Φ if I is a fixed point that is compatible with every fixed point of Φ.

Proposition 6.2. Let $\langle \mathbf{C}, \leq \rangle$ be a complete semilattice, and let Φ be monotone on $\mathbf{C}$.

(1) *If $A \leq \Phi(A)$, then Φ has a maximal fixed point above A.*

(2) (*a*) *If $A \leq \Phi(A)$, then Φ has a smallest fixed point above A.*
(*b*) *If $A \leq \Phi(A)$, $A \leq B$, and $\Phi(B) \leq B$, then the smallest fixed point above A will be below B.*

(3) *If $\Phi(B) \leq B$, then Φ has a largest fixed point below B.*

PROOF. (1): Suppose $A \leq \Phi(A)$. Let $\mathbf{D} = \{ X \in \mathbf{C} \mid A \leq X$ and $X \leq \Phi(X)\}$. We show $\mathbf{D}$ has a maximal member and it is a fixed point of Φ. Since every fixed point of Φ above A is in $\mathbf{D}$, this will establish part (1).

Since $A \in \mathbf{D}$, $\mathbf{D}$ is not empty. We claim every chain in $\mathbf{D}$ has an upper bound in $\mathbf{D}$. Let $\mathbf{K} \subseteq \mathbf{D}$ be a chain. Then $\mathbf{K}$ is a directed subset of $\mathbf{C}$, so $\bigcup\mathbf{K} \in \mathbf{C}$. Choose an arbitrary $Y \in \mathbf{K}$. Then $Y \leq \bigcup\mathbf{K}$, so $\Phi(Y) \leq \Phi(\bigcup\mathbf{K})$. But $\mathbf{K} \subseteq \mathbf{D}$, so $Y \leq \Phi(Y)$. Then $Y \leq \Phi(\bigcup\mathbf{K})$ and, since Y was arbitrary, $\bigcup\mathbf{K} \leq \Phi(\bigcup\mathbf{K})$. Also $A \leq \bigcup\mathbf{K}$; hence $\bigcup\mathbf{K} \in \mathbf{D}$.

Since $\mathbf{D}$ is a nonempty partially ordered set in which each chain has an upper bound, by Zorn's lemma $\mathbf{D}$ has a maximal member, say M. Then $M \leq \Phi(M)$, so also $\Phi(M) \leq \Phi(\Phi(M))$. Also $A \leq M$, so $A \leq \Phi(A) \leq \Phi(M)$. Hence $\Phi(M) \in \mathbf{D}$. If $M < \Phi(M)$, then $\Phi(M)$ would dominate a maximal member of $\mathbf{D}$; hence $M = \Phi(M)$.

(2): Again suppose $A \leq \Phi(A)$. Let $\mathbf{E} = \{ X \in \mathbf{C} \mid A \leq X$ and $\Phi(X) \leq X\}$. We show $\mathbf{E}$ has a smallest member and it is a fixed point of Φ. Since every fixed point of Φ above A is in $\mathbf{E}$, this will establish part (2).

Let $X \in \mathbf{E}$. Then $A \leq X$, $\Phi(A) \leq \Phi(X)$, and so $A \leq \Phi(X)$. Further, $\Phi(X) \leq X$, so $\Phi(\Phi(X)) \leq \Phi(X)$. It follows that $\Phi(X) \in \mathbf{E}$; $\mathbf{E}$ is closed under Φ.

By part (1) $\mathbf{E}$ is not empty. Hence $\cap\mathbf{E} \in \mathbf{C}$. Certainly $A \leq \cap\mathbf{E}$. Let $Y \in \mathbf{E}$. Then $\cap\mathbf{E} \leq Y$, so $\Phi(\cap\mathbf{E}) \leq \Phi(Y) \leq Y$. Since Y was arbitrary, $\Phi(\cap\mathbf{E}) \leq \cap\mathbf{E}$; $\cap\mathbf{E} \in \mathbf{E}$. Then since $\mathbf{E}$ is closed under Φ, $\Phi(\cap\mathbf{E}) \in \mathbf{E}$, so $\cap\mathbf{E} \leq \Phi(\cap\mathbf{E})$. The conclusion now follows easily.

(3): Suppose $\Phi(B) \leq B$. Let $\mathbf{F} = \{ X \in \mathbf{C} \mid X \leq B \text{ and } X \leq \Phi(X) \}$. Since $\mathbf{C}$ has a smallest member, $\mathbf{F}$ is not empty. And it is easy to check that $\mathbf{F}$ is closed under Φ.

$\mathbf{F}$ has an upper bound, B; hence by Lemma 6.1, $\cup\mathbf{F} \in \mathbf{C}$. Trivially $\cup\mathbf{F} \leq B$. Suppose $X \in \mathbf{F}$. Then $X \leq \Phi(X) \leq \Phi(\cup\mathbf{F})$. Hence $\cup\mathbf{F} \leq \Phi(\cup\mathbf{F})$. We have $\cup\mathbf{F} \in \mathbf{F}$. Since $\mathbf{F}$ is closed under Φ, $\Phi(\cup\mathbf{F}) \in \mathbf{F}$, so $\Phi(\cup\mathbf{F}) \leq \cup\mathbf{F}$. Thus $\mathbf{F}$ contains the fixed point $\cup\mathbf{F}$. Since every fixed point below B is in $\mathbf{F}$, $\cup\mathbf{F}$ must be the largest such. □

NOTE. Since a complete semilattice $\mathbf{C}$ must contain a smallest member, (1) and (2) above guarantee the existence of a smallest and a maximal fixed point. There need not be a largest member of $\mathbf{C}$, so we cannot use (3) to conclude the existence of a largest fixed point. The following proposition is as close as we can come.

Proposition 6.3. Let $\langle \mathbf{C}, \leq \rangle$ be a complete semilattice, and let Φ be monotone on $\mathbf{C}$. Also let $\mathbf{M}$ be the family of maximal fixed points of Φ.

(1) *A fixed point I of Φ is intrinsic if and only if $I \leq \cap\mathbf{M}$.*

(2) *Φ has a largest intrinsic fixed point.*

(3) *If $A \leq \Phi(A)$ and $A \leq \cap\mathbf{M}$, then the smallest fixed point of Φ above A is intrinsic.*

(4) *The family of intrinsic fixed points of Φ, ordered by $\leq$, is a complete lattice.*

PROOF. (1):

(a) Suppose I is an intrinsic fixed point of Φ. Choose an arbitrary member M of $\mathbf{M}$. Then I and M are compatible, so by Lemma 6.1, $\cup\{I, M\} \in \mathbf{C}$.

Now $I = \Phi(I) \leq \Phi(\cup\{I, M\})$ and $M = \Phi(M) \leq \Phi(\cup\{I, M\})$, so $\cup\{I, M\} \leq \Phi(\cup\{I, M\})$. It follows by Proposition 6.1, part (1), that there is a maximal fixed point, say N, above $\cup\{I, M\}$.

$M \leq \cup\{I, M\} \leq N$, but M is maximal; hence $M = \cup\{I, M\} = N$. $I \leq \cup\{I, M\}$; hence $I \leq M$. M was arbitrary; hence $I \leq \cap\mathbf{M}$.

(b) Suppose I is a fixed point of Φ and $I \leq \cap\mathbf{M}$. We show I is intrinsic. Choose any fixed point F. Then $F \leq \Phi(F)$, so by Proposition 6.1, part (1), there is a maximal fixed point M above F. Then $I \leq \cap\mathbf{M} \leq M$ and $F \leq M$, so I and F are compatible.

(2): Choose $M \in \mathbf{M}$. Then $\cap\mathbf{M} \leq M$, so $\Phi(\cap\mathbf{M}) \leq \Phi(M) = M$. Thus $\Phi(\cap\mathbf{M}) \leq \cap\mathbf{M}$. By Proposition 6.2, part (3), Φ has a largest fixed point below $\cap\mathbf{M}$, which is the largest intrinsic fixed point by part (1).

(3): Suppose $A \leq \Phi(A)$ and $A \leq \cap\mathbf{M}$. As shown in the proof of part (2), $\Phi(\cap\mathbf{M}) \leq \cap\mathbf{M}$. Then Proposition 6.2, part (2)(b), says the smallest fixed point of Φ above A will be below $\cap\mathbf{M}$, and hence intrinsic by part (1).

(4): Let $\langle \mathbf{I}, \leq \rangle$ be the family of intrinsic fixed points of Φ, ordered by the ordering relation of $\langle \mathbf{C}, \leq \rangle$ restricted to $\mathbf{I}$. Note that $\cap$ and $\cup$ are defined in terms of $\leq$ and so do not necessarily have the same meaning in $\langle \mathbf{I}, \leq \rangle$ as in $\langle \mathbf{C}, \leq \rangle$.

We show every nonempty subset of **I** has a least upper bound in $\langle \mathbf{I}, \leq \rangle$. Let $\mathbf{D} \subseteq \mathbf{I}$ be nonempty. It must be shown that there is in $\langle \mathbf{C}, \leq \rangle$ a smallest intrinsic fixed point above all members of **D**.

The argument in this paragraph takes place in $\langle \mathbf{C}, \leq \rangle$. Since every member of **D** is intrinsic, by part (2) **D** has an upper bound, and hence by Lemma 6.1 a least upper bound $\bigcup \mathbf{D} \in \mathbf{C}$. It is easy to show that $\bigcup \mathbf{D} \leq \Phi(\bigcup \mathbf{D})$ and $\bigcup \mathbf{D} \leq \bigcap \mathbf{M}$. Also $\Phi(\bigcap \mathbf{M}) \leq \bigcap \mathbf{M}$. Then Proposition 6.2, part (2)(b), says Φ has a smallest fixed point, D, in **C** above $\bigcup \mathbf{D}$ and which will be below $\bigcap \mathbf{M}$, and hence intrinsic. Then D must be the smallest intrinsic fixed point above all the members of **D**. □

Finally we discuss the notion of approximating to fixed points.

Proposition 6.4. Let $\langle \mathbf{C}, \leq \rangle$ be a complete semilattice and Φ be monotone.

(1) *Suppose $S \leq \Phi(S)$.*

(*a*) *The following defines a sequence of members $\Phi \uparrow^{\alpha}(S)$ of* **C** *for each ordinal α:*

$$\Phi \uparrow^{0}(S) = S,$$
$$\Phi \uparrow^{\alpha+1}(S) = \Phi(\Phi \uparrow^{\alpha}(S)),$$
for limit ordinals λ, $\quad \Phi \uparrow^{\lambda}(S) = \bigcup_{\alpha<\lambda} \Phi \uparrow^{\alpha}(S).$

(*b*) *The sequence $\Phi \uparrow^{\alpha}(S)$ increases with α and converges to the smallest fixed point of Φ above S.*

(2) *Suppose $\Phi(S) \leq S$.*

(*a*) *The following defines a sequence of members $\Phi \downarrow^{\alpha}(S)$ of* **C**:

$$\Phi \downarrow^{0}(S) = S,$$
$$\Phi \downarrow^{\alpha+1}(S) = \Phi(\Phi \downarrow^{\alpha}(S)),$$
for limit ordinals λ, $\quad \Phi \downarrow^{\lambda}(S) = \bigcap_{\alpha<\lambda} \Phi \downarrow^{\alpha}(S).$

(*b*) *The sequence $\Phi \downarrow^{\alpha}(S)$ decreases as α increases and converges to the largest fixed point of Φ below S.*

PROOF. Omitted. □

Say $\perp$ is the smallest member of **C**. Trivially $\perp \leq \Phi(\perp)$, so by (1) the sequence $\Phi \uparrow^{\alpha}(\perp)$ converges to the smallest fixed point of Φ. The least ordinal α for which $\Phi \uparrow^{\alpha}(\perp)$ *is* the least fixed point of Φ is called the *closure ordinal* of Φ.

Again, say **M** is the family of maximal fixed points of Φ. $\bigcap \mathbf{M}$ need not be a fixed point (see Example IV in Section 8). But $\Phi(\bigcap \mathbf{M}) \leq \bigcap \mathbf{M}$, so by Proposition 6.3, part (1), the sequence $\Phi \downarrow^{\alpha}(\bigcap \mathbf{M})$ converges to the largest *intrinsic* fixed point of Φ.

7. OPERATORS ASSOCIATED WITH PROGRAMS

Let **A** be the collection of all basic sets. Then $\langle \mathbf{A}, \subseteq \rangle$ is a partial ordering. And it is easy to check that we have closure under infs, but not under sups, only under sups of directed sets. Thus $\langle \mathbf{A}, \subseteq \rangle$ is a complete semilattice.

If P is a logic program, we let P^* be the program consisting of all clauses $C\theta$ where $C \in P$, θ is a substitution, and $C\theta$ is a statement (has no free variables). In general P^* will be infinite even if P is not.

Now, let P be a finite logic program; we associate with it an operator $\Phi_P : \mathbf{A} \to \mathbf{A}$ as follows. Let $S \in \mathbf{A}$. For an atomic statement A of L (not involving $=$),

$TA \in \Phi_P(S)$ provided some clause in P^* has conclusion A and a premise that S makes true;

$FA \in \Phi_P(S)$ provided every clause in P^* having conclusion A has a premise that S makes false.

If we had both $TA, FA \in \Phi_P(S)$, then there would be some clause in P^* whose premise S would make both true and false. If $\mathbf{C}$ is the conjunction of that premise, $TC \in S^U$ and $FC \in S^U$, so by Proposition 3.4, S would not have been atomically consistent. It follows that Φ_P maps members of $\mathbf{A}$ to members of $\mathbf{A}$.

Lemma 7.1. Let S_1 and S_2 be sets of signed atomic statements, and let X be a statement. If S_1 makes X true (false) and $S_1 \subseteq S_2$, then S_2 makes X true (false).

PROOF. Immediate from Proposition 3.2. □

It follows that for a logic program P, the operator Φ_P is monotone, and hence the results in Section 6 apply.

Example. Let P be the program

$$\text{even}(a) \leftarrow$$

$$\text{even}(s(s(x))) \leftarrow \text{even}(x)$$

$$\text{even}(x) \leftarrow \text{even}(s(s(x))).$$

For convenience we uses s^n to denote n applications of s in what follows. Let $S = \{T\,\text{even}(s(a)),\ F\,\text{even}(a),\ F\,\text{even}(s^4(a))\}$. Then $\Phi_P(S) = \{T\text{even}(a), T\text{even}(s^3(a)), F\text{even}(s^2(a))\}$.

The least fixed point of Φ_P is $\{T\text{even}(a), T\text{even}(s^2(a)), T\text{even}(s^4(a)), \dots\}$. Also both of the following are fixed points: $\{T\text{even}(a), T\text{even}(s^2(a)), T\text{even}(s^4(a)), \dots, T\text{even}(s(a)), T\text{even}(s^3(a)), \dots\}$ and $\{T\text{even}(a), T\text{even}(s^2(a)), T\text{even}(s^4(a)), \dots, F\text{even}(s(a)), F\text{even}(s^3(a)), \dots\}$. Obviously both are maximal in $\mathbf{A}$, so it follows by Proposition 6.3, part (1), that the largest intrinsic fixed point of Φ_P is also its least fixed point.

Proposition 7.2. Let P be a finite logic program. S is a fixed point of Φ_P if and only if S is a partial model for P.

PROOF. Straightforward, and omitted here. □

In [1] monotone operators of a different kind were associated with a narrower class of logic programs than we are considering here. We look at the relationship between the two kinds of operators on this class of programs.

Definition. A *definite clause* in the Apt–Van Emden sense is a clause of the form

$$A \leftarrow B_1, \ldots, B_n, \qquad n \geq 0,$$

where $A, B_1, \ldots, B_n$ are all atomic.

Apt and Van Emden associate an operator $\mathbf{T}_P$ with a program P made up of definite clauses in essentially the following way. Let S be a set of unsigned atomic statements. Then, for an atomic statement A, $A \in \mathbf{T}_P(S)$ if there is some clause in P^* of the form

$$A \leftarrow B_1, \ldots, B_n, \qquad \text{with} \quad B_1, \ldots, B_n \in S.$$

$\mathbf{T}_P$ is monotone on $\langle \mathbf{B}, \subseteq \rangle$, where $\mathbf{B}$ is the collection of all sets of atomic statements, a complete lattice.

Another definition from [1]. Let $\mathbf{T}$ be the operator associated with program P. Then

$$\mathbf{T}\uparrow_\alpha = \mathbf{T}\uparrow^\alpha(\varnothing),$$

$$\mathbf{T}\downarrow_\alpha = \mathbf{T}\downarrow^\alpha(U),$$

where U is the set of all atomic statements, the top of $\langle \mathbf{B}, \subseteq \rangle$, as $\varnothing$ is its bottom.

Proposition 7.3. For a logic program P made up of definite clauses, for all ordinals α,

(1) $\mathbf{T}\uparrow_\alpha = \{A \mid TA \in \Phi_P\uparrow^\alpha(\varnothing)\}$,

(2) $\mathbf{T}\downarrow_\alpha = U - \{A \mid FA \in \Phi_P\uparrow^\alpha(\varnothing)\}$.

PROOF. A straightforward induction on α. □

It follows from this and Apt and Van Emden's paper that the *finite failure set* for a program made up of definite clauses is $\{A \mid FA \in \Phi_P\uparrow^\omega(\varnothing)\}$.

8. EXAMPLES AND ANALOGIES

We give several simple logic programs that are useful for explicating the various fixpoint notions introduced earlier. The program examples are closely related to examples of statements given by Kripke to help illustrate the machinery of his theory of truth. We point out resemblances as we go along.

Example I. Let P be the program

$$R(a) \leftarrow R(a),$$

and let Φ_P be the associated operator. Both $\{TR(a)\}$ and $\{FR(a)\}$ are maximal fixed points of Φ_P. It follows by Proposition 6.3, part (1), that the only intrinsic fixed point of Φ_P is $\varnothing$, which is also the smallest fixed point.

The Kripke analog to this is the following statement (or rather, its formalized counterpart):

R: Statement R is true.

In Kripke's theory some fixed points make statement R true, some make it false, so it has no truth value in the least fixed point of Kripke's operator.

Example II. This time let P be the program

$R(a) \leftarrow \neg R(a)$.

Let S be a fixed point of Φ_P. If $TR(a) \in S$ then $F\neg R(a) \in S^U$ and hence $FR(a) \in \Phi_P(S)$. Since S is a fixed point, $FR(a) \in S$. But since members of the domain of Φ_P are *consistent* sets of signed atomic statements, this is impossible. Hence $TR(a) \notin S$. By a similar argument $FR(a) \notin S$. Thus $R(a)$ is given a truth value in *no* fixed point for Φ_P; the only fixed point is $\varnothing$.

The Kripke analog now is the following statement:

R: Statement R is false.

Notice the essential difference between Examples I and II. In both cases $R(a)$ receives no truth value in the least fixed point. But in I it is because either value is possible (in extensions of the least fixed point), while in II, neither value is possible.

Example III. Let P be the program

$R(a) \leftarrow R(a) \vee \neg R(a)$.

The least fixed point of Φ_P is easily seen to be $\varnothing$; hence in it, $R(a)$ receives no truth value. On the other hand, if M is a maximal fixed point, it will assign $R(a)$ a truth value; hence M will make $R(a) \vee \neg R(a)$ true, and hence $TR(a) \in \Phi_P(M) = M$. Thus $\{TR(a)\}$ is the only maximal fixed point, which is thus the largest intrinsic fixed point.

The Kripke analog is the following:

R: Statement R is either true or false.

Example IV. Let P be the following program:

$Q(a) \leftarrow R(a) \vee \neg R(a)$,

$R(a) \leftarrow R(a)$.

The least fixed point of Φ_P, once again, is $\varnothing$. Both $\{TQ(a), TR(a)\}$ and $\{TQ(a), FR(a)\}$ are maximal fixed points. And it is easy to see that if M is any maximal fixed point, either $TR(a) \in M$ or $FR(a) \in M$, and hence in any case $TQ(a) \in M$. Thus, if $\mathbf{M}$ is the family of maximal fixed points, $\bigcap\mathbf{M} = \{TQ(a)\}$.

On the other hand, if I is the largest (or any) intrinsic fixed point, $I \subseteq \bigcap\mathbf{M}$ by Proposition 6.3, part (1). Then I can assign no truth value to $R(a)$. It follows that $\Phi_P(I)$ (that is, I) assigns no truth value to $Q(a)$. Thus $I = \varnothing$ and we have an example in which the largest intrinsic fixed point I is strictly below $\bigcap\mathbf{M}$.

The Kripke analog is the following pair of statements:

Q: Statement R is either true or false.

R: Statement R is true.

In Kripke's terms, Q is a statement that is true in every fixed point in which it has a truth value, yet it has no *intrinsic* truth value.

The reader may enjoy determining the fixpoint structure of the operators associated with the following programs.

Example V.

$Q(a) \leftarrow R(a) \vee \neg R(a)$,

$R(a) \leftarrow \neg R(a)$.

Example VI.

$$Q(a) \leftarrow R(a) \vee \neg R(a),$$
$$R(a) \leftarrow \neg Q(a).$$

Example VII.

$$Q(a) \leftarrow \neg R(a),$$
$$R(a) \leftarrow \neg Q(a).$$

Example VIII.

$$Q(a) \leftarrow R(a) \vee \neg R(a),$$
$$R(a) \leftarrow Q(a) \vee \neg Q(a).$$

Finally, consider the following variation on Example IV; program P_1:

$$R(a) \leftarrow R(a) \vee \neg R(a),$$
$$R(a) \leftarrow R(a).$$

We use this to give a simple example of Kleene logic manipulations.

As defined in Section 5, a partial model of P_1 is any basic set whose associated full set satisfies $D(P_1)$, which is

$$R(x) \cong \{[x = a \wedge R(a)] \vee [x = a \wedge (R(a) \vee \neg R(a))]\}.$$

Definition. Let us call a formula $X \cong Y$ *K-valid* if every consistent set of signed atomic statements satisfies $X \cong Y$.

The following replacement result is easy to verify. It is done by induction on formula complexity, just as with the replacement theorem for classical logic.

If $X_1 \cong X_2$ is K-valid, and if Y_2 differs from Y_1 by the replacement of one or more occurrences of X_1 by X_2, then $Y_1 \cong Y_2$ is K-valid.

Also we have the following easy transitivity result.

If S satisfies $X \cong Y$ and $Y \cong Z$, then S satisfies $X \cong Z$.

Now, the following are K-valid, where A, B, and C are any formulas:

(1) $(A \wedge B) \vee (A \wedge C) \cong A \wedge (B \vee C)$,

(2) $A \vee (A \vee B) \cong A \vee B$.

We give a small portion of the verification, by way of illustration. Suppose, for simplicity, that A, B, and C have no free variables. Let S be an arbitrary consistent set of signed atomic statements, making the left-hand side of (1) false; we show S also makes the right-hand side of S false.

By supposition, $F(A \wedge B) \vee (A \wedge C) \in S^U$. Since S^U is downward saturated (Proposition 3.3), $F(A \wedge B) \in S^U$ and $F(A \wedge C) \in S^U$. Again by downward saturation, since $F(A \wedge B) \in S^U$, either $FA \in S^U$ or $FB \in S^U$. If $FA \in S^U$, by upward saturation, $FA \wedge (B \vee C) \in S^U$. Suppose now that $FA \notin S^U$. Then $FB \in S^U$. Simi-

larly $FC \in S^U$. Then by upward saturation, $FB \vee C \in S^U$; hence $FA \wedge (B \vee C) \in S^U$. Thus in either case S makes $A \wedge (B \vee C)$ false.

Now, returning to the example. As special cases of (1) and (2), the following are K-valid:

(1′) $[x = a \wedge R(a)] \vee [x = a \wedge (R(a) \vee \neg R(a))]$
$\cong x = a \wedge [R(a) \vee (R(a) \vee \neg R(a))]$,

(2′) $R(a) \vee (R(a) \vee \neg R(a)) \cong R(a) \vee \neg R(a)$.

It follows, using replacement and transitivity, that a full set S satisfies $D(P_1)$ if and only if S satisfies

$$R(x) \cong x = a \wedge (R(a) \vee \neg R(a)).$$

That is, S is a partial model for P_1 if and only if S is a partial model for

$$R(a) \leftarrow R(a) \vee \neg R(a),$$

the program of Example III. Then, by Proposition 7.2, the operators associated with P_1 and the program of Example III have the same fixed points.

Definition. We call two logic programs P_1 and P_2 *equivalent* if they have the same partial models.

The logic programs of Examples I and II are *not* equivalent, though they have the same *smallest* partial models. As we use it, equivalence requires that all partial models be considered.

When converting one logic program into another using logic manipulations, the possibility of no response must be taken into account. That is, three-valued logic should be used. The rather trivial example above is sufficient to demonstrate that such arguments are very much like classical ones, and are essentially no harder to carry out. The reader may like to practice by establishing the equivalence of the following two logic programs:

$$Q(a) \leftarrow \neg R(a),$$
$$R(a) \leftarrow \neg Q(a)$$

and

$$Q(a) \leftarrow \neg R(a),$$
$$R(a) \leftarrow R(a).$$

Finally, we use this notion of equivalence to establish a normal-form theorem.

Definition. A *literal* is an atomic formula or the negation of an atomic formula.

Proposition 8.1. Every logic program is equivalent to one in which clauses are of the form

$$A \leftarrow B_1, \ldots, B_n$$

where each of $B_1, \ldots, B_n$ *is a literal.*

PROOF. We give a series of replacements, each of which turns a program into an equivalent one. We omit a proof of this equivalence. The replacements are the

expected ones; the point is, they preserve behavior in *all partial* models, not just in the smallest ones, or in classical ones. The program

$$A \leftarrow B_1, \ldots, B_n, C \wedge D$$

can be replaced by

$$A \leftarrow B_1, \ldots, B_n, C, D.$$

The program

$$A \leftarrow B_1, \ldots, B_n, \neg\neg C$$

can be replaced by

$$A \leftarrow B_1, \ldots, B_n, C.$$

The program

$$A \leftarrow B_1, \ldots, B_n, \neg(C \wedge D)$$

can be replaced by

$$A \leftarrow B_1, \ldots, B_n, \neg C,$$
$$A \leftarrow B_1, \ldots, B_n, \neg D. \quad \square$$

9. EXCESS STRENGTH:

In [1] an example is given of a logic program for which the closure ordinal, in their sense, is not ω but $\omega + \omega$. The example carries over directly to the present setting. The program is the following:

$$P(a) \leftarrow P(x), Q(x),$$
$$P(s(x)) \leftarrow P(x),$$
$$Q(b) \leftarrow ,$$
$$Q(s(x)) \leftarrow Q(x).$$

Let Φ be the operator associated with this program. It is easily verified that, for $\alpha \leq \omega$,

$$\Phi \uparrow^{\alpha}(\varnothing) = \left\{TQ(s^k(b)), FQ(s^k(a)), FP(s^k(b)) \middle| k < \alpha\right\}$$

and

$$\Phi \uparrow^{\omega+\alpha}(\varnothing) = \Phi \uparrow^{\omega}(\varnothing) \cup \left\{FP(s^k(a)) \middle| k < \alpha\right\}.$$

The least fixed point is $\Phi \uparrow^{\omega+\omega}(\varnothing)$.

This example can be generalized, raising the closure ordinal. The question is, how high can it be pushed? In fact, it follows from [2] that the limit is Church-Kleene ω_1, the first nonrecursive ordinal. We sketch an alternate proof that the machinery introduced here allows the semantic characterization of the Π^1_1 relations, and thus is much too powerful for computational purposes.

In [12] Smullyan presented *elementary formal systems* as a mechanism for defining and proving things about the recursively enumerable relations. Elementary formal systems are essentially notational variants of the definite clauses of Section 7. In [5] the elementary-formal-system machinery was generalized in several directions. In

one, arbitrary data structures were allowed, and a connection with search computability established. In another, universal quantifiers were allowed (in premise parts only), creating what were called ω *elementary formal systems*. Connections were established between ω elementary formal systems and hyperelementary theory, a generalization of hyperarithmetic theory. In particular, for a data structure of numbers (or of terms that are Gödel numberable), it was shown that the relations characterizable by ω elementary formal systems are the Π_1^1 relations (also see [4]).

Consequently, to establish our claim here it is enough to show how ω elementary formal systems can be translated into logic programs. In fact, all the elementary-formal-system machinery is directly available. What remains is to show how to simulate universal quantifiers, and that is straightforward. The following illustrates how it is done. Consider the program

$$A(x) \leftarrow \neg B(x),$$

$$B(x) \leftarrow C(x, y),$$

$$C(x, y) \leftarrow \neg D(x, y).$$

It is easily verified that, if S is any partial model for this program, S satisfies $A(x) \cong (\forall y)D(x, y)$.

We note that Kripke's theory of truth displayed a similar Church-Kleene ω_1 phenomenon. See the remarks at the end of [8].

Let $\langle \mathbf{C}, \leq \rangle$ be a complete semilattice. If $\mathbf{D} \subseteq \mathbf{C}$ is directed and Φ is monotone on $\mathbf{C}$, then $\{\Phi(D) \mid D \in \mathbf{D}\}$ is directed. We say Φ is *continuous* if, for any directed set $\mathbf{D}$, $\Phi(\bigcup \mathbf{D}) = \bigcup\{\Phi(D) \mid D \in \mathbf{D}\}$. In the case of interest to us here—$\langle \mathbf{A}, \subseteq \rangle$ where $\mathbf{A}$ is the collection of consistent sets of signed atomic statements—continuity takes on a simple character. For every $S \in \mathbf{A}$, $\{S_0 \subseteq S \mid S_0$ is finite$\}$ is directed, and $S = \bigcup\{S_0 \subseteq S \mid S_0$ is finite$\}$. So, if Φ is monotone *and continuous*, $Z \in \Phi(S) \Leftrightarrow Z \in \Phi(S_0)$ for some finite $S_0 \subseteq S$. Conversely, if a mapping Φ meets this condition, it follows that it is monotone and continuous. Finally, such mappings have closure ordinal ω.

In more conventional programming languages, only programming constructs whose interpretation is continuous are available. This is simply not the case with the logic-programming machinery considered here. Continuity must be imposed as a separate condition. We propose the following.

Call a logic program P *acceptable* if Φ_P is a continuous map. Only acceptable logic programs should be considered acceptable.

We conclude with the following, somewhat vague questions:

What are syntactic criteria for recognizing acceptable programs?

What is the relationship between a program being acceptable and Clark's notion of an allowed querry [3]?

What useful notions can be developed that are similar to acceptability, but weaker? For example, if P is a program in which negation is not used, the closure ordinal of Φ_P need not be ω, but the problem is with F-signed statements. TX is in the least fixed point of Φ_P if and only if $TX \in \Phi_P \uparrow {}^{\omega}(\varnothing)$. Are there other "semiacceptable" notions like this?

REFERENCES

1. Apt, K. R. and Van Emden, M. H., Contributions to the Theory of Logic Programming, *J. Assoc. Comput. Mach.* 29:841–862 (1982).
2. Blair, H., The Recursion-Theoretic Complexity of the Semantics of Predicate Logic as a Programming Language, *Inform. and Control* 54:25–47 (1982).
3. Clark, K. L., Negation as failure, in H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 293–322.
4. Fitting, M., Elementary Formal Systems for Hyperarithmetical Relations, *Z. Math. Logik Grundlag. Math.* 24:25–30 (1978).
5. Fitting, M., *Fundamentals of Generalized Recursion Theory*, North-Holland, Amsterdam, 1981.
6. Fitting, M., Notes on the Mathematical Aspects of Kripke's Theory of Truth, *Notre Dame J. Formal Logic*, to appear.
7. Kleene, S. C., *Introduction To Metamathematics*, Van Nostrand, New York, 1952.
8. Kripke, S., Outline of a Theory of Truth, *J. Philos.* 72:690–716 (1975).
9. Manna, Z. and Shamir, A., The Theoretical Aspect of the Optimal Fixed Point, *SIAM J. Comput.* 5:414–426 (1976).
10. Manna, Z. and Shamir, A., The optimal approach to recursive programs, *Comm. ACM* 20:824–831 (1977).
11. Mycroft, A., Logic Programs and Many-Valued Logic, in: M. Fontet and K. Mehlhorn (eds.), *STACS 84, Symposium of Theoretical Aspects of Computer Science, Proceedings*, Springer Lecture Notes in Computer Science, 166, pp. 274–286.
12. Smullyan, R. M., *Theory of Formal Systems*, revised edition, Princeton U.P., Princeton, 1961.
13. Smullyan, R. M., *First-Order Logic*, Springer, Berlin, 1968.
14. Tarski, A. A lattice-theoretical fixpoint theorem and its applications, *Pacific J. Math.* 5:285–309 (1955).