Theoretical Computer Science 48 (1986) 229-255
North-Holland

PARTIAL MODELS AND LOGIC PROGRAMMING *

Melvin FITTING

Department of Mathematics and Computer Science, Herbert H. Lehmann College, Bronx, NY 10468, U.S.A., and Department of Computer Science, The Graduate School and University Center, City University of New York, NY10036, U.S.A.

Communicated by A. Meyer
Received November 1985
Revised July 1986

Abstract. Three extensions of the standard PROLOG fixpoint semantics are presented (called *sat*, *strong*, and *weak*), using partial models, models which may fail to assign truth values to all formulas. Each of these semantics takes negation and quantification into account. All three are *conservative*: they agree with the conventional semantics on pure Horn clause programs. The sat and the strong semantics incorporate the domain closure assumption, but differ on whether to assign a truth value to a classically valid formula some part of which lacks a truth value. The weak semantics is similar to the strong semantics but abandons the domain closure condition, and consequently, all programs give rise to continuous operators in this semantics. For the weak semantics, a sound and complete proof procedure is given, based on semantics tableaus (or equivalently, Gentzen Sequents).

1. Introduction

Logic programming with pure Horn clause formulas has a semantics that is well-understood, thanks to [11, 1]. Extending the machinery to add negation (and other connectives and quantifiers) is more problematic. Most fundamentally, the relations that one can characterize through Horn clause programs are exactly the recursively enumerable ones. Since not every recursively enumerable relation is recursive, the most natural notion of negation, complementation, is not available. 'Approximate' versions such as negation by failure are generally used instead.

But there is also a semantic problem having nothing to do with issues of computability. Given a program with the single axiom $P \leftarrow \neg P$, it seems clear that *no* truth value can meaningfully be assigned to P if negation is to behave in an intuitively plausible way. The issue is the same as in the traditional liar paradox. If P is true, the axiom $P \leftarrow \neg P$ says P is false, and conversely. A semantics based on conventional classical models cannot be appropriate.

* Partially supported by NSF Grant DCR 8504825.

Partial models are a natural generalization of classical models. In a partial model, the mapping from statements to truth values is a partial, not necessarily total, function. Statements can be true, false, or lack a truth value. Of course, the assignment of truth values must meet some constraints to reflect our understanding of the logical connectives and quantifiers. Partial models seem to have been first considered in philosophy, as a means of providing a semantics for languages containing their own truth predicate. Kripke's work [6], in fact, was motivation for the present work. Partial models have been introduced into computer science as well in [7, 3]. We use them here as a natural tool for the semantical treatment of negation in logic programming.

In addition to the inherent problems raised above concerning negation, there are also issues that are matters of opinion. For instance, consider a program containing the axiom $P \leftarrow Q \vee \neg Q$. Should the truth value of P depend on that of Q? In PROLOG using negation by failure, for instance, a goal of $\leftarrow P$ will invoke a further goal of $\leftarrow Q$, and if the work on this goal never terminates, neither will that for $\leftarrow P$. Loosely, P has no truth value if Q has none. But one can also make a case for the position that P should be true, or $\leftarrow P$ should succeed simply because $Q \vee \neg Q$ is a tautology.

We do not intend to discuss which of these positions is the 'right' one. We believe both are acceptable, and rightness is relative to the application one has in mind. The only firm requirement we see is that any proposed extension of pure Horn clause programming should be *conservative*, that is, when negation and the other added programming machinery is not explicitly used in a program, then program behavior should agree with the conventional semantics. This still allows for a multiplicity of extensions, which should be considered a virtue, not a vice.

In this paper we use partial models to provide semantics for logic programming allowing all connectives and quantifiers. In fact, we give three different semantics, all meeting the conservativeness condition. The first of these, which we call the *saturated semantics*, embodies a natural generalization of the treatment of negation in [1], to allow negation in the body. This was previously considered at some length in [3] in a slightly different form. In the saturated semantics, with a program containing $P \leftarrow Q \vee \neg Q$, whether or not P has a truth value depends on whether or not Q does. The second semantics we consider, called the *strong model set semantics*, differs from the first on precisely this point. $Q \vee \neg Q$ has a truth value of true, whether or not Q has a truth value. The third semantics, called the *weak model set semantics*, is like the strong model set semantics but the domain-closure assumption, which had been made in the first two semantics, is dropped. The domain-closure assumption essentially says there is a fixed domain that variables range over, consisting of things that have names in the language. It is a common experience in logic that restricting things to a single domain gives rise to uncomputability problems. Here, dropping the domain-closure assumption has remarkable consequences: all programs now give rise to continuous operators. This is not the case with the first two semantics. Note, for instance, the example given in [1] showing their T-operator

need not be downward continuous. Downward continuity of the T-operator corresponds to continuity of our saturated semantics operator via Proposition 8.10 below.

In fact, for the weak model set semantics we provide a corresponding proof procedure, based on Smullyan-style semantic tableaus, and we prove completeness and soundness. From the proof procedure it is evident that we have negation by *refutation* rather than negation by failure. The tableau system we give has a nice duality between proof and disproof procedures. We plan to investigate this system further elsewhere.

We begin with several background sections on logical and algebraic machinery before we get to the semantics proper in Section 7.

2. Logic background

We find it simplifies things to use Smullyan's device of *signed* formula, prefixing a formula with T or F to indicate its intended truth value, instead of introducing a valuation function mapping formulas to truth values. Also we find it natural to use *model sets*, Hintikka style, rather than models themselves. A model set can be thought of as the set of all (signed) statements of some language that are true in a particular model. Then a *partial* model will simply be some subset of a model set that meets certain natural closure conditions. The definition, however, will be a direct one and we will not talk about models themselves. All this is a matter of personal taste, not theoretical necessity.

Definition. A *language* is determined by specifying its constant, function, and relation symbols. We assume = is a relation symbol of every language. We generally use L to stand for a language. *Terms* are built up from constant symbols and variables (which are common to all our languages), using function symbols in the usual way. *Atomic formulas* are expressions of the form $R(t_1, \ldots, t_n)$ where R is an n-place relation symbol and $t_1, \ldots, t_n$ are terms. (We usually write $t_1 = t_2$ instead of $=(t_1, t_2)$ though.) We also allow truth and falsehood constants, $\top$ and $\bot$, as atomic formulas. Finally, *formulas* are built up from atomic formulas in the usual way allowing all of $\wedge$, $\vee$, $\neg$, $\supset$, $\forall$, and $\exists$. We call a formula a *statement* if it contains no free variables. If φ is a formula with free variables among $x_1, \ldots, x_n$, we indicate this by writing $\varphi(x_1, \ldots, x_n)$, and we write $\varphi(t_1, \ldots, t_n)$ for the result of substituting the terms $t_1, \ldots, t_n$ for free occurrences of $x_1, \ldots, x_n$ in φ.

Definition. A *signed formula* of L is TX or FX where X is a formula of L and T and F are two new symbols, intuitively representing truth and falsehood.

In order not to have to consider a multiplicity of similar cases in definitions and proofs, we use notational grouping conventions of Smullyan. We sketch briefly here; a full discussion is in [10].

Definition. Signed nonatomic formulas are grouped into four categories: α (conjunctive), β (disjunctive), γ (universal) and δ (existential). These categories and their *components* (for α and β) or *instances* (for γ and δ) are specified in Table 1(a)-(d).

Table 1

α	α_1	α_2
$TX \wedge Y$	TX	TY
$FX \vee Y$	FX	FY
$FX \supset Y$	TX	FY
$F\neg X$	TX	TX

(a)

β	β_1	β_2
$TX \vee Y$	TX	TY
$FX \wedge Y$	FX	FY
$TX \supset Y$	FX	TY
$T\neg X$	FX	FX

(b)

γ	$\gamma(t)$
$T(\forall x)\varphi(x)$	$T\varphi(t)$
$F(\exists x)\varphi(x)$	$F\varphi(t)$

(c)

δ	$\delta(t)$
$T(\exists x)\varphi(x)$	$T\varphi(t)$
$F(\forall x)\varphi(x)$	$F\varphi(t)$

(d)

Negation is treated somewhat redundantly. We are not after efficiency here—only theoretical convenience. In the definition below, and throughout the paper, "$\Rightarrow$" is used in the metalanguage to stand for "implies".

Definition. Let S be a set of signed statements of L. S is *consistent* if not both $TX \in S$ and $FX \in S$ for any statement X, $F\top \notin S$ and $T\bot \notin S$; S is *atomically consistent* if not both $TX \in S$ and $FX \in S$ for an atomic statement X, $F\top \notin S$ and $T\bot \notin S$. S is *L-complete* if either $TX \in S$ or $FX \in S$ for each statement X of L; S is *L-atomically complete* if either $TX \in S$ or $FX \in S$ for each atomic statement X of L other than $\top$ and $\bot$.

S is *L-downward saturated* if

- $T\top \in S$ and $F\bot \in S$,
- $\alpha \in S \Rightarrow \alpha_1 \in S$ and $\alpha_2 \in S$,
- $\beta \in S \Rightarrow \beta_1 \in S$ or $\beta_2 \in S$,
- $\gamma \in S \Rightarrow \gamma(t) \in S$ for every closed term t of L,
- $\delta \in S \Rightarrow \delta(t) \in S$ for some closed term t of L.

S is *L-upward saturated* if

- $T\top \in S$ and $F\bot \in S$,
- $\alpha_1 \in S$ and $\alpha_2 \in S \Rightarrow \alpha \in S$,
- $\beta_1 \in S$ or $\beta_2 \in S \Rightarrow \beta \in S$,
- $\gamma(t) \in S$ for every closed term t of $L \Rightarrow \gamma \in S$,
- $\delta(t) \in S$ for some closed term t of $L \Rightarrow \delta \in S$.

S is *L-saturated* if S is both L-downward and upward saturated. Finally, S is an *L-model set* if S is consistent, L-complete and L-saturated.

As we use it, *consistent* means 'obviously' consistent: there is no syntactic contradiction directly present. No notion of derivation is involved. This notion of consistency, when combined with the other notions, does correspond to more familiar usage. We sketch the main facts we need; more detailed proofs can be found in [3].

It is easy to see that the intersection of a family of L-upward saturated sets is again L-upward saturated. Also, the set of all signed statements of L is L-upward saturated. It follows that any set S of signed statements of L has a *smallest* L-upward saturated extension; intersect all L-upward saturated sets that extend S.

Definition. The smallest L-upward saturated extension of S is called the *upward saturated closure* of S, and is denoted by S^{U}.

If S is consistent, S^{U} is easily shown to be consistent. More generally, if S is *atomically* consistent and L-downward saturated, S, and hence S^{U}, will be consistent. Likewise, if S is L-downward saturated, S^{U} will be L-downward saturated. We thus have the following fundamental facts.

Proposition 2.1. *Any L-downward saturated, (atomically) consistent set S has a smallest upward saturated extension S^{U} which is consistent and L-saturated. In particular, any consistent set of signed atomic statements has a unique smallest consistent L-saturated extension.*

L-saturated, consistent sets will provide us with one notion of partial model once equality has been taken into account. If S is such a set, think of S as saying X is true if $TX \in S$, X is false if $FX \in S$, and assigning no truth value to X if $TX \notin S$ and $FX \notin S$. According to the definition of L-saturated, S will make $P \vee Q$ false only if it makes both P and Q false; S will make $P \vee Q$ true only if it makes one of P or Q true. Consequently, if S assigns no truth value to P and to Q, $P \vee Q$ will have none either. In particular, not every L-saturated, consistent set will make $P \vee \neg P$ true, though none can make it false.

In [5], a three-valued logic was introduced, with the third truth value intended to represent undefined or unknown. L-saturated, consistent sets correspond exactly to Kleene's logic. If we have an L-saturated, consistent set S and we use it to define a three-valued mapping which maps a statement X to the truth value that S assigns to X, and to 'undefined' if S assigns no value to X, then we have a Kleene three-valued truth function. Conversely, each Kleene three-valued truth function determines an L-saturated, consistent set.

Next we turn to model sets. It is straightforward to show that any L-upward saturated set that is atomically L-complete is simply L-complete. Combining this with earlier results we have the following proposition.

Proposition 2.2. *If S is L-downward saturated, (atomically) consistent, and atomically L-complete, then S^U is an L-model set.*

The following easy consequence is sometimes called Hintikka's Lemma, and plays a key role in tableau completeness proofs.

Proposition 2.3. *Any consistent, L-downward saturated set is a subset of some L-model set.*

Proof. Let S be a consistent, L-downward saturated set. For each atomic L-statement A other than $\top$ and $\bot$, with neither TA nor FA in S, arbitrarily add one. The result is still consistent, L-downward saturated, and is complete at the atomic level. Take its upward saturated closure. This is an L-model set. □

Notice that, though a consistent, L-downward saturated set S has a *unique* upward saturated closure, it will not have a unique L-model set extension in general because of the arbitrariness involved in making S complete on the atomic level in the proof above.

Finally, we wish to give equality special handling.

Definition. We say a set S of signed statements has the *substitution property* provided $\mathrm{T}t = u \in S$, $Z \in S \Rightarrow Z' \in S$ where Z' is like Z except for containing occurrences of the closed term u at zero or more places where Z contains occurrences of the closed term t. We say S has the *atomic substitution property* if the condition above is true for Z signed *atomic*.

We say a set S is *L-normal* if

(1) $\mathrm{T}t = t \in S$ for all closed terms t of L,

(2) S has the substitution property.

The definition of normality postulates reflexivity of equality but not transitivity or symmetry, but it is easy to show that these follow. For instance, suppose S is L-normal and $\mathrm{T}t = u \in S$. By definition of normality, $\mathrm{T}t = t \in S$. Now, replacing one t-occurrence in $\mathrm{T}t = t$ by u we get $\mathrm{T}u = t \in S$. Transitivity is similar.

To avoid redundancy, if S is an L-saturated set that is L-normal we will simply call it an *L-normal saturated set*. Similarly, for *L-normal model set*.

Proposition 2.4. *Let S be consistent, L-downward saturated, and let S contain all signed statements of the form $\mathrm{T}t = t$ for closed terms t of L, and have the atomic substitution property. Then S^U is an L-normal saturated set. Further, if S is atomically L-complete, then S^U is an L-normal model set.*

The following combines this proposition with Proposition 2.3.

Proposition 2.5. *Let S be consistent, L-downward saturated, contain all signed statements of the form $\mathrm{T}t = t$ for closed terms t of L, and have the atomic substitution property. Then S is a subset of some L-normal model set.*

The connection with conventional models is straightforward, though we make no direct use of it. A signed statement $\mathrm{T}X$ is said to be true in a model if X is true; $\mathrm{F}X$ is true if X is false.

Proposition 2.6. *Any L-normal model set is satisfiable in some classical model in which the interpretation of the relation symbol $=$ is the equality relation.*

3. Free treatment of equality

The conditions for equality in the previous section provide a foundation on which special equality conditions can be placed. We impose a *free interpretation* of equality, as is common in logic programming work: different terms of L are assumed to be semantically distinct. But when we abandon the domain-closure condition later on, we will have to deal with languages that extend L, and for terms of such a language we do not want to make so strong an assumption. The following pushes freedom of L terms as far as prudence allows.

Definition. For two languages L and K we write $L \leqslant K$ if L and K have the same relation and function symbols, and every constant symbol of L also occurs in K, though K may contain more constant symbols.

Definition. Let $L \leqslant K$ and let S be a set of signed statements of K. We say S is *free over* L if:

(1) $\mathrm{F}c = d \in S$ for distinct constant symbols c and d of L;

(2) $\mathrm{F}f(t_1, \ldots, t_n) = c \in S$ for function symbol f and constant symbol c of L, and closed terms $t_1, \ldots, t_n$ of K;

(3) $\mathrm{F}f(t_1, \ldots, t_n) = g(u_1, \ldots, u_m) \in S$ for distinct function symbols f and g of L, and closed terms $t_1, \ldots, t_n, u_1, \ldots, u_m$ of K;

(4) $\mathrm{F}t_i = u_i \in S$ for some $i \Rightarrow \mathrm{F}f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n) \in S$ for a function symbol f of L and closed terms $t_1, \ldots, t_n, u_1, \ldots, u_n$ of K.

The definition above embodies the notion that symbols of L are to be interpreted freely within the possibly larger language K. If $K = L$, i.e., if S is a set of signed L-statements, things can be said much more simply. It is easily shown by structural induction that, for a set S of signed L-statements that is free over L, we have $\mathrm{F}t = u \in S$ for distinct closed terms t and u of L. Since $\mathrm{T}t = t$ is in normal sets, for closed terms t, the following proposition is easily established.

Definition. Let F_L be $\{T t = t \mid t$ is a closed term of $L\} \cup \{F t = u \mid t$ and u are distinct closed terms of $L\}$.

Proposition 3.1. *Let S be a set of signed L-statements. S is L-normal and free over L if and only if $F_L \subseteq S$.*

4. Closure notions

An interpretation in classical logic is an assignment of truth values to atomic statements. We will not allow assignment to atomic statements involving $=$; this is to have the free interpretation. But more importantly, we will allow interpretations to be *partial* functions even when $=$ is not involved. Not every atomic statement need get a truth value. As usual, instead of working with maps to truth values, we work with sets of signed statements. The key fact then is that atomic completeness is not required of an interpretation.

Definition. A *partial L-interpretation* is a consistent set of signed atomic statements of L, none involving $=$, $\top$, or $\bot$.

The problem is, what about nonatomic statements. We present three ways of extending partial L-interpretations to cover all statements. Each has a certain intuitive plausibility, but makes different assumptions about the behavior of statements lacking a truth value. The definitions are actually given as *closure* notions for partial L-interpretations.

Definition. Let I be a partial L-interpretation. By the *saturated closure of I*, denoted I^{sat}, we mean

$$\bigcap \{U \mid U \text{ is } L\text{-normal, } L\text{-upward saturated, free over } L, \text{ and } I \subseteq U\}.$$

The collection of *all* signed statements of L is L-normal, L-upward saturated, free over L, and extends I. Consequently, the set being intersected in the definition above is nonempty, and the definition is meaningful. In fact, the intersection will also be consistent. This follows most easily from Proposition 4.1 below.

Now we can think of I as assigning truth values to a nonatomic statement X of L according to whether $TX \in I^{sat}$ or $FX \in I^{sat}$. For any partial L-interpretation I, I^{sat} will never say that a logically valid statement of L is false, but it need not assign truth in every valid case. For instance, if P is atomic and neither TP nor FP is in I, $TP \vee \neg P$ will not be in I^{sat}. A more detailed study of this notion can be found in [3].

The definition above parallels the other two closure notions presented below, but it will be convenient to have an alternate characterization of I^{sat}. The following is easily shown using Proposition 3.1.

Proposition 4.1. $I^{\text{sat}} = (I \cup F_L)^{\text{U}}$.

The previous closure notion is based on the Kleene three-valued logic from [5]. The next one is based on the *supervaluation* notion from [12].

Definition. Again, let I be a partial L-interpretation. By the *strong model set closure of* I, denoted I^{strong}, we mean

$$\bigcap \{M \mid M \text{ is an } L\text{-normal model set, free over } L, \text{ and } I \subseteq M\}.$$

For any partial L-interpretation I (which cannot mention $=$), $I \cup F_L$ is consistent, trivially L-downward saturated (its members are signed atomic), has the atomic substitution property (because the only statements of the form $\mathrm{T}t = u$ it contains are those for which t and u are identical), and contains all statements $\mathrm{T}t = t$. So, by Proposition 2.5, there is at least one L-normal model set extending it. Such a model set is free over L by Proposition 3.1. Thus the intersection in the definition above is nonempty, and the definition of I^{strong} is meaningful.

For a statement X of L, since $\mathrm{T}X \vee \neg X$ is in every L-model set, I^{strong} will assign to $X \vee \neg X$ "true", though it is easy to produce examples in which I^{strong} assigns X itself no truth value. In this semantics, valid statements will be true, though still not every statement will get a truth value.

The notion above makes the domain closure assumption explicitly: only the language L is involved. The following definition abandons this.

Definition. Again let I be a partial L-interpretation. By the *weak model set closure of* I, denoted I^{weak}, we mean

$$\bigcap \{M \mid \text{for some language } K,\ L \leqslant K,$$
$$M \text{ is a } K\text{-normal model set, free over } L, \text{ and } I \subseteq M\}.$$

We established that the family being intersected to form I^{strong} was nonempty. Members of that family also satisfy the conditions above, so I^{weak} is a well-defined notion. The following statements are easy consequences of the definitions.

Proposition 4.2. *For partial L-interpretations* I *and* J:

(1) $I^{\text{sat}} \subseteq I^{\text{strong}}$ (*because every L-model set is L-upward saturated*);

(2) $I^{\text{weak}} \subseteq I^{\text{strong}}$ (*because every L-model set is a K-model set for some language K with* $L \leqslant K$, *namely for* $K = L$);

(3) $I \subseteq J \Rightarrow I^{\text{sat}} \subseteq J^{\text{sat}}$;

(4) $I \subseteq J \Rightarrow I^{\text{strong}} \subseteq J^{\text{strong}}$;

(5) $I \subseteq J \Rightarrow I^{\text{weak}} \subseteq J^{\text{weak}}$.

The use of both saturated and strong model set closure leads to noncomputable relations (Π^1_1 relations, in fact). This can be shown by an argument similar to that used in [3], embedding ω-elementary formal systems. But we will see that the use of weak closure does not.

5. Lattice background

In the semantical treatment of conventional logic programming, complete lattices are involved. This is not the case here. When one is dealing with subsets of the Herbrand base, consistency issues are trivial; any subset is consistent. But we have negative as well as positive information; F-signed statements as well as T-signed ones. And while the intersection of two consistent sets of signed statements (such as partial L-interpretations) must be consistent, this need not be the case for union. So a complete lattice is too strong a notion. On the other hand, a complete partial ordering is too weak. The just right structure we need is that of complete semi-lattice.

Definition. A *complete semi-lattice* is a structure $\langle \boldsymbol{D}, \leqslant \rangle$ which is a partial ordering that is closed under *arbitrary* infs, and under sups of *directed* subsets. A subset S of $\boldsymbol{D}$ is directed if any two members of S have a common upper bound in S.

Just as in a complete lattice, monotone maps in a complete semi-lattice have smallest fixed points, though they may not have largest ones. We still have the usual generalization of induction: if F is monotone and $F(c) \leqslant c$, then the least fixed point of F is $\leqslant c$. And, as in a complete lattice, one can approximate to the least fixed point of a monotone map F from below by starting with the least member of $\boldsymbol{D}$ and repeatedly applying F, continuing the sequence of approximations into the transfinite if F is not continuous. A monotone map may have several *maximal* fixed points, and it must have a unique largest fixed point that is compatible with every fixed point, where compatible means having a common upper bound. Such a fixed point has been called *intrinsic* in [6] or *optimal* is [8].

Proofs of the results stated above can be found in [3, 4, 8].

6. Program syntax

We abandon Horn clauses as such because we wish to use the full machinery of classical logic in explicit form. We do this in the obvious way.

Definition. A *definition* is an expression of the form

$$R(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n),$$

where R is a relation symbol other than $=$; φ is a formula (possibly $\top$ or $\bot$), and $x_1, \ldots, x_n$ are variables. The definition displayed is *of* R.

A *program* is a finite set of definitions no two of which are of the same relation.

We could relax the requirement that no two definitions in a program can be of the same relation, but we gain nothing (except possibly convenience). The semantical behavior we would want to ascribe to, say,

$$R \leftarrow \varphi, \qquad R \leftarrow \psi$$

would be identical to that of

$$R \leftarrow \varphi \vee \psi.$$

We choose to keep things syntactically simple.

Definition. If every definition in a program P involves only relation symbols and formulas of the language L, we say P is an *L-program.*

For example, the following is an L-program in a language L containing constant symbol 0, function symbol s, relation symbols even, odd, and $=$.

$$\text{even}(x) \leftarrow x = 0 \vee (\exists y)[\text{odd}(y) \wedge x = \text{s}(y)]$$
$$\text{odd}(x) \leftarrow (\exists y)[\text{even}(y) \wedge x = \text{s}(y)].$$

Another example, in the same language, is

$$\text{even}(x) \leftarrow x = 0 \vee (\exists y)[\text{odd}(y) \wedge x = \text{s}(y)]$$
$$\text{odd}(x) \leftarrow (\forall y)[\text{even}(y) \supset \neg(x = y)].$$

Although Horn clause programs are not programs in the present sense, they correspond to some of our programs via Clark's completed data base translation [2]. We briefly sketch the translation. Suppose P is a *Horn clause* program. First, each clause of P, say $R(t_1, \ldots, t_n) \leftarrow B_1, \ldots, B_m$, is replaced by

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

where $x_1, \ldots, x_n$ are new variables, and $y_1, \ldots, y_k$ are all the variables of $t_1, \ldots, t_n$, $B_1, \ldots, B_m$. Then, all rewritten clauses with the same conclusion, say $R \leftarrow D_1$, $\ldots, R \leftarrow D_s$ are replaced by the single expression $R \leftarrow D_1 \vee \cdots \vee D_s$. Also, if R is an n-place relation symbol of L not occurring in the head of any axiom of P, add the expression $R(x_1, \ldots, x_n) \leftarrow \bot$. The result is a program in our sense.

Definition. If P is a conventional Horn clause program, $D(P)$ is the translation of P into a program in the present sense, according to the translation procedure above.

7. Program semantics

We associate a 'meaning' with a program P in three different ways, corresponding to the three closure notions introduced in Section 4. The general techniques are similar in the three cases, and they are treated together.

Definition. $\langle \boldsymbol{P}(L), \subseteq \rangle$ is the space of all partial L-interpretations, ordered by subset.

$\langle \boldsymbol{P}(L), \subseteq \rangle$ is not a complete lattice, but it is a complete semi-lattice as in Section 5, so a monotone map will have a least fixed point. We create a different monotone map for each of the three closure notions discussed in Section 4 but the definitions can be given simultaneously.

Definition. Let P be an L-program. Let C stand for any of the closure operators *sat*, *weak*, or *strong*. For each choice of C an operator $[C]_P : \boldsymbol{P}(L) \to \boldsymbol{P}(L)$ is defined as follows. Let $\boldsymbol{I} \in \boldsymbol{P}(L)$. $[C]_P(\boldsymbol{I})$ is the smallest set such that, for an atomic L-statement $R(t_1, \ldots, t_n)$ (with R not $=$), if there is a definition $R(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n)$ in P, then

$$\mathrm{T}R(t_1, \ldots, t_n) \in [C]_P(\boldsymbol{I}) \quad \text{provided } \mathrm{T}\varphi(t_1, \ldots, t_n) \in \boldsymbol{I}^C,$$

$$\mathrm{F}R(t_1, \ldots, t_n) \in [C]_P(\boldsymbol{I}) \quad \text{provided } \mathrm{F}\varphi(t_1, \ldots, t_n) \in \boldsymbol{I}^C.$$

In other words, the output of $[C]_P(\boldsymbol{I})$ 'says' that an atomic statement R is true (or false) if the program P 'says' that R depends on a statement φ and the input $\boldsymbol{I}$ 'says' φ is true (or false) in the C-semantical sense.

It is straightforward to verify that each of $[\mathrm{sat}]_P$, $[\mathrm{strong}]_P$ and $[\mathrm{weak}]_P$ map $\boldsymbol{P}(L)$ to $\boldsymbol{P}(L)$, and all are monotone by Proposition 4.2(3)–(5). Then all have smallest fixed points. Proposition 4.2(1) gives us that $[\mathrm{sat}]_P(\boldsymbol{I}) \subseteq [\mathrm{strong}]_P(\boldsymbol{I})$, and it easily follows that the smallest fixed point of $[\mathrm{sat}]_P \subseteq$ the smallest fixed point of $[\mathrm{strong}]_P$. Similarly, the smallest fixed point of $[\mathrm{weak}]_P \subseteq$ the smallest fixed point of $[\mathrm{strong}]_P$, using Proposition 4.2(2).

In fact, for all these results it is enough that $\boldsymbol{P}(L)$ be a complete partial ordering. The stronger fact that it is a complete semi-lattice guarantees us so-called *optimal* fixed points as well [8]. These and other fixed points can be used to account for differences between programs such as $P \leftarrow P$ and $P \leftarrow \neg P$, which give rise to identical least fixed points but which do not seem equivalent intuitively. In [3], we investigated this issue for the saturated semantics. We do not consider it here.

Examples. First, suppose P is the following program, where 'test' and A are statements

$$\mathrm{test} \leftarrow A \vee \neg A.$$

It is easy to see that the saturated semantics assigns neither 'test' nor A a truth value. That is, the least fixed point of $[\mathrm{sat}]_P$ is $\emptyset$. On the other hand, both the strong and the weak model set semantics give 'test' the value true, though neither assigns a truth value to A.

Next, a more elaborate example. For convenience we write $s^n(t)$ for $s(s(\ldots(t)\ldots))$ where there are n applications of s, and we refer to $s^n(0)$ as the *number n*. L is the smallest language in which the following is a program. Let P be

$$\begin{aligned}
&\text{even}(x) \leftarrow [x = 0 \vee (\exists y)(\text{even}(y) \wedge x = s^2(y))]\\
&\text{odd}(x) \leftarrow \neg\text{even}(x)\\
&\text{test}(x) \leftarrow \text{even}(x) \vee \text{odd}(x)\\
&\text{all1} \leftarrow (\forall x)[\text{even}(x) \vee \text{odd}(x)]\\
&\text{all2} \leftarrow (\forall x)[\text{even}(x) \vee \neg\text{even}(x)].
\end{aligned}$$

Loosely speaking, all three partial model semantics say each even number is even but not odd, and each odd number is odd but not even. Consequently, all three semantics assign true to $\text{test}(s^n(0))$, for every n. Further, because of the domain closure condition, both the saturated and the strong model set semantics assign true to all1 and to all2, though the presence of the definition for all1 means the operators $[\text{sat}]_P$ and $[\text{strong}]_P$ are not continuous. On the other hand, even though the weak model set semantics does make every number satisfy either $\text{even}(x)$ or $\text{odd}(x)$, it does not 'know' that these are the only objects because it does not assume the domain closure condition. Consequently, the weak model set semantics assigns no truth value to all1. But the weak model set semantics assigns to all2 the value true, simply because $(\forall x)[\text{even}(x) \vee \neg\text{even}(x)]$ is universally valid.

8. Connections

We have provided three semantics for extensions of Horn clause programming, and there is the conventional semantics as well. Now we establish some relationships between these semantics. We begin by stating the main result of this section, a conservativeness result.

Definition. Let P be an L-program, and let C be one of our three closure notions. The *success set for P in the C-semantics* is the set of atomic L-statements A such that TA is in the least fixed point of $[C]_P$.

Theorem 8.1. *Let P be a conventional Horn clause program, and so D(P) is a program in our sense. The success set for D(P) is the same in the saturated, strong model set and weak model set semantics, and coincides with the minimal model for P in the standard Van Emden, Kowalski, Apt semantics.*

This theorem follows from results proved below, relating the various semantics two at a time. We begin by showing that, for certain programs, T-signed output only depends on T-signed input.

Definition. A formula X is *positive* if it contains no negation or implication symbols (the falsehood constant $\perp$ is allowed). A program P is *positive* if every definition in P has a body consisting of a positive formula.

Lemma 8.2. *Suppose A and B are L-saturated sets of signed statements, and every* T*-signed atomic statement in A is also in B. Then every* T*-signed positive statement in A is also in B.*

Proof. By structural induction on statements. Suppose, for instance, that $X \vee Y$ is positive, $\mathrm{T}X \vee Y \in A$, and the result is known for simpler statements. Since $\mathrm{T}X \vee Y \in A$, which is *downward* saturated, $\mathrm{T}X \in A$ or $\mathrm{T}Y \in A$. Each of X and Y is positive, so, by the induction hypothesis, $\mathrm{T}X \in B$ or $\mathrm{T}Y \in B$, and by *upward* saturation, $\mathrm{T}X \vee Y \in B$. The other cases are similar. □

Definition. For a partial L-interpretation $\boldsymbol{I}$, let $\boldsymbol{I}^+ = \{\mathrm{T}X \mid \mathrm{T}X \in \boldsymbol{I}\}$.

Proposition 8.3. *Let P be a positive L-program. Then*:

(1) $[\mathrm{sat}]_P(\boldsymbol{I}^+)$ *and* $[\mathrm{sat}]_P(\boldsymbol{I})$ *have the same* T*-signed members*;

(2) $[\mathrm{strong}]_P(\boldsymbol{I}^+)$ *and* $[\mathrm{strong}]_P(\boldsymbol{I})$ *have the same* T*-signed members*;

(3) $[\mathrm{weak}]_P(\boldsymbol{I}^+)$ *and* $[\mathrm{weak}]_P(\boldsymbol{I})$ *have the same* T*-signed members.*

Proof. We show part (3); the other parts have similar, and easier, proofs. Half is by monotonicity; since $\boldsymbol{I}^+ \subseteq \boldsymbol{I}$, $[\mathrm{weak}]_P(\boldsymbol{I}^+) \subseteq [\mathrm{weak}]_P(\boldsymbol{I})$.

Now suppose A is an atomic L-statement, $\mathrm{T}A \in [\mathrm{weak}]_P(\boldsymbol{I})$, but $\mathrm{T}A \notin [\mathrm{weak}]_P(\boldsymbol{I}^+)$. We derive a contradiction.

Since $\mathrm{T}A \in [\mathrm{weak}]_P(\boldsymbol{I})$, there is a substitution instance $A \leftarrow X$ of some P-definition and $\mathrm{T}X \in \boldsymbol{I}^{\mathrm{weak}}$. And since $\mathrm{T}A \notin [\mathrm{weak}]_P(\boldsymbol{I}^+)$, $\mathrm{T}X \notin (\boldsymbol{I}^+)^{\mathrm{weak}}$. Since P is a positive program, X is a positive statement.

Since $\mathrm{T}X \notin (\boldsymbol{I}^+)^{\mathrm{weak}}$, for some language K with $L \leqslant K$, there is a K-normal model set $\boldsymbol{M}$, free over L, with $\boldsymbol{I}^+ \subseteq \boldsymbol{M}$ but $\mathrm{T}X \notin \boldsymbol{M}$. Define a set $\boldsymbol{B}$ to be

$$\boldsymbol{I}^+ \cup \{\mathrm{F}Y \mid Y \text{ is an atomic } K\text{-statement not involving } =, \top, \text{ or } \perp, \text{ and } \mathrm{T}Y \notin \boldsymbol{I}\} \cup \boldsymbol{F}_K.$$

$\boldsymbol{B}$ is consistent, atomically K-complete, and trivially has the atomic substitution property. Then $\boldsymbol{B}^{\mathrm{U}}$ is a K-normal model set. Also it is easy to see $\boldsymbol{B}^{\mathrm{U}}$ is free over L, and $\boldsymbol{I} \subseteq \boldsymbol{B}^{\mathrm{U}}$. Further, every T-signed atomic statement of $\boldsymbol{B}^{\mathrm{U}}$ must be in $\boldsymbol{B}$ and hence in $\boldsymbol{M}$. It follows from Lemma 8.2 that $\mathrm{T}X \notin \boldsymbol{B}^{\mathrm{U}}$. But this is impossible since $\mathrm{T}X \in \boldsymbol{I}^{\mathrm{weak}}$. □

Next we show that the strong model set semantics and the saturated semantics assign the same success set to a positive program. This is a consequence of the following proposition.

Proposition 8.4. *Let P be a positive program. For a partial L-interpretation* $\boldsymbol{I}$, $[\mathrm{sat}]_P(\boldsymbol{I})$ *and* $[\mathrm{strong}]_P(\boldsymbol{I})$ *both have the same* T*-signed members.*

Proof. The argument is similar to that for Proposition 8.3. As was remarked in Section 7, $[\mathrm{sat}]_P(\boldsymbol{J}) \subseteq [\mathrm{strong}]_P(\boldsymbol{J})$ for any $\boldsymbol{J}$, which gives half the result. Conversely, suppose $\mathrm{T}A \in [\mathrm{strong}]_P(\boldsymbol{I})$; we show $\mathrm{T}A \in [\mathrm{sat}]_P(\boldsymbol{I})$.

Since $\mathrm{T}A \in [\mathrm{strong}]_P(\boldsymbol{I})$ there is a substitution instance of an axiom in P, $A \leftarrow X$, with $\mathrm{T}X \in \boldsymbol{I}^{\mathrm{strong}}$. Since P is a positive program, X is a positive statement. Now, define a set $\boldsymbol{B}$ to be

$$\boldsymbol{I}^+ \cup \{\mathrm{F}Y \mid Y \text{ is an atomic } L\text{-statement not involving } =, \top, \text{ or } \bot, \text{ and } \mathrm{T}Y \notin \boldsymbol{I}\}.$$

$\boldsymbol{B}$ is also a partial L-interpretation, $\boldsymbol{I} \subseteq \boldsymbol{B}$, and $\boldsymbol{I}^+ = \boldsymbol{B}^+$.

Now, every T-signed atomic L-statement in $\boldsymbol{B} \cup \boldsymbol{F}_L$ is in $\boldsymbol{B}^+ \cup \boldsymbol{F}_L = \boldsymbol{I}^+ \cup \boldsymbol{F}_L \subseteq \boldsymbol{I} \cup \boldsymbol{F}_L$. Then, by Lemma 8.2, every T-signed positive statement of L in $(\boldsymbol{B} \cup \boldsymbol{F}_L)^{\mathrm{U}}$ is in $(\boldsymbol{I} \cup \boldsymbol{F}_L)^{\mathrm{U}}$. And, by Proposition 4.1, $(\boldsymbol{I} \cup \boldsymbol{F}_L)^{\mathrm{U}} = \boldsymbol{I}^{\mathrm{sat}}$. Further, since $\boldsymbol{B} \cup \boldsymbol{F}_L$ is atomically L-complete, $(\boldsymbol{B} \cup \boldsymbol{F}_L)^{\mathrm{U}}$ is an L-model set by Proposition 2.2. It is L-normal by Proposition 2.4, and free over L by Proposition 3.1. Then, since $\mathrm{T}X \in \boldsymbol{I}^{\mathrm{strong}}$, it follows that $\mathrm{T}X \in (\boldsymbol{B} \cup \boldsymbol{F}_L)^{\mathrm{U}}$, hence, $\mathrm{T}X \in \boldsymbol{I}^{\mathrm{sat}}$, and $\mathrm{T}A \in [\mathrm{sat}]_P(\boldsymbol{I})$. □

Corollary 8.5. *The success set for a positive program is the same in the strong model and the saturated semantics.*

Proof. As was pointed out in Section 5, the 'usual' way of approximating the least fixed point of a monotonic operator applies in a complete *semi*-lattice as well as in a complete lattice. For a monotone map $\boldsymbol{\Phi}$ on $\langle \boldsymbol{P}(L), \subseteq \rangle$, define an ordinally indexed sequence as follows:

$$\boldsymbol{\Phi}^0 = \emptyset, \qquad \boldsymbol{\Phi}^{\alpha+1} = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\alpha),$$

$$\boldsymbol{\Phi}^\lambda = \bigcup_{\alpha<\lambda} \boldsymbol{\Phi}^\alpha \quad \text{for limit ordinals } \lambda.$$

Then the least fixed point of $\boldsymbol{\Phi}$ is the limit of the $\boldsymbol{\Phi}^\alpha$ sequence as α increases, and the limit is actually attained at some (closure) ordinal, after which the sequence remains constant. Consequently, if we prove that, for a positive program P and for each ordinal α, $([\mathrm{sat}]_P)^\alpha$ and $([\mathrm{strong}]_P)^\alpha$ have the same T-signed members, it will establish the theorem. This done by induction on α. The $\alpha = 0$ case is trivial, as is the limit ordinal case. And the successor ordinal case easily follows from Propositions 8.3 and 8.4. □

Next we turn to the relationship between the strong and the weak model set semantics.

Definition. The *existential formulas* of L are the members of the smallest set S of L-formulas such that:

(1) all signed atomic L-formulas are in S;
(2) $\alpha_1, \alpha_2 \in S \Rightarrow \alpha \in S$;
(3) $\beta_1, \beta_2 \in S \Rightarrow \beta \in S$;
(4) $\delta(x) \in S \Rightarrow \delta \in S$.

The idea is that in an existential formula all existential quantifiers are in 'positive' locations and all universal quantifiers are in 'negative' locations. Then all quantifiers act like existential ones.

It follows from the definition that the set of existential formulas of L meets conditions (2)-(4) in 'iff'-form.

Lemma 8.6. *Suppose I and J are partial L-interpretations with $I \subseteq J$, and $L \leqslant K$. Let I^U be the L-upward saturated closure of I, but let J^U be the K-upward saturated closure of J. Then, if Z is a signed existential statement of L, $Z \in I^U \Rightarrow Z \in J^U$.*

Proof. By structural induction on Z. We consider one case. Suppose Z is some signed, existential L-statement of type δ, and the result is known for statements simpler than Z. Suppose $Z \in I^U$, i.e., $\delta \in I^U$. By L-downward saturation of I^U, $\delta(t) \in I^U$ for some closed L-term t. Since δ is existential, so is $\delta(t)$; so $\delta(t) \in J^U$ by the induction hypothesis. Since $L \leqslant K$, t is also a closed K-term, hence $\delta \in J^U$ by K-upward saturation. □

Lemma 8.7. *Let I be a partial L-interpretation. Every signed, existential L-statement of I^{strong} is in I^{weak}.*

Proof. Suppose Z is a signed, existential L-statement, but $Z \notin I^{\text{weak}}$. Then, for some language K with $L \leqslant K$, there is a K-normal model set $\boldsymbol{M}$, free over L, with $I \subseteq \boldsymbol{M}$, but $Z \notin \boldsymbol{M}$.

Let $\boldsymbol{B}$ be the collection of all signed, atomic L-statements that are in $\boldsymbol{M}$. $\boldsymbol{B}$ is atomically L-complete, so if $\boldsymbol{B}^U$ is the L-upward saturated closure of $\boldsymbol{B}$, $\boldsymbol{B}^U$ is an L-model set. It is easy to see that $\boldsymbol{B}^U$ is L-normal ($\boldsymbol{B}$ has the atomic substitution property), is free over L, and $I \subseteq \boldsymbol{B}^U$.

Let $\boldsymbol{M}^U$ be the K-upward saturated closure of $\boldsymbol{M}$. Trivially, $\boldsymbol{M}^U = \boldsymbol{M}$. Then by Lemma 8.6, since $Z \notin \boldsymbol{M}$, it follows that $Z \notin \boldsymbol{B}^U$ and hence $Z \notin I^{\text{strong}}$. □

Definition. A program P is *existential* if, for any definition $X \leftarrow B$ in P, the signed formula $\mathrm{T}B$ is existential.

Proposition 8.8. *Suppose P is an existential L-program and I is a partial L-interpretation. Then* $[\text{weak}]_P(I)$ *and* $[\text{strong}]_P(I)$ *have the same* T*-signed members.*

Proof. Half is by the general fact that $[\text{weak}]_P(I) \subseteq [\text{strong}]_P(I)$. Conversely, suppose $\mathrm{T}A \in [\text{strong}]_P(I)$. Then there is some substitution instance $A \leftarrow X$ of a

definition in P, with $TX \in \boldsymbol{I}^{\text{strong}}$. Since P is an existential program, TX is an existential statement, hence $TX \in \boldsymbol{I}^{\text{weak}}$ by Lemma 8.7. Then $TA \in [\text{weak}]_P(\boldsymbol{I})$. □

Corollary 8.9. *The success set for a positive, existential program is the same in the strong model set and the weak model set semantics.*

Proof. Like that of Corollary 8.5. □

By combining Corollary 8.5 and Corollary 8.9, all three partial model semantics assign the same success set to a positive, existential program. If P is a conventional Horn clause program, $D(P)$ will be positive and existential. The final items needed to finish the proof of Theorem 8.1 are straightforward and are stated below, with proof omitted. The $T\uparrow$ and $T\downarrow$ notation is from [1].

Proposition 8.10. *Let P be a conventional Horn clause program, and let* T *be the corresponding operator. Then, for each ordinal α,*

(1) $T\uparrow\alpha = \{A \mid TA \in [\text{sat}]^{\alpha}_{D(P)}(\emptyset)\}$;

(2) $T\downarrow\alpha = U - \{A \mid FA \in [\text{sat}]^{\alpha}_{D(P)}(\emptyset)\}$, *where U is the Herbrand base.*

It follows that the success set for $D(P)$ in the saturated semantics is the smallest fixed point of the T*-operator for P.*

9. Tableaus

For the rest of the paper we concentrate on the weak model set semantics. Specifically, we define a proof procedure based on semantic tableaus allowing 'recursive' calls, and we prove its soundness and completeness relative to the weak model set semantics. An easy consequence is the continuity of the $[\text{weak}]_P$ operator.

As background for Section 10 we sketch the tableau system for classical first-order logic, as given in [10]. Then we present extra rules to deal with equality. The semantic tableau method is essentially equivalent to that of Gentzen sequents, though we find the tableau style more natural in the present setting. A discussion of the relationship between tableaus and Gentzen systems for classical logic may be found in [10, Chapter XI].

Definition. For a given language L, we mean by L^* the language that results from L when a designated, countable set of new constant symbols is added to L. Then, $L \leqslant L^*$ of course. The constant symbols of L^* that are not in L are referred to as *parameters.*

Tableau proofs of statements of L will be trees labelled with signed statements from the larger language L^*.

Branch extension rules

$$\frac{\alpha}{\begin{array}{l}\alpha_1\\ \alpha_2\end{array}} \qquad \begin{array}{ccc} & \beta & \\ \swarrow & & \searrow \\ \beta_1 & & \beta_2 \end{array}$$

$$\frac{\gamma}{\gamma(t)} \quad \text{for any closed term } t \text{ of } L^*,$$

$$\frac{\delta}{\delta(t)} \quad \text{for a } \textit{parameter } t \text{ hitherto } \textit{unused} \text{ in the tree.}$$

Definitions. A branch of a tableau is *closed* if it has TX and FX on it for some statement X, or if it has T$\perp$ or F$\top$ on it. A branch is *open*, or *unclosed*, if it is not closed. A tableau is *closed* if every branch is closed.

A *tableau for* a signed statement Z of L is defined as follows:

(1) the one branch tree with a single node labelled Z is a tableau for Z;

(2) if we have a tableau for Z, nondeterministically choose an unclosed branch, choose a signed statement on it, and extend the branch by applying the appropriate branch extension rule. The result is another tableau for Z.

If X is a statement of L, a *proof* of X is a closed tableau for FX. A *disproof* of X is a closed tableau for TX.

Informally, identify a tableau branch with the conjunction of the signed statements on it, and a tableau with the disjunction of its branches. Then one may think of a tableau as a *constraint* on a model. Tableau proofs are *refutation* arguments. To prove X, start with FX and generate a closed tableau (which is a constraint no model can satisfy). Then, informally at least, X could not have been false in any model, i.e., X is valid. This can be made the basis of a formal soundness proof.

A version of the completeness theorem for tableaus can be stated as follows: *a statement X of L has a proof provided* TX *is in every L^*-model set.*

Next we add general branch extension rules for equality.

Reflexivity rule: In a tableau for some signed L statement, T$t = t$ can be added to the end of any branch, for any closed term t of L^*.

Substitution rule: Let Z be a signed statement of L^* and let Z' be like Z except that zero or more occurrences of the closed term t in Z have been replaced by occurrences of the closed term u. If T$t = u$ and Z occur on a branch, Z' may be added to the end of it.

We did not assume symmetry or transitivity rules, but it is easy to see that they follow as derived rules. That is, if T$t = u$ occurs on a branch, the rules above will allow us to add T$u = t$ to the branch. Similarly, for transitivity.

Definition. We say we have a *normal* proof (or disproof) of X if we have constructed a closed tableau for FX (or TX) allowing the two additional equality rules above.

Completeness can be given the following form: *A statement X of L has a normal proof provided* TX *is in every L^*-normal model set.*

10. Recursive tableau calls

Now we present our additions to the tableau machinery to deal with programs. Let L be a fixed language, and let P be an L-program. We define the notions of *P-tableau*, *P-proof*, and *P-disproof*. To begin with, all the tableau terminology given in Section 9 carries over to the P-case. Thus, a closed normal P-tableau for FX constitutes a P-proof of X, and so on. The only addition is that we have some extra branch closure and extension rules for P-tableaus.

First, we add closure and extension rules to take care of the free interpretation we are giving equality.

Free interpretation rules: A P-tableau branch is closed if it contains:

(1) T$c = d$ for distinct constant symbols c and d of L;

(2) T$f(t_1, \ldots, t_n) = c$ for a function symbol f and a constant symbol c of L;

(3) T$f(t_1, \ldots, t_n) = g(u_1, \ldots, u_m)$ for distinct function symbols f and g of L.

A branch containing T$f(t_1, \ldots, t_n) = f(u_1, \ldots, u_n)$ may be extended by adding T$t_i = u_i$ to the end, for any $i = 1, 2, \ldots, n$.

Finally, we present the rules that take program P specifically into account.

Recursive program rules: Let $R(t_1, \ldots, t_n)$ be an atomic statement of L.

(1) A branch of a P-tableau is closed if it contains F$R(t_1, \ldots, t_n)$, there is a definition $R(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n)$ in P, and $\varphi(t_1, \ldots, t_n)$ has a P-tableau proof.

(2) A branch of a P-tableau is closed if it contains T$R(t_1, \ldots, t_n)$, there is a definition $R(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n)$ in P, and $\varphi(t_1, \ldots, t_n)$ has a P-tableau disproof.

Definition. For a program P and an L-statement Z, we say *a query of P with Z succeeds* if Z has a P-tableau proof, and *a query of P with Z fails* if Z has a P-tableau disproof. (Z need not be atomic, though, in the interests of simplicity, atomic queries are all we will consider here.)

It will follow from the soundness result in Section 11 that no query can both succeed and fail. It is possible, however, for a query to do neither. Partial models are essential.

Example. P is the program

$$\text{even}(x) \leftarrow x = 0 \vee (\exists y)[\text{odd}(y) \wedge x = \text{s}(y)]$$
$$\text{odd}(x) \leftarrow (\forall y)[\text{even}(y) \supset \neg(x = y)].$$

We show that a query of P with even(s(0)) fails. That is, even(s(0)) has a P-tableau disproof; there is a closed P-tableau beginning with Teven(s(0)). Now, the only applicable rule is recursive program rule (2). To close the tableau we need a closed P-tableau for $\mathrm{Ts}(0) = 0 \vee (\exists y)[\mathrm{odd}(y) \wedge \mathrm{s}(0) = \mathrm{s}(y)]$. Such a tableau begins as follows. The numbers are not part of the tableau, but are for reference only.

$$\mathrm{Ts}(0) = 0 \vee (\exists y)[\mathrm{odd}(y) \wedge \mathrm{s}(0) = \mathrm{s}(y)] \quad (1)$$

$$\mathrm{Ts}(0) = 0 \quad (2) \qquad \mathrm{T}(\exists y)[\mathrm{odd}(y) \wedge \mathrm{s}(0) = \mathrm{s}(y)] \quad (3)$$

$$\mathrm{Todd}(p) \wedge \mathrm{s}(0) = \mathrm{s}(p) \quad (4)$$

$$\mathrm{Todd}(p) \quad (5)$$

$$\mathrm{Ts}(0) = \mathrm{s}(p) \quad (6)$$

$$\mathrm{T}0 = p \quad (7)$$

$$\mathrm{Todd}(0) \quad (8)$$

Reasons: (2) and (3) are from (1) by the β-branch extension rule; (4) is from (3) by the δ-rule (p is a parameter); (5) and (6) are from (4) by α; (7) is from (6) by a free interpretation rule; (8) is from (5) and (7) by the substitution rule, tacitly making use of symmetry.

The left branch is closed because of (2) and free interpretation rule (2). By recursive program rule (2) applied to (8), the right branch, and hence the tableau, would be closed if we had a P-tableau disproof of $(\forall y)[\mathrm{even}(y) \supset \neg(0 = y)]$. Such a P-tableau follows.

$$\mathrm{T}(\forall y)[\mathrm{even}(y) \supset \neg(0 = y)] \quad (9)$$

$$\mathrm{Teven}(0) \supset \neg(0 = 0) \quad (10)$$

$$\mathrm{Feven}(0) \quad (11) \qquad \mathrm{T}\neg(0 = 0) \quad (12)$$

$$\mathrm{F}(0 = 0) \quad (13)$$

$$\mathrm{T}(0 = 0) \quad (14)$$

Reasons: (10) is from (9) by the γ-rule; (11) and (12) are from (10) by β; (13) is from (12), and (14) is by the reflexivity rule. The right branch is closed because of (13) and (14). Finally, the left branch would be closed if we had a closed P-tableau for $\mathrm{F}0 = 0 \vee (\exists y)[\mathrm{odd}(y) \wedge 0 = \mathrm{s}(y)]$. But a tableau for this quickly closes using the α-rule and the reflexive rule.

11. Tableau maps

We introduce a nonrecursive tableau system meant to capture a single application of the weak semantic operator $[\mathrm{weak}]_P$ associated with program P. We use it in the next section to derive the soundness and completeness of the P-tableau system of

Section 10 relative to the least fixed point semantics of $[\text{weak}]_P$. Continuity of $[\text{weak}]_P$ will be another byproduct.

Let L be a language that is fixed for this section.

Definition. For an L-program P and a partial L-interpretation $\boldsymbol{I}$, by a *P-I-tableau* we mean a tableau meeting the conditions for a P-tableau as given in Section 10, except that recursive program rules (1) and (2) are replaced by

(1′) a branch of a *P-I*-tableau is closed if it contains $\mathrm{F}R(t_1, \ldots, t_n)$, where $\mathrm{T}R(t_1, \ldots, t_n) \in \boldsymbol{I}$;

(2′) a branch of a *P-I*-tableau is closed if it contains $\mathrm{T}R(t_1, \ldots, t_n)$, where $\mathrm{F}R(t_1, \ldots, t_n) \in \boldsymbol{I}$.

We can use *P-I*-tableaus to define yet another mapping on $\langle \boldsymbol{P}(L), \subseteq \rangle$, the space of partial L-interpretations, in a straightforward way. We call it the *tableau mapping.*

Definition. $[\text{tab}]_P: \boldsymbol{P}(L) \to \boldsymbol{P}(L)$ is defined as follows. Let $\boldsymbol{I} \in \boldsymbol{P}(L)$. Then,

$$[\text{tab}]_P(\boldsymbol{I}) = \{\mathrm{T}R(t_1, \ldots, t_n) \mid R(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n) \text{ is in } P \text{ and there is a closed } P\text{-}\boldsymbol{I}\text{-tableau for } \mathrm{F}\varphi(t_1, \ldots, t_n)\}$$

$$\cup \{\mathrm{F}R(t_1, \ldots, t_n) \mid R(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n) \text{ is in } P \text{ and there is a closed } P\text{-}\boldsymbol{I}\text{-tableau for } \mathrm{T}\varphi(t_1, \ldots, t_n)\}.$$

The primary result of this section is easily stated.

Theorem 11.1. $[\text{weak}]_P = [\text{tab}]_P$.

The proof is broken into two arguments, a soundness and a completeness half. We begin with soundness. Note that in the definition below, an extra language K is brought in. This is to take care of the δ-rule. Informally, if δ is true in a model, some instance must be true, but that instance may not have a name in the original language.

Definition. Let $\boldsymbol{I}$ be a partial L-interpretation and let S be a set of signed statements of L^*. Call S *I-satisfiable* if there is a language K with $L \leqslant K$, and a K-normal model set $\boldsymbol{M}$, free over L, with $\boldsymbol{I} \cup S \subseteq \boldsymbol{M}$. Call a branch of a tableau *I*-satisfiable if the set of signed statements on it is *I*-satisfiable. And call a tableau *I*-satisfiable if some branch is.

Lemma 11.2. *The result of applying any branch extension rule to an I-satisfiable P-I-tableau yields another I-satisfiable P-I-tableau.*

Proof. The α- and β-rules are trivial. We consider the δ-rule in detail and leave the other rules to the reader.

Suppose we have a *P-I*-tableau with an *I*-satisfiable branch ϑ having δ on it, and we add $\delta(c)$ to the end of ϑ, where c is a parameter hitherto unused in the tableau. Let S be the set of signed statements on ϑ. Then the situation is this. S is *I*-satisfiable, $\delta \in S$, and c is a parameter not occurring in S; we must show $S \cup \{\delta(c)\}$ is *I*-satisfiable.

Since S is *I*-satisfiable, there is some K-normal model set $\boldsymbol{M}$, with $L \leqslant K$, $\boldsymbol{M}$ free over L, and with $\boldsymbol{I} \cup S \subseteq \boldsymbol{M}$. It is possible that c is a constant symbol of K and so already has a 'role' in $\boldsymbol{M}$. If so, we remove it as outlined in the next paragraph. If not, we skip this step and go on from the paragraph following the next.

Choose a constant symbol d not of K and let $K^\dagger$ be the language like K, but with c removed and d added. Since c was a parameter, hence not in L, we still have $L \leqslant K^\dagger$. Likewise let $\boldsymbol{M}^\dagger$ be the set of signed statements that results from the replacement of all occurrences of c in $\boldsymbol{M}$ by occurrences of d. Trivially, $\boldsymbol{M}^\dagger$ is $K^\dagger$-normal and free over L. Finally, c did not occur in S, and c was a parameter and so could not occur in $\boldsymbol{I}$, so the replacement of c by d leaves $\boldsymbol{I}$ and S unchanged. Then $\boldsymbol{I} \cup S \subseteq \boldsymbol{M}^\dagger$.

From now on we may assume the following: $\delta \in S$, there is a $K^\dagger$-normal model set $\boldsymbol{M}^\dagger$ with $L \leqslant K^\dagger$, free over L, with $\boldsymbol{I} \cup S \subseteq \boldsymbol{M}^\dagger$, and c is a parameter not occurring in S and not in the language $K^\dagger$. We still must show $S \cup \{\delta(c)\}$ is *I*-satisfiable.

Since $\delta \in S \subseteq \boldsymbol{M}^\dagger$ and $\boldsymbol{M}^\dagger$ is a model set, for some closed term t of $K^\dagger$, we have $\delta(t) \in \boldsymbol{M}^\dagger$. Let $K^{\dagger\dagger}$ be the language like $K^\dagger$, but with c added as an extra constant symbol. Trivially, $L \leqslant K^{\dagger\dagger}$. Let $\boldsymbol{M}^{\dagger\dagger}$ be the result of enlarging $\boldsymbol{M}^\dagger$ by adding every signed statement obtained by replacing zero or more occurrences of t by occurrences of c in any signed statement of $\boldsymbol{M}^\dagger$. It is not hard to see that $\boldsymbol{M}^{\dagger\dagger}$ is a $K^{\dagger\dagger}$-normal model set, and that $\boldsymbol{M}^{\dagger\dagger}$ is also free over L. Certainly, $\boldsymbol{I} \cup S \subseteq \boldsymbol{M}^{\dagger\dagger}$, but also $\delta(c) \in \boldsymbol{M}^{\dagger\dagger}$ too. Thus $S \cup \{\delta(c)\}$ is *I*-satisfiable. □

Lemma 11.3. *A closed P-I-tableau is not I-satisfiable.*

Proof. Straightforward. □

Proposition 11.4 (*soundness*). $[\text{tab}]_P(\boldsymbol{I}) \subseteq [\text{weak}]_P(\boldsymbol{I})$.

Proof. Let $\boldsymbol{I}$ be a partial L-interpretation, and suppose $\mathbf{T}R(t_1, \ldots, t_n) \in [\text{tab}]_P(\boldsymbol{I})$, but $\mathbf{T}R(t_1, \ldots, t_n) \notin [\text{weak}]_P(\boldsymbol{I})$; we derive a contradiction. (The F-signed case is similar.)

Say the definition for R in P is $R(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n)$. Then, by the first supposition, there is a closed *P-I*-tableau for $\mathbf{F}\varphi(t_1, \ldots, t_n)$. By the second supposition, $\mathbf{T}\varphi(t_1, \ldots, t_n) \notin \boldsymbol{I}^{\text{weak}}$, so there is some K-normal model set $\boldsymbol{M}$ with $L \leqslant K$,

free over L, with $\boldsymbol{I} \subseteq \boldsymbol{M}$, but $\mathrm{T}\varphi(t_1, \ldots, t_n) \notin \boldsymbol{M}$. Since $\boldsymbol{M}$ is a model set, and hence complete, $\mathrm{F}\varphi(t_1, \ldots, t_n) \in \boldsymbol{M}$. Then $\{\mathrm{F}\varphi(t_1, \ldots, t_n)\}$ is $\boldsymbol{I}$-satisfiable.

A P-$\boldsymbol{I}$-tableau for $\mathrm{F}\varphi(t_1, \ldots, t_n)$ begins with the one branch, one node tree, whose node is labelled $\mathrm{F}\varphi(t_1, \ldots, t_n)$. By the previous paragraph, this is an $\boldsymbol{I}$-satisfiable tableau, so, by Lemma 11.2, every subsequent tableau will also be $\boldsymbol{I}$-satisfiable. Then the existence of a closed P-$\boldsymbol{I}$-tableau for $\mathrm{F}\varphi(t_1, \ldots, t_n)$ contradicts Lemma 11.3. □

Proposition 11.5 (*completeness*). $[\mathrm{weak}]_P(\boldsymbol{I}) \subseteq [\mathrm{tab}]_P(\boldsymbol{I})$.

Proof. Let $\boldsymbol{I}$ be a partial L-interpretation, and suppose $\mathrm{T}R(t_1, \ldots, t_n) \notin [\mathrm{tab}]_P(\boldsymbol{I})$. We show $\mathrm{T}R(t_1, \ldots, t_n) \notin [\mathrm{weak}]_P(\boldsymbol{I})$. (Again the F-signed case is similar.)

Say the definition for R in P is $R(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n)$. Then the supposition is, there is no closed P-$\boldsymbol{I}$-tableau for $\mathrm{F}\varphi(t_1, \ldots, t_n)$. Now, tableau construction is inherently nondeterministic, but there are many systematic, deterministic routines one can follow for constructing tableaus, which will ensure that any applicable branch extension rule will eventually be applied (a *fair* procedure, in other words). One simple such systematic procedure is given in [10] for the classical tableau case, and can easily be adapted to the present setting. We omit details.

So, construct a sequence of tableaus for $\mathrm{F}\varphi(t_1, \ldots, t_n)$ following a systematic procedure under which every applicable rule is eventually applied. The procedure can never terminate, for it can only do so by producing a closed tableau for $\mathrm{F}\varphi(t_1, \ldots, t_n)$. Thus an infinite tableau, and hence an infinite branch, is being generated (König's Lemma is used here).

Let S be the set of signed statements on one such infinite branch ϑ. S will be consistent and because of the systematic construction, S will be L^*-downward saturated, will contain all $\mathrm{T}t = t$ for closed terms t of L^*, and will have the atomic (indeed the full) substitution property.

Let $S^\dagger$ be the result of adding to S all signed statements of L^* of the form $\mathrm{F}t = u$, where $\mathrm{T}t = u$ is not in S. Trivially, $S^\dagger$ is still consistent, and is L^*-downward saturated since only signed atomic statements have been added.

Suppose c and d are distinct constant symbols of L. Then $\mathrm{T}c = d$ cannot be in S, or else branch ϑ would have closed at some finite stage using a free interpretation rule. Then, $\mathrm{F}c = d \in S^\dagger$. Similar considerations show that any L^*-model set that extends $S^\dagger$ will be free over L.

Also the atomic substitution property still holds for $S^\dagger$ by the following reasoning. Suppose $\mathrm{T}t = u \in S^\dagger$ and $Z' \notin S^\dagger$, where Z' results from Z by the replacement of some occurrences of t by occurrences of u; we show $Z \notin S^\dagger$. Since $S^\dagger$ resulted from the addition of F-signed statements to S, $\mathrm{T}t = u \in S$. If Z is T-signed, the conclusion follows using the fact that S had the (atomic) substitution property. Now suppose Z is $\mathrm{F}Y$, and Z' is $\mathrm{F}Y'$. Since $Z' \notin S'^\dagger$, $\mathrm{T}Y' \in S$. Using symmetry, $\mathrm{T}u = t \in S$. But then, by substitutivity (of t for u), $\mathrm{T}Y \in S$, hence $Z \notin S^\dagger$.

Now, let $\boldsymbol{I}/S^\dagger$ be the collection of all the substitutional variants of members of $\boldsymbol{I}$ that $S^\dagger$ allows. More precisely, $\boldsymbol{I}/S^\dagger$ is the smallest set of signed atomic statements

such that (i) $I \subseteq I/S^\dagger$, and (ii) $Z \in I/S^\dagger$, $Tt = u \in S^\dagger$, Z' is the result of replacing an occurrence of t in Z by an occurrence of $u \Rightarrow Z' \in I/S^\dagger$.

We are interested in the set $S^\dagger \cup I/S^\dagger$, and will establish some basic facts concerning it.

$S^\dagger \cup I/S^\dagger$ is trivially L^*-downward saturated since $S^\dagger$ was, and $I/S^\dagger$ contains only signed atomic statements. Also $S^\dagger \cup I/S^\dagger$ has the atomic substitution property by the following argument. Suppose $Tt = u \in S^\dagger \cup I/S^\dagger$. Then, in fact, $Tt = u \in S^\dagger$ since members of I cannot contain the equality symbol. Then a substitution replacing an occurrence of t by u in an atomic member of $S^\dagger$ will produce a member of $S^\dagger$ since it has the atomic substitution property, and replacement in a member of $I/S^\dagger$ produces another member of $I/S^\dagger$ by definition. The final trivial result is that any L^*-model set that extends $S^\dagger \cup I/S^\dagger$ will be free over L since this is the case for extensions of $S^\dagger$.

It remains for us to show that $S^\dagger \cup I/S^\dagger$ is consistent, which requires some work. Suppose we do not have consistency, say $TA(d_1, \ldots, d_h)$, $FA(d_1, \ldots, d_h) \in S^\dagger \cup I/S^\dagger$. We derive a contradiction. Since $S^\dagger$ was consistent, it must be that one or both of $TA(d_1, \ldots, d_h)$, $FA(d_1, \ldots, d_h)$ is in $I/S^\dagger$. We will treat the case that both are; the other cases are similar, but easier.

So, suppose $TA(d_1, \ldots, d_h)$, $FA(d_1, \ldots, d_h) \in I/S^\dagger$. Then there must be a signed atomic statement $TA(\tau_1, \ldots, \tau_h) \in I$ and a sequence of equality statements, $Tt_1 = u_1$, $Tt_2 = u_2, \ldots, Tt_k = u_k$, all in S, such that by starting with $TA(\tau_1, \ldots, \tau_h)$ and successively replacing occurrences of t_1 by u_1, t_2 by $u_2, \ldots, t_k$ by u_k, we wind up with $TA(d_1, \ldots, d_h)$. Also there must be $FA(\eta_1, \ldots, \eta_h) \in I$ and $Tv_1 = w_1$, $Tv_2 = w_2, \ldots, Tv_m = w_m \in S$ such that successively replacing v_i by w_i starting with $FA(\eta_1, \ldots, \eta_h)$ yields $FA(d_1, \ldots, d_h)$. We will show that τ_1 and η_1 are identical, $\ldots$, τ_h and η_h are identical, which implies I is inconsistent. This is impossible since I was a partial L-interpretation, which must be consistent.

Consider τ_1 and η_1. By starting with τ_1 and successively replacing t_1 by $u_1, \ldots, t_k$ by u_k, τ_1 can be turned into d_1. Then starting with d_1 and working backward, replacing u_k by $t_k, \ldots, u_1$ by t_1, d_1 can be turned into τ_1. Similarly, by replacing w_m by $v_m, \ldots, w_1$ by v_1, d_1 can be turned into η_1. But $Tt_i = u_i \in S$, hence is on branch ϑ, and hence $Tu_i = t_i$ is on ϑ (since the tableau construction was systematic and symmetry is a derived rule). Also, $Td_1 \doteq d_1$ is on ϑ. Then, using the substitution rule, $T\tau_1 = \eta_1$ must be on ϑ too. But τ_1 and η_1 are closed terms of L since they occur in I. If τ_1 and η_1 were distinct, the free interpretation rules for equality would have allowed us to close branch ϑ. Consequently, τ_1 is identical to η_1. Similarly for τ_2 and η_2, and so on. It follows that I is inconsistent, and we have a contradiction. Conclusion: $S^\dagger \cup I/S^\dagger$ is consistent.

So $S \cup I$ is a subset of a consistent set $S^\dagger \cup I/S^\dagger$ having the atomic substitution property, L^*-downward saturated, and with $Tt = t$ present for all closed L^*-terms t. By Proposition 2.5 this set can be further extended to an L^*-normal model set M, which must be free over L. Since every branch in a tableau goes through the origin, which is labelled $F\varphi(t_1, \ldots, t_n)$, it follows that $F\varphi(t_1, \ldots, t_n) \in S$, and so

$F\varphi(t_1, \ldots, t_n) \in \boldsymbol{M}$. Then $T\varphi(t_1, \ldots, t_n) \notin \boldsymbol{M}$, $\boldsymbol{M}$ is L^*-normal, free over L, and $L \leq L^*$, so $T\varphi(t_1, \ldots, t_n) \notin \boldsymbol{I}^{\text{weak}}$. But then $TR(t_1, \ldots, t_n) \notin [\text{weak}]_P(\boldsymbol{I})$. □

12. Continuity, soundness and correctness

Let L be a language and let P be an L-program, fixed for this section. We return to the recursive P-tableau system of Section 10, and the fixed point semantics of $[\text{weak}]_P$.

Proposition 12.1. $[\text{weak}]_P$ *is continuous.*

Proof. Continuity is often defined as meaning preserving directed unions but, given monotonicity, it is well known to be equivalent to the following finiteness property: for a signed atomic statement Z, $Z \in [\text{weak}]_P(\boldsymbol{I}) \Rightarrow Z \in [\text{weak}]_P(\boldsymbol{I}_0)$ for some finite $\boldsymbol{I}_0 \subseteq \boldsymbol{I}$. By the results of Section 11 it is enough to show a similar result for the operator $[\text{tab}]_P$. But this is simple. If $Z \in [\text{tab}]_P(\boldsymbol{I})$ it is because a closed P-$\boldsymbol{I}$-tableau exists for the signed statement φ that program P associates with Z. A closed tableau, being finite, only uses a finite part of $\boldsymbol{I}$, say $\boldsymbol{I}_0$. Then we have a closed P-$\boldsymbol{I}_0$-tableau for φ, so $Z \in [\text{tab}]_P(\boldsymbol{I}_0)$. □

Proposition 12.2 (completeness). *Let A be an atomic statement of L. If TA is in the least fixed point of* $[\text{weak}]_P$, *then a query of P with A succeeds. If FA is in the least fixed point of* $[\text{weak}]_P$, *then a query of P with A fails.*

Proof. Let $\boldsymbol{I}$ be the partial L-interpretation consisting of all TA where a query of P with A succeeds, and all FA where a query of P with A fails. It is easy to see that $[\text{tab}]_P(\boldsymbol{I}) \subseteq \boldsymbol{I}$, so $[\text{weak}]_P(\boldsymbol{I}) \subseteq \boldsymbol{I}$, and hence the least fixed point of $[\text{weak}]_P$ is a subset of $\boldsymbol{I}$. □

Proposition 12.3 (soundness). *Again, let A be an atomic statement of L. If a query of P with A succeeds, then TA is in the least fixed point of* $[\text{weak}]_P$. *If a query of P with A fails, then FA is in the least fixed point of* $[\text{weak}]_P$.

Proof. For a query of P with A to succeed (or fail) there must exist a closed P-tableau $\boldsymbol{T}_0$ for FA (or for TA). That tableau will, through the recursive program rules, require the existence of other closed P-tableaus, these may call on others, and so on. We can think of these as being arranged in a *tree of tableaus.* At the origin is the P-tableau $\boldsymbol{T}_0$. The children of a node containing $\boldsymbol{T}_n$ are the closed P-tableaus needed (via the recursive program rules) to ensure closure of $\boldsymbol{T}_n$. At the leaves are P-tableaus making no recursive calls. By the *depth* of such a tree we mean the length of a maximal branch. By the *depth* of a query that succeeds or fails we mean the minimal depth of a tree of tableaus establishing success or failure of that query.

Now the soundness argument is by induction on depth. Let $\boldsymbol{M}$ be the smallest fixed point of $[\text{weak}]_P$. For each n, let $\boldsymbol{D}_n$ be the set of all TB where a query of P with B succeeds with depth $\leqslant n$, together with all FB where a query of P with B fails with depth $\leqslant n$. We need that $\boldsymbol{D}_n \subseteq \boldsymbol{M}$ for all n.

It is easy to see that $\boldsymbol{D}_1 = \emptyset$ (only atomic queries are being considered, so at least one recursive call is required). It is equally easy to see that $\boldsymbol{D}_{k+1} = [\text{tab}]_P(\boldsymbol{D}_k)$. Then the induction argument is simple: $\boldsymbol{D}_1 \subseteq \boldsymbol{M}$. And if $\boldsymbol{D}_k \subseteq \boldsymbol{M}$, then

$$\boldsymbol{D}_{k+1} = [\text{tab}]_P(\boldsymbol{D}_k) = [\text{weak}]_P(\boldsymbol{D}_k) \subseteq [\text{weak}]_P(\boldsymbol{M}) = \boldsymbol{M}. \qquad \square$$

13. Directions for future work

The tableau-based logic programming language outlined above is attractive in theory. Whether or not it is practical to implement is another story. The need for a theorem prover for full first-order logic with equality is an obvious block. On the other hand, it may be possible to identify syntactically restricted subsystems of the tableau-based system that are efficient to implement and that still properly extend pure Horn clause programming. We have made a small beginning on this.

Further, the tableau-based system involves both proof and dual disproof notions. This forces an implementation to have both a unification algorithm and a dualized version of it. (This assumes that suitable variables have been introduced into the language whose values in queries are being sought.) If a tableau branch contains F$t = u$, and t and u unify, the branch closes. Dually, if a branch contains T$t = u$, the branch closes if t and u can be made un-unifiable (call it disunifying t and u). Often, this can be done in many ways. For instance, perhaps t and u can be made into distinct constant symbols or can be made to begin with distinct function symbols. Combinatorially, this multiplicity of ways is not good. On the other hand, different ways of disunifying t and u are independent of each other, which suggests that an implementation involving parallelism can be of considerable utility here, unlike with the unification part.

Finally, on the theoretical level, the connection between conventional finite failure and tableau-based failure remains open. Likewise, the relationship with the intuitionistic-logic-based generalization of [9] is not clear. It is possible that the well-known relationships between classical and intuitionistic logics will have some bearing here.

References

[1] K.R. Apt and M.H. Van Emden, Contributions to the theory of logic programming, *J. Assoc. Comput. Mach.* **29** (1982) 841-862.

[2] K.L. Clark, Negation as failure, in: H. Gallaire and J. Minker eds., *Logic and Data Bases* (Plenum, New York, 1978) 293-322.

[3] M.C. Fitting, A Kripke/Kleene semantics for logic programs, *J. Logic Programming* (1985) 295-312.

[4] J.H. Gallier, On the existence of optimal fixpoints, *Math. Systems Theory* **13** (1980) 209-217.

[5] S.C. Kleene, *Introduction to Metamathematics* (Van Nostrand, New York, 1952).
[6] S. Kripke, Outline of a theory of truth, *J. Philosophy* **72** (1975) 690–716.
[7] J. Lassez and M. Maher, Optimal fixedpoints of logic programs, *Theoret. Comput. Sci.* **39** (1985) 15–25.
[8] Z. Manna and A. Shamir, The theoretical aspect of the optimal fixed point, *SIAM J. Computing* **5** (1976) 414–426.
[9] L.T. McCarthy, Fixed point semantics and tableau proof procedures for a clausal intuitionistic logic, Tech. Rept. LRP-TR-18, Rutgers University.
[10] R.M. Smullyan, *First Order Logic* (Springer, Berlin, 1968).
[11] M.H. Van Emden and R. Kowalski, The semantics of predicate logic as a programming language, *J. Assoc. Comput. Mach.* **23** (1976) 733–742.
[12] B. Van Fraassen, Singular terms, truth-value gaps, and free logic, *J. Philosophy* **63** (1966) 481–485.