PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1984,
edited by ICOT.

A THEORY OF COMPLETE LOGIC PROGRAMS WITH EQUALITY

Joxan Jaffar*, Jean-Louis Lassez† and Michael J. Maher†

*Department of Computer Science
Monash University
Clayton, Victoria 3168
Australia.

†Department of Computer Science
University of Melbourne
Parkville, Victoria 3052
Australia.

ABSTRACT

Incorporating equality into the unification process has added great power to automated theorem provers. We see a similar trend in logic programming where a number of languages are proposed with specialized or extended unification algorithms. There is a need to give a logical basis to these languages. We present here a general framework for logic programming with definite clauses, equality theories and generalized unification. The classic results for definite clause logic programs are extended in a simple and natural manner. The extension of the soundness and completeness of the negation-as-failure rule for complete logic programs is conceptually more delicate and represents the main result of this paper.

1 INTRODUCTION

In this paper, we consider generalized unification (e.g. Siekmann and Szabo 1982), i.e. unification of terms in equality theories, in the framework of logic programming. The theoretical foundation of incorporating equality into the unification process of theorem-proving was given by (Plotkin 1972). The major result here was that for a set of clauses augmented with an equational theory, one can work on the clauses alone and yet have a complete inference system in a theorem-prover using a generalized unification algorithm, which respects the equational theory in question, and the usual resolution and paramodulation inference rules. As argued in both the above mentioned papers and (ICOT 1984), the study of generalized unification can have a tremendous practical significance. Our aims here are to present the counterpart of such results for logic programming, in particular for complete logic programs.

It is well-known that by restricting the logic to definite clauses, we can have an elegant semantics for the resulting programming language (Van Emden and Kowalski 1976, Apt and van Emden 1982, Lassez and Maher 1984); furthermore, indications are that appropriate logic programming systems can be practically efficient (e.g. the various PROLOGs). In this paper, we show that the main desirable results for logic programming continue to hold in a more general framework of equality formulas in logic programs and generalized unification in the inference system. Thus this paper provides theoretical foundations for works such as those of (Kornfeld 1983) on equality in logic programs, and is relevant to works on functional programming in logic programming such as (Kahn 1981, Subrahmanyam and You 1984). Furthermore, the work of (Hansson and Haridi 1981) and (van Emden and Lloyd 1984) on various soundness results falls within this general framework which can be used to address the issues of completeness and negation as failure for Prolog II (Jaffar et al 1984).

The main result however concerns complete logic programs. A promising approach toward handling the assertion of negative facts in logic programming is in using the concept of complete logic programs (Clark 1978, Apt and van Emden 1982, Jaffar et al 1983). A particular attraction in the present efforts is that results on complete logic programs are associated with implementations of *standard* logic programs. That is, we gain additional expressive power for no additional cost.

One interesting (and indeed crucial as far as the present results are concerned) aspect of completed logic programs is the equality axioms embedded. In (Clark 1978) and (Jaffar et al 1983) these axioms enforce what is essentially, but not only, the Herbrand interpretations. These axioms are intimately connected with the standard unification algorithm which corresponds to syntactic equality. Upon close inspection one sees that these axioms are used to state explicitly those properties which are already built into the unification

† Research partially supported by the Australian Computer Research Board.

algorithm.

Here we consider complete logic programs incorporating equality axioms of the form of definite clauses. We show that a logic programming system (for the corresponding standard program) using appropriate generalized unification is a sound and complete inference mechanism for the complete program. In particular, the negation-as-failure rule in this general framework remains sound and complete.

The paper is organized as follows. In the next section we generalize the theory of definite clause logic programs whose equality theory is based on syntactic identity to cater for programs with a class of equality theories. In section 3 we develop a theory of complete logic programs and unification-complete equality theories. This contrasts with present results which are restrictive in that complete programs are defined incorporating equality axioms which enforce interpretations which are essentially Herbrand ones.

2 LOGIC PROGRAMMING WITH EQUALITY AND GENERALIZED UNIFICATION

We use the symbols V, Σ and Π to denote the sets of variables, function symbols and non-logical predicate symbols respectively. Thus the latter set does not contain the symbol $=$. $\tau(\Sigma)$ and $\tau(\Sigma \cup V)$ denote respectively the ground terms and the terms possibly containing variables. Throughout this paper we follow (Shoenfield 1967) for our mathematical logic terminology.

A *definite clause logic program* is defined in the usual manner, i.e. a finite set of definite clauses (van Emden and Kowalski 1976). Note that there are no $=$ symbols in definite clauses. In this paper, we also consider *definite clause equality theories*. As usual, *equations* are of the form $s = t$ where s and t are terms over $\tau(\Sigma \cup V)$. A *definite equality clause* is of the form

$$e \leftarrow e_1, e_2, \dots, e_m$$

where $m \geq 0$ and all the atoms therein are equations. As usual, variables in definite equality clauses are implicitly universally quantified. We define a definite clause equality theory to be a possibly infinite set of definite equality clauses. See (Selman 1972) for some properties of definite clause equality theories. Finally, we define a *logic program* to be a pair (P, E) where P is a definite clause logic program and E a definite clause equality theory.

Generalized unification is defined here with respect to a definite clause equality theory E. Our definitions below are compatible with those in the literature (e.g. Huet and Oppen 1980, Siekmann and Szabo 1982). (However, such works usually consider only equational theories, i.e. universal closures of equations.) A *substitution* is defined to be a mapping from the set of variables V into the set of terms $\tau(\Sigma \cup V)$. In what follows we sometimes (a) use obvious generalizations of substitutions to maps from terms into terms, and (b) speak of substitutions as equations, e.g. the substitution $\{x/t, y/u\}$ can be regarded as the set of equations $\{x = t, y = u\}$. An *E-unifier* for two terms s and t is a substitution θ such that $E \models s\theta = t\theta$.

We consider next the semantics of logic programs. As mentioned earlier, definite clause programs have an elegant formal semantics. The major reason for this is the existence of a canonical class of interpretations, namely the Herbrand ones, for the clauses. This follows from

$$P \models p \text{ iff } P \models_{\tau(\Sigma)} p$$

where P is a definite clause logic program, p an atom and where $\models_D$ denotes logical implication in the context of a fixed domain and functional assignment, in this case, D is the Herbrand universe and functional assignment. This means that logical inference and refutations can be obtained within the purely syntactic framework of Herbrand interpretations. Furthermore, the existence of a least model for definite clauses provides a rigorous and simple declarative semantics for the corresponding programs (van Emden and Kowalski 1976).

Consider now our logic programs (P, E) which contain equality theories. Here also we have a canonical class of interpretations. Let $\tau(\Sigma)/R$ denote the quotient of $\tau(\Sigma)$ by the congruence relation R. Thus the functional assignment is given by $f([t_1], \dots, [t_n]) = [f(t_1, \dots, t_n)]$ for all n-ary f in S. It is well-known (see e.g. Loveland 1978) that

$$(P, E) \models p \text{ iff } (P, E) \models_{\tau(\Sigma)/R} p \text{ for all R}$$

where p is an atom, possibly an equation. What we require however, is a fixed domain and functional assignment, that is, a canonical congruence relation R for a given program (P, E). Clearly the only relations R we need consider are given by the models of E. However, there is in general more than one model of E. The problem then, is to select a model which is representative of this collection. We prove in the lemma below that a least such R exists. This then gives us Theorem 1, i.e. R provides the desired canonical class of interpretations. This is in perfect analogy with canonicality of least models of definite clause programs.

Lemma 1: There exists a finest Σ-congruence over $\tau(\Sigma)$ generated by each definite clause equality theory E.

Proof. Consider models of E over $\tau(\Sigma)$, and for our purposes here, a model is a set of pairs. Suppose now that I is the intersection of a set of models of E. If I is not a model itself, then some ground instance of a clause in E, say $e \leftarrow e_1, \dots, e_n$, is falsified by I. This means that e is not in I and $e_1, \dots, e_n$ are in I, contradicting the fact that e is in the models of the set in question. The finest Σ-congruence then is given by the intersection of all models of E, with the obvious functional assignment such that $f([t_1], \dots, [t_n]) = [f(t_1, \dots, t_n)]$ for all n-ary f in Σ. ∎

Although we consider only definite clause equality theories E in this section, all the results below continue to hold for *any* open equality theory which has a finest Σ-congruence.

Let R0 be the finest Σ-congruence generated by E.

Theorem 1. $(P, E) \models p$ iff $(P, E) \models_{\tau(\Sigma)/R0} p$.

Proof. From the above remarks, it suffices to prove that $(P, E) \models_{\tau(\Sigma)/R0} p$ iff $(P, E) \models_{\tau(\Sigma)/R} p$ for all R. The if part is trivial; consider the other part. For some R, let I be any model over $\tau(\Sigma)/R$ for (P, E) such that p is false. Construct the following model J over $\tau(\Sigma)/R0$ by defining that $q([\tilde{t}]_{R0})$ is true in J iff $q([\tilde{t}]_R)$ true in I for all n-ary predicates symbol q. This is well-defined because R0 is finer than R. That J is indeed a model, in which p is false, is now easy to see. ∎

We now have the justification of working in a fixed domain. That is to say, we have that there is a canonical domain corresponding to a logic program, namely $\tau(\Sigma)/R0$ where R0 is the finest Σ-congruence over $\tau(\Sigma)$ generated by E. We henceforth may write $\tau(\Sigma)/E$ for $\tau(\Sigma)/R0$ for *any* open equality theory E which has a finest Σ-congruence R0.

Let $\tilde{t}$ denote a sequence of terms $t_1, t_2, \dots, t_n$, $n \geq 0$. We now give definitions with respect to a given logic program (P, E). The *E-base* is $\cup\{p(\tilde{d}): \tilde{d} \in (\tau(\Sigma)/E)^n\}$ over all n-ary predicate symbols p. An *E-interpretation* I is a subset of the E-base. We write $[s]$ to denote the element in $\tau(\Sigma)/E$ assigned to the ground term s. Similarly, $[\tilde{t}]$ is a element of $(\tau(\Sigma)/E)^n$ and $[p(\tilde{t})]$ is an element of the E-base. Where S is a set of ground terms, $[S]$ denotes $\{[s]: s \in S\}$.

We now define the appropriate generalizations of derivation sequences success and finite failure sets for our logic programs. We point out here that while these sets are defined in an operational manner, we do not address in this paper the issue of corresponding computational methods implementing them. In what follows, we write $\tilde{t} = \tilde{u}$ to mean $t_1 = u_1 \wedge u_2 = t_2 \wedge \dots \wedge t_n = u_n$. Thus we say $\tilde{t}$ E-unifies with $\tilde{u}$ to mean that $E \models (t_1 = u_1 \wedge \dots \wedge t_n = u_n)$. We observe here that an immediate consequence of Theorem 1 is that $\tilde{t}$ E-unifies with $\tilde{s}$ iff $[\tilde{s}] = [\tilde{t}]$.

For notational convenience, we assume that the variables in V are not subscripted. A *(P, E)-derivation sequence* is a (finite or infinite) sequence of triples $<G_i, \tilde{C}_i, \theta_i>$, $i = 0, 1, \dots$ such that (a) G_i is of the form $B_1, \dots, B_m$ where $m \geq 0$ and each B_j is an atom, for all $1 \leq j \leq m$, (b) $\tilde{C}_i$ is a list of m clauses

$$A^{(1)} \leftarrow D_1^{(1)}, \cdots, D_{n_1}^{(1)}$$

$$A^{(2)} \leftarrow D_1^{(2)}, \cdots, D_{n_2}^{(2)}$$

$$\cdots$$

$$A^{(m)} \leftarrow D_1^{(m)}, \cdots, D_{n_m}^{(m)}$$

where each clause above is a clause from P with variables renamed in that they are now subscripted with numbers never before used in subscripting in any G_j where $j < i$, (c) θ_i is an E-unifier of $(B_1, \dots, B_m)$ and $(A^{(1)}, \dots, A^{(m)})$, and (d) G_{i+1} is

$$(D_1^{(1)}, \cdots, D_{n_1}^{(1)}, \cdots D_1^{(m)}, \cdots, D_{n_m}^{(m)})\theta_i$$

A derivation sequence is ***finitely failed*** with length i if θ_i cannot be formed, that is, $(B_1, \dots, B_m)$ and $(A^{(1)}, \dots, A^{(m)})$ do not E-unify. A derivation sequence is ***successful*** if some G_i is empty (i.e.

$m = 0$). Note that a derivation sequence is either successful, finitely failed or infinite.

The following defines the success, finite failure and general failure sets, denoted SS, FF and GF respectively, for a given logic program (P, E).

$SS(P, E) = \{p(\tilde{s}): \tilde{s}$ is ground and
there exists a successful (P, E)-derivation sequence of $p(\tilde{s})\}$

$FF(P, E) = \{p(\tilde{s}): \tilde{s}$ is ground and
there exists a number n such that
all (P, E)-derivation sequences of $p(\tilde{s})$ are finitely failed with length $\leq n\}$

$GF(P, E) = \{p(\tilde{s}): \tilde{s}$ is ground and
all (P, E)-derivation sequences of $p(\tilde{s})$ are finitely failed$\}$

Thus these definitions relate closely to resolution-like implementations of logic programming systems. It is necessary for us to consider the set GF because in general there is a ground atom which may not have an infinite derivation sequence and yet there is no number n such that all derivation sequences of this atom are finitely failed with length $\leq$ n. This possibility can arise because E can be such that there is an infinite set of maximally general E-unifiers for some pair of terms s and t. However, if E is such that for all pairs of terms s and t, there is a finite set of maximally general unifiers which subsume all the E-unifiers of s and t, then GF is identical to FF.

In the standard framework, the success and finite failure sets have also been defined inductively. We give the appropriate generalizations here.

$SS_0(P, E) = \{\}$

$SS_{i+1}(P, E) = \{p(\tilde{t}): \tilde{t}$ is ground and there is
a ground instance of a clause in P
$p(\tilde{u}) \leftarrow B_1, \ldots, B_m$
such that $\tilde{t}$ E-unifies with $\tilde{u}$, and
$B_k \in SS_i(P, E)$ for all $1 \leq k \leq m\}$

$SS(P, E) = \bigcup_{i=0}^{\infty} SS_i(P, E)$

$FF_0(P, E) = \{\}$

$FF_{i+1}(P, E) = \{p(\tilde{t}): \tilde{t}$ is ground, and
for each ground instance of a clause in P
$p(\tilde{u}) \leftarrow B_1, \ldots, B_m$
either $\tilde{t}$ does not E-unify with $\tilde{u}$, or
$B_k \in FF_i(P, E)$ for some $1 \leq k \leq m\}$

$FF(P, E) = \bigcup_{i=0}^{\infty} FF_i(P, E)$

The proof of the following proposition is long but follows lines similar to the standard case and is therefore omitted.

Proposition 1.
(a) The two definitions of SS(P, E) define the same set.
(b) The two definitions of FF(P, E) define the same set.

One therefore might suspect that a corresponding (transfinite) inductive definition can be made for the set GF(P, E). If α and β denote not necessarily finite ordinals, then one could try:

$GF_0(P, E) = \{\}$

$GF_\alpha(P, E) =$
if ($\alpha \neq 0$ and α is not a limit ordinal) *then*
$\{p(\tilde{t}): \tilde{t}$ is ground, and
for each ground instance of a clause in P
$p(\tilde{u}) \leftarrow B_1, \ldots, B_m$
either $\tilde{t}$ does not E-unify with $\tilde{u}$, or
$B_k \in GF_{\alpha-1}(P, E)$ for some $1 \leq k \leq m\}$
else
$\bigcup_{\beta<\alpha} GF_\beta(P, E)$

GF(P, E) is such that $A \in GF(P, E)$ iff $A \in GF_\alpha(P, E)$ for some ordinal α.

Unfortunately, one can show that this definition is *not* equivalent to the above. Thus while we may use either one of the definitions for SS and FF, we have only one definition of GF in this paper. The problem of finding an inductive definition for GF(P, E) remains.

As in (van Emden and Kowalski 1976) we make use of a function T in which terms most of the fundamental results can be framed. In the definition below, E denotes *any* open equality theory which has a finest congruence. $T_{(P, E)}$ is a function from and into E-interpretations.

$T_{(P,E)}(I) = \{p(\tilde{d})$: there is
a ground instance of a clause in P
$p(\tilde{s}) \leftarrow B_1, \dots, B_m$
such that $[\tilde{s}] = \tilde{d}$ and
$[B_k] \in I$ for $1 \leq k \leq m\}$

We are now in a position to extend the classic results of standard logic programming theory. Since we have a canonical domain (cf. Theorem 1) for a given program (P, E), the proofs of the lemmas leading to Theorem 2 and the theorem itself are simple extensions of their counterparts in the standard theory. Theorems 3 and 4 follow from the various lemmas and Proposition 1. The main new concept to be found in the lemmas below is generalized unification. Below we write I for some E-interpretation of (P, E), and for brevity, we sometimes write T, SS and FF for $T_{(P, E)}$, SS(P, E) and FF(P, E) respectively.

Lemma 2. $T_{(P, E)}$ is continuous.

Lemma 3. I models (P, E) iff $T_{(P, E)}(I) \subseteq I$.

Lemma 4. For all $i \geq 0$,
(a) $p(\tilde{t}) \in SS_i$ iff $[p(\tilde{t})] \in [SS_i]$.
(b) $p(\tilde{t}) \in FF_i$ iff $[p(\tilde{t})] \in [FF_i]$.

We write $T\uparrow\omega$ for $\bigcup_{i=0}^{\infty} T^i(\{\})$, and $T\downarrow\omega$ for $\bigcap_{i=0}^{\infty} T^i$(E-base). Using appropriate fixpoint theorems (see e.g. Lassez et al 1982), we have, from lemma 2, that $T\uparrow\omega$ is the least fixpoint of T and, from lemma 3, that $T\uparrow\omega$ is the least model of (P, E), similarly to the standard case (van Emden and Kowalski 1976). Thus

Theorem 2. The least model of (P, E) is equal to the least fixpoint of $T_{(P, E)}$.

One more characterization of $T\uparrow\omega$ is given by

Lemma 5. $T_{(P, E)}\uparrow\omega = [SS(P, E)]$.

Proof. Let $T\uparrow i$ denote $T^i(\{\})$. We show $[SS_i] = T\uparrow i$ for all $i \geq 0$ by induction; the lemma then follows. The base case $i = 0$ is trivially proved. Now

$[SS_{i+1}] = \{[p(\tilde{t})]: p(\tilde{t}) \in SS_{i+1}\}$,
$= \{[p(\tilde{t})]: \tilde{t}$ is ground and
there is a ground instance of a clause in P
$p(\tilde{u}) \leftarrow B_1, \dots, B_m$
such that $\tilde{t}$ E-unifies with $\tilde{u}$, and
$B_k \in SS_i$ for all $1 \leq k \leq m\}$
by the definition of SS_{i+1}
$= \{[p(\tilde{t})]: \tilde{t}$ is ground and
there is a ground instance of a clause in P
$p(\tilde{u}) \leftarrow B_1, \dots, B_m$
such that $[\tilde{t}] = [\tilde{u}]$, and
$[B_k] \in [SS_i]$ for all $1 \leq k \leq m\}$,
by Theorem 1 and lemma 4(a)
$= T([SS_i])$,
by the definition of T
$= T(T\uparrow i)$,
by the induction hypothesis
$= T\uparrow i+1$. ∎

The following theorem establishes the soundness and completeness of a proof strategy based on (P, E)-derivation sequences.

Theorem 3. If $p(\tilde{t})$ is a ground atom
$(P, E) \models p(\tilde{t})$ iff $p(\tilde{t}) \in SS(P, E)$

Finally we have a dual result for finite failure.

Theorem 4. If $p(\tilde{t})$ is a ground atom
$p(\tilde{t}) \in FF(P, E)$ iff $[p(\tilde{t})] \in \overline{T_{(P, E)}\downarrow\omega}$.

3 COMPLETE LOGIC PROGRAMS WITH EQUALITY THEORIES

As mentioned before, generalized unification is usually defined over an equational theory E, i.e. a set of open equations in some fixed alphabet Σ. Two terms are then said to be E-unifiable iff there is a ground substitution over $\tau(\Sigma)$ of the terms such that the ground instances are both in the same class of the finest Σ-congruence over $\tau(\Sigma)$ generated by E. This does not however mean that if two terms are equal in another Σ-algebra modelling E then they are E-unifiable.

In this section we want to establish a relationship between falsity and failure of atoms. We thus require in this section that an equality theory dictates that equality holds only if E-unification is possible. Formally, we say that an open equality theory E is *unification complete* over $\tau(\Sigma)$ if for every equality formula e of the form

$$\exists \tilde{y}(\tilde{s} = \tilde{t}),$$

where $\tilde{y}$ are the variables appearing in the terms $\tilde{s}$ and $\tilde{t}$, either $E \models \neg e$, or else there exists a nonempty and possibly infinite set $\{\theta_i\}$ of E-unifiers of $\tilde{s}$ and $\tilde{t}$ such that

$$\forall \tilde{y}((\tilde{s} = \tilde{t}) \rightarrow \vee\{\theta_i\}).$$

Note that the above expression means that in any model for E, the following holds: If a valuation of the variables in terms s and t is such that $s = t$ in the model, then at least one of the E-unifiers θ_i (looked upon as a set of equations) is also true in the model and valuation.

An *augmented definite clause logic program* consists of a conjunction of *predicate definitions*, exactly one for each predicate symbol in Π. These definitions take one of two forms:

$$p_i(\tilde{x}) \longleftrightarrow D_i, \tag{1}$$

or

$$\neg p_i(\tilde{x}) \tag{2}$$

where the $\tilde{x}$ are a list of n_i distinct variables, the p_i are n_i-ary predicate symbols, and the D_i are the *definition bodies* of p_i. These bodies are each a disjunction of formulas of the form

$$\exists \tilde{y}(\tilde{x} = \tilde{t} \wedge B_1 \wedge B_2 \wedge \dots \wedge B_m)$$

where the B_j are atoms and $\tilde{y}$ are the variables distinct from $\tilde{x}$ appearing in the formula. Note that these augmented programs are the same as the complete programs of (Clark 1978) except that we do not include his equality axioms. Finally we can define our *complete logic programs*: these are of the form (P*, E*) where P* is an augmented definite clause program and E* a unification complete equality theory.

It is well-known (see e.g. Clark 1978) how one obtains from a definite clause logic program a corresponding augmented version. The converse is also easy to define, that is to say, we can obtain from a given P* an unaugmented program P. This is done as follows: For each predicate definition of type (2) in P*, obtain k definite clauses where k is the number of disjunctions in the definition. Then if

$$\exists \tilde{y}(\tilde{x} = \tilde{t} \wedge B_1 \wedge B_2 \wedge \dots \wedge B_m) \tag{3}$$

is one such disjunct, obtain the corresponding definite clause

$$p_i(\tilde{t}) \leftarrow B_1, \dots, B_m. \tag{4}$$

Note that we do not construct any definite clauses from predicate definitions of type (2) in P*.

For unification complete equality theories E* however, we have the following as the un-complete counterpart: $E = \{e\colon e \text{ is a ground equation and } E* \models e\}$. In the other direction, one can deal separately with each definite clause equality theory E. For example, (Clark's 1978) axioms form a unification complete extension of the trivial equality theory consisting only of the usual equality axioms. In general however, there is no unique E* corresponding to an E. In what follows, we are only concerned with the E corresponding to some E*.

Since E* is unification complete, we say that I is an E*-interpretation to mean, as in section 2, that I has the domain given by $\tau(\Sigma)/E*$, this being the unique Σ-congruence over $\tau(\Sigma)$ generated by E*. Thus I may be regarded as an interpretation of arbitrary formulas in the obvious way, i.e. I defines the domain and functional assignment by virtue of it being an E*-interpretation, and I defines truth values via its elements. For brevity, we now write, when convenient, T, SS and GF for $T_{(P,\,E)}$, SS(P, E) and GF(P, E) respectively, where (P, E) is the corresponding logic program to the complete logic program (P*, E*) in question. We write $p(\tilde{s})$ to denote some ground atom. The following lemma generalizes a result of (Apt and van Emden 1982).

Lemma 6. If I is an E-interpretation,
I is a fixpoint of $T_{(P,\,E)}$ iff I is a model for (P*, E).

Proof. Let p be any non-logical n-ary predicate symbol in P* and recall that there is only one definition of p there. If it is of the form (1), i.e. $p(\tilde{x}) \longleftrightarrow \bigvee_{i=1}^{k} C_i$ where each conjunction C_i, $1 \le i \le k$, is of the form (3), then this definition is satisfied by I iff
for all $\tilde{d}$ in $(\tau(\Sigma)/E)^n$,
$p(\tilde{d}) \in I \;\longleftrightarrow\;$ for some C_i and ground substitution θ, $\tilde{d} = [\tilde{t}\theta]$ and $[B_j\theta] \in I$ for all $1 \le j \le m$.
Since for each C_i there is a definite clause about p in P and vice versa, this is the same as
$p(\tilde{d}) \in I \;\longleftrightarrow\; p(\tilde{d}) \in T(I)$ for all $\tilde{d}$.

If however the definition of p is of the form (2), $\neg p(\tilde{x})$ is satisfied by I iff $p(\tilde{d}) \notin I$ for all $\tilde{d}$. By the definition of $T_{(P,\,E)}$, we have for each such p that for all E-interpretations J and all $\tilde{d}$, $p(\tilde{d}) \notin T(J)$. Hence (P*, E) is satisfied by I iff $T_{(P,\,E)}(I) =$

I. ∎

We are ready for two main theorems. The first proves the soundness and completeness of successful (P, E)-derivations for positive atoms valid in (P*, E*). We phrase our theorem thus:

Theorem 5. $(P*, E*) \models p(\bar{s})$ iff $p(\bar{s}) \in SS$.

Proof. (→).

By the lemmas in section 2, if $p(\bar{s}) \notin SS$, then $p(\bar{s}) \notin I$ where I is the least E-model of P. Again by these lemmas, $I = T\uparrow\omega$ is a fixpoint of T. Since I is also an E*-interpretation, $T_{(P, E*)} = T$, and thus I is a fixpoint of $T_{(P, E*)}$. By lemma 6, I is a model for (P*, E*).

(←).

It suffices to show that $P* \models P$. This is easily done along the following chain of reasoning: Any definition of the form $p(\bar{x}) \leftrightarrow D$ in P* contains the subformula

$$p(\bar{x}) \leftarrow D$$

where D is of the form $C_1 \vee ... \vee C_k$ for some $k > 0$. This in turn is equivalent to the conjunction of

$$p(\bar{x}) \leftarrow C_j$$

for $1 \leq j \leq k$. That is to say we have a conjunction of formulas of the form

$$p(\bar{x}) \leftarrow \exists\bar{y}(\bar{x} = \bar{t} \wedge B_1 \wedge ... \wedge B_m).$$

Each such formula is equivalent to

$$p(\bar{x}) \leftarrow (\bar{x} = \bar{t} \wedge B_1 \wedge ... \wedge B_m).$$

by a suitable manipulation of quantifiers. Finally, this clearly implies the definite clause which appears in P

$$p(\bar{t}) \leftarrow B_1, ..., B_m.$$

Since every definite clause in P is implied by some definition such as the $p(\bar{x}) \leftrightarrow D$ above, we are done. ∎

We now prove the soundness and completeness of generally failed (P, E)-derivations for negative atoms valid in (P*, E*). Thus we justify a form of the negation-as-failure rule (Clark 1978). Our theorem reads

Theorem 6. $(P*, E*) \models \neg p(\bar{s})$ iff $p(\bar{s}) \in GF$.

Proof. (←).

We prove that if for some model M of (P*, E*), $\exists\bar{x}(A_1 \wedge A_2 \wedge ... \wedge A_n)$ is true, then the goal $A_1, A_2, ..., A_n$ has an infinite (P, E)-derivation sequence or a successful one. It suffices to show that either $A_1, ..., A_n$ is empty or we can have a derivation step starting from this goal and obtaining a goal $B_1, B_2, ..., B_m$ such that $\exists\bar{y}(B_1 \wedge B_2 \wedge ... \wedge B_m)$ is also true in M. Repeated application of this construction proves the existence of a (P, E)-derivation sequence which is either infinite or successful.

Suppose that for each $1 \leq i \leq n$, A_i is of the form $p^{(i)}(...\bar{x}^{(i)}...)$ where $...\bar{x}^{(i)}...$ stands for a list of terms over $\tau(\Sigma \cup V)$ whose variables appear in the list $\bar{x}^{(i)}$. Consider the definition in P* of each of these (not necessarily distinct) predicate symbols $p^{(i)}$:

$$p^{(i)}(...\bar{x}^{(i)}...) \longleftrightarrow D$$

where D is a disjunction of formulas of the form

$$\exists\bar{y}^{(i)}(...\bar{x}^{(i)}... = \bar{t}^{(i)} \wedge B_1^{(i)} \wedge ... \wedge B_m^{(i)}) \qquad \#(i)$$

Let V_0 denote a valuation of $A_1, ..., A_n$, i.e. an assignment of an element in the domain of M to each variable in $\bar{x}$ such that this conjunction is true in M. Therefore, for each $1 \leq i \leq n$, $p^{(i)}(...\bar{x}^{(i)}...)$ is true in M under this valuation V_0. It thus follows that one of the formulas #(i) is true under this valuation. Hence the conjunction of these n formulas

$$\#(1) \wedge \#(2) \wedge ... \wedge \#(n)$$

is true in M under the valuation V_0. Thus so is

$$\exists\bar{y}^{(1)}...\bar{y}^{(n)}(\bigwedge_{i=1}^{n} (...\bar{x}^{(i)}... = \bar{t}^{(i)} \wedge B_1^{(i)} \wedge ... \wedge B_{m(i)}^{(i)})).$$

Hence the following is true in M:

$$\exists\bar{x}\bar{y}^{(1)}...\bar{y}^{(n)}(\bigwedge_{i=1}^{n} (...\bar{x}^{(i)}... = \bar{t}^{(i)} \wedge B_1^{(i)} \wedge ... \wedge B_{m(i)}^{(i)})).$$

where $\bar{x}$ is the list of variables in $\bar{x}^{(1)}, ..., \bar{x}^{(n)}$. Our proof is now complete by three observations: (a) Since E* is unification complete, there exists at least one E-unifier θ for the equations

$$\bigwedge_{i=1}^{n} (\ldots\tilde{x}^{(i)}\ldots = \tilde{t}^{(i)})$$

such that θ is true under any valuation for which V_0 is a restriction. (b) Thus for each $1 \leq i \leq n$, $(B_1^{(i)} \wedge B_2^{(i)} \wedge \ldots \wedge B_m^{(i)})\theta$ is true in M for some valuation of the variables therein. (c) There exists in P definite clauses of the form

$$p^{(i)}(\tilde{t}^{(i)}) \leftarrow B_1^{(i)}, B_2^{(i)}, \ldots, B_{m(i)}^{(i)}$$

for all $1 \leq i \leq n$. Putting (a), (b) and (c) together, we may conclude that from $A_1, \ldots, A_n$ and these definite clauses above, we can (P, E)-derive

$$(B_1^{(1)}, \ldots, B_{m(1)}^{(1)}, \ldots, B_1^{(n)}, \ldots, B_{m(n)}^{(n)})\theta.$$

$(\rightarrow)$.

Assuming that $p(\tilde{s}) \notin GF$, we now construct a model for (P*, E*) in which $p(\tilde{s})$ is true. We may as well assume that $p(\tilde{s}) \notin SS$. By the results above, $p(\tilde{s})$ is the first goal in an infinite derivation sequence $\langle G_i, \tilde{C}_i, \theta_i \rangle$, $i = 0, 1, \ldots$. Recall that by our variable renaming convention, there are no common variables in $\tilde{C}_i$ and $\tilde{C}_j$ where $i \neq j$. Let E_i denote a finite set of ground equations over a larger alphabet $\Sigma+$ in that E_i is obtained from θ_i (looked upon as a set of equations) by replacing each distinct occurrence of a variable x_j with a distinct new constant symbol c_j. In what follows we make use of the fundamental property of the θ_i:

$$E \models \exists\tilde{x}(\theta_1 \wedge \theta_2 \wedge \ldots \wedge \theta_n) \quad (5)$$

for any finite n.

We now complete the proof in two main steps. Firstly, we show that $E{*}{+} = E{*} \cup \{E_i\}$ is consistent. Thus since E*+ is open, we may have (E*+)-interpretations I. Secondly we build a fixpoint I of $T_{(P,\ E*+)}$. Using lemma 7, we are done.

To show that E*+ is consistent, it suffices to show, by the Compactness Theorem, that $E{*}{+}_n = E{*} \cup \{E_1, \ldots, E_n\}$ is consistent for all finite n. Let A be any closed formula over $\tau(\Sigma)$. Consider the following chain of reasoning:

$$E{*}{+}_n \models A$$

implies

$$E{*} \models (E_1 \wedge \ldots \wedge E_n) \rightarrow A$$

by the fact that $E_1, \ldots, E_n$ have no variables and the Deduction theorem. Since E* contains only symbols in Σ, by the Theorem on Constants, we get

$$E{*} \models \forall\tilde{x}((\theta_1 \wedge \ldots \wedge \theta_n) \rightarrow A).$$

Since E* is an extension of E, from (5) we have

$$E{*} \models \exists\tilde{x}(\theta_1 \wedge \theta_2 \wedge \ldots \wedge \theta_n)$$

It easily follows that

$$E{*} \models A.$$

Thus $E{*}{+}_n$ is a conservative extension of E* and thus is consistent.

We can now complete the proof by constructing a fixpoint of $T_{(P,\ E*+)}$. Recall that all goals in the derivation sequence contain, if any, only subscripted variables x_j. Above we have defined, for each such variable x_j, a new constant c_j, i.e. a constant not in Σ. Now let [t] denote the congruence class over $\tau(\Sigma+)/(E{*}{+})$ containing the ground term obtained from t by replacing each occurrence of a subscripted variable x_j by the corresponding constant c_j. The important point here is that for any ground terms s and t, $[s] = [t]$ iff $E{*}{+} \models s = t$. Thus our notation [s] is consistent with our previous usage for the congruence class of s under some equality theory, in this case E*+.

We can now define

$I_0 = \{A\colon A \in [G_i]$ for some G_i in the derivation sequence$\}$

Next we show that $I_0 \subseteq T(I_0)$. Any atom in I_0 must be in $[G_i]$ for some i. Suppose G_i is of the form $B_1, B_2, \ldots, B_m$ and the associated input clauses $\tilde{C}_i$ are of the form

$$A^{(1)} \leftarrow D_1^{(1)}, \cdots, D_{n_1}^{(1)}$$

$$A^{(2)} \leftarrow D_1^{(2)}, \cdots, D_{n_2}^{(2)}$$

$$\cdots$$

$$A^{(m)} \leftarrow D_1^{(m)}, \cdots, D_{n_m}^{(m)}$$

Recall that we rename variables so that they are subscripted. Since the derivation sequence is infinite, G_{i+1} exists and must be of the form

$$(D_1^{(1)}, \cdots, D_{n_1}^{(1)}, \cdots D_1^{(m)}, \cdots, D_{n_m}^{(m)})\theta_i$$

where θ_i is an E-unifier of G_i and $A^{(1)}, \ldots, A^{(m)}$. Since $[G_{i+1}] \subseteq I_0$, we have $[A^{(1)}\theta_i, \ldots, A^{(m)}\theta_i] \subseteq T(I_0)$.

It remains to prove that $[A^{(j)}\theta_i] = [B_j]$ for all $1 \leq j \leq m$. Suppose now that θ_i is of the form

$$\{x_1/t_1(\bar{x}), x_2/t_2(\bar{x}), \ldots, x_k/t_k(\bar{x})\}$$

where $\bar{x}$ is the list of all subscripted variables appearing here. By construction, E_i is of the form

$$c_1 = t_1(\bar{c}), \ldots, c_k = t_k(\bar{c})$$

where $\bar{c}$ is the list of new constants corresponding to the $\bar{x}$. Since E*+ contains E_i, it follows that for all $1 \leq j \leq m$,

$$[B_j] = [B_j\theta_i]$$

Since also θ_i E-unifies $A^{(j)}$ and B_j, we obtain $[A^{(j)}\theta_i] = [B_j]$ for all $1 \leq j \leq m$, Thus $[B_1, \ldots, B_m] \subseteq T(I_0)$ and hence $I_0 \subseteq T(I_0)$.

Finally, we can use the Knaster-Tarski theorem about fixpoints for monotonic functions to show that there exists an I which contains I_0 such that $T(I) = I$. ∎

REFERENCES

Apt, K.R. and van Emden, M.H., Contributions to the Theory of Logic Programming, JACM 29(3), July 1983, 841 - 862.

Clark, K.L. Negation as Failure, in Logic and Databases, H. Gallaire and J. Minker (Eds), Plenum Press, New York, 1978, 293 - 322.

van Emden, M.H. and Kowalski, R. Semantics of Predicate Logic as a Programming Language, JACM 23(4), October 1976, 733 - 742.

van Emden, M.H. and Lloyd, J.W. A Logical Reconstruction of Prolog II, Proc. 2nd. Int. Logic Programming Conf., Uppsala, July 1984, 35 - 40; also Journal of Logic Programming, to appear.

Hansson, A. and Haridi, A.S. Programming in Natural Deduction Framework in Symposium on Functional Languages and Computer Architecture, Gottenberg, 1981.

Huet, G. and Oppen, D.C. Equations and Rewrite Rules: A Survey, in Formal Languages: Perspectives and Open Problems, R.V. Book (Ed), Academic Press, 1980.

ICOT, Several Aspects of Unification, Technical Report 0029, February 1984.

Jaffar, J., Lassez, J-L. and Lloyd, J.W. Completeness of the Negation-as-Failure Rule, Proc. 8th. IJCAI, Karlsruhe, August 1983, 500 - 506.

Jaffar, J., Lassez, J-L. and Maher, M.J. A Logical Foundation for Prolog II, Technical Report, Dept. of Computer Science, Monash University, 1984.

Kahn, K.M. Uniform: A Language Based Upon Unification Which Unifies Much of LISP, Prolog and Act 1, Proc. 7th. IJCAI, 1981, Vancouver, Canada.

Kornfeld, W.A. Equality for Prolog, Proc. 8th. IJCAI, Karlsruhe, August 1983, 514 - 519.

Lassez, J-L. and Maher, M.J. Closures and Fairness in the Semantics of Programming Logic, Theoretical Computer Science 29 (1984), 167 - 184.

Lassez, J-L., Nguyen, V.L. and Sonenberg, E.A. Fixedpoint Theorems and Semantics: A Folk Tale, Information Processing Letters, 1982, 112 - 116.

Loveland, D.W. Automated Theorem Proving: A Logical Basis, North-Holland, 1978.

Plotkin, G.D. Building in Equational Theories, in Machine Intelligence 7, B. Meltzer and D. Michie (Eds), Halsted Press, 1972, 73 - 90.

Selman, A. Completeness of Calculi for Axiomatically Defined Classes of Algebras, Algebra Universalis 2 (1) 1972, 20 - 32.

Shoenfield, J.R. Mathematical Logic, Addison-Wesley, 1967.

Siekmann, J. and Szabo, P. Universal Unification and a Classification of Equational Theories, Proc.

6th. Conf. on Automated Deduction, New York, June 1982, Lecture Notes in Computer Science 138, Springer-Verlag, 369 - 389.

Subrahmanyam, P.A. and You, J-H. Conceptual Basis and Evaluation Strategies for Integrating Functional and Logic Programming, International Symposium on Logic Programming, Atlantic City, 1984, 144 - 153.