COMPLETENESS OF THE NEGATION AS FAILURE RULE

Joxan Jaffar*, Jean-Louis Lassez' and John Lloyd

* Dept. of Computer Science, Monash University, Clayton, Victoria.
Dept. of Computer Science, University of Melbourne, Parkville, Victoria.

ABSTRACT

Let P be a Horn clause logic program and comp(p) be its completion in the sense of Clark. Clark gave a justification for the negation as failure rule by showing that if a ground atom A is in the finite failure set of P, then ~A is a logical consequence of comp(P), that is, the negation as failure rule is sound. We prove here that the converse also holds, that is, the negation as failure rule is complete.

I INTRODUCTION

If P is a Horn clause logic program, then we can use P to deduce "positive" information. In other words, if A is a ground atom, then the interpreter, by using SLD-resolution, can attempt to prove that A is indeed a logical consequence of P. However, we cannot deduce "negative" information using SLD-resolution. To be precise, we cannot prove that ~A is a logical consequence of P. The reason is that P{U) is satisfiable, having the Herbrand base as a model.

To remedy this defect, logic programming interpreters are usually augmented by the negation as failure rule. This rule states that if A is in the finite failure set of P, then -A holds. Thus we interpret the failure of the attempt to prove A as a "proof" that ~A holds. Rules very similar to the negation as failure rule have previously been widely used in artificial intelligence systems (for example, PLANNER, various non-monotonic logics).

This research was partially supported by a grant from the Australian Computer Research Board.

While the negation as failure rule is intuitively appealing, it is preferable to find some firm theoretical foundation for it. In particular, we would like ~A to be a logical consequence of something connected with P. Clark [2] showed that the "something" is the completion of P, denoted by comp(p), which is essentially P together with the only-if halves of each of its clauses, plus some axioms to constrain the equality predicate. Clark showed that if A is in the finite failure set of P, then -A is a logical consequence of corap(P). This amounts to a soundness proof of the negation as failure rule. (We note that Clark proves this result for a more general class of logic programs, ones where literals in a clause body may be negated. For this class, the converse of his result is false).

In this paper, we give the corresponding completeness proof of the rule, that is, we show that if ~A is a logical consequence of comp(p), then A is in the finite failure set of P.

In the next section, we discuss what is currently known about finite failure and put our theorem into that context. In the last section, we give the proof of the theorem.

II FINITE FAILURE AND THE COMPLETION OF A PROGRAM

Throughout this paper, P denotes a Horn clause logic program and B(P) the Herbrand base of P. A denotes an arbitrary element of B(P).

We make the usual identification between Herbrand interpretations for P and subsets of B(P) ([1],[5]). Thus, for any Herbrand interpretation, the corresponding subset of the Herbrand base is the set of all ground atoms which are true in the interpretation. The set of all Herbrand interpretations of P is a complete lattice under the partial order of set inclusion. We define the usual mapping T_P from the lattice of Herbrand interpretations to itself as follows. Let I be a Herbrand interpretation. Then

$$T_P(I)=\{A \in B(P) : A \leftarrow B_1,\ldots,B_n \text{ is a ground instance of a clause in P and } B_1,\ldots,B_n \in I\}$$

T_P is clearly monotonic. As in [1], we define $T_P\downarrow\omega$ to be $\bigcap_{n=0}^{\omega} T_P^n(B(P))$.

Next we define the finite failure set of P. The usual definition of finite failure ([1],[2]) is given in terms of finitely failed SLD trees. A general definition of finite failure, independent of any implementation, is given in [4] and we adopt the same definition here.

Definition FF_d, the set of atoms in B(P) which are finitely failed by depth d, is defined as follows:

(a) $A \in FF_0$ if $A \notin T_P(B(P))$.

(b) $A \in FF_d$, for d>0, if for each clause $A' \leftarrow B_1,\ldots,B_n$ in P and for each substitution θ such that $A=A'\theta$ and $B_1\theta,\ldots,B_n\theta$ are ground, there exists k such that $1 \le k \le n$ and $B_k\theta \in FF_{d-1}$.

Definition The finite failure set FF of P is defined as follows: $A \in FF$ if there exists d such that $A \in FF_d$.

Next we give the more usual definition of finite failure.

Definition The SLD finite failure set of P is the set of all ACB(p) for which there exists a finitely failed SLD tree which has <-A at the root.

Now in [1] the following theorem is proved (a much shorter proof of this result is given in [5]): A ia in the SLD finite failure set if and only if AE T }ui. However, it ia easy to show that FF - B(P)\T ui [4] and thus this result of [l] can be considered as a form of soundness and completeness for an SLD implementation of finite failure. However, this is not quite satisfactory: SLD finite failure only guarantees the existence of one finitely failed SLD tree - others may be infinite. The problem is to identify exactly those computation rules which guarantee to find a finitely failed SLD tree, if one exists at all.

Definition A computation rule is a rule which selects the atom to be expanded in the current goal. A computation rule is fair if for every atom B in a derivation using this rule, either (some further instantiated version of) B is selected within a finite number of steps or (some further instantiated version of) B ia in a failed goal. A fair SLD tree is an SLD tree obtained via a fair computation rule.

Then in [4] the following result is proved: AEFF iff, for every fair computation rule, the corresponding SLD tree with <—A at the root is finitely failed. Furthermore, the desirable strong form of completeness is obtained: all fair SLD trees are equivalent in the sense that if any one is finitely failed, all are.

Summarizing the results so far, we have:

Proposition 2.1 The following are equivalent:

(a) A is in the finite failure set.

(b) AE T_p|m.

(c) There exists an SLD tree with <-A at the root which is finitely failed.

(d) Every fair SLD tree with <-A at the root is finitely failed.

Next we give Clark's definition of the completion of a program. Let $p(t_1,\ldots,t_n) \leftarrow B_1,\ldots,B_m$ be a clause in a program P. We will require a new predicate =, whose intended interpretation is the equality relation. The first step is to transform the given clause into

$p(x_1,\ldots,x_n) \leftarrow (x_1=t_1) \wedge \ldots \wedge (x_n=t_n) \wedge B_1 \wedge \ldots \wedge B_m$,

where $x_1,\ldots,x_n$ are variables not appearing in the clause. Then, if $y_1,\ldots,y_d$ are the variables of the original clause, we transform this into

$$p(x_1,\ldots,x_n) \leftarrow \exists y_1 \ldots \exists y_d \; ((x_1=t_1) \wedge \ldots \wedge (x_n=t_n) \wedge B_1 \wedge \ldots \wedge B_m)$$

Now suppose this transformation is made for each clause which has the predicate p in the head. Then we obtain $k \geq 1$ transformed clauses of the form

$$p(x_1,\ldots,x_n) \leftarrow E_1$$
$$\vdots$$
$$p(x_1,\ldots,x_n) \leftarrow E_k$$

where each E_i has the general form

$\exists y_1 \ldots \exists y_d \; ((x_1=t_1) \wedge \ldots \wedge (x_n=t_n) \wedge B_1 \wedge \ldots \wedge B_m)$.

The <u>completed</u> <u>definition</u> of p is then the formula

$$\forall x_1 \ldots \forall x_n \; (p(x_1,\ldots,x_n) \leftrightarrow E_1 \vee \ldots \vee E_k)$$

However, if a predicate q in P does not appear in the head of any clause, the <u>completed definition</u> of q is

$$\forall x_1 \ldots \forall x_n \; \sim q(x_1,\ldots,x_n).$$

To prove his result, Clark needed the following equality axiom schemas:

1. $c \neq d$, for all pairs c,d of distinct constants.
2. $f(x_1,\ldots,x_n) \neq g(y_1,\ldots,y_m)$, for all pairs f,g of distinct functions.
3. $(x_1 \neq y_1) \vee \ldots \vee (x_n \neq y_n) \rightarrow f(x_1,\ldots,x_n) \neq f(y_1,\ldots,y_n)$, for each function f.
4. $f(x_1,\ldots,x_n) \neq c$, for each constant c and function f.
5. $t[x] \neq x$, for each non-variable term $t[x]$ containing x.
6. $x=x$.
7. $(x_1=y_1) \wedge \ldots \wedge (x_n=y_n) \rightarrow f(x_1,\ldots,x_n)=f(y_1,\ldots,y_n)$, for each function f.
8. $(x_1=y_1) \wedge \ldots \wedge (x_n=y_n) \rightarrow (p(x_1,\ldots,x_n) \rightarrow p(y_1,\ldots,y_n))$, for each predicate p.

<u>Definition</u> The <u>completion of</u> P, denoted comp(P), is the collection of the completed definitions for each predicate in P and the above equality axiom schemas.

The result of this paper is as follows:

<u>Theorem</u> If -A. is a logical consequence of comp(p), then' A is in the finite failure set of P.

Using this theorem, proposition 2.1 and Clark's theorem, we obtain the following result:

<u>Theorem</u> A is in the finite failure set of P iff ~A is a logical consequence of comp(p).

III <u>PROOF</u> 0£ *THE* <u>THEOREM</u>

This section contains a proof of our theorem. In fact, we prove the contrapositive of the result. Thus we assume that A is not in the finite failure set of P and prove that comp(p)U{A(has a model. Unfortunately, we cannot restrict attention to Herbrand models. It is easy to construct examples where A is not in the finite failure set and yet comp(P)U (A) has no <u>Herbrand</u> model. Indeed, this is the main difficulty of the proof - to find the right kind of model.

The first task is to generalize the mapping T_P introduced earlier. Let D be a fixed domain of interpretation for P and assume some fixed assignment of constants in P to elements of D and functions in P to functions on D. With all this fixed, we can now obtain a variety of interpretations for P by varying the assignments of the predicates of P. In fact, as for Herbrand interpretations, each such interpretation can be identified with some subset of "atoms" (where the predicate of each "atom" is in P and each argument is in D). We simply make $p(d_1,\dots,d_n)$ true precisely when $p(d_1,\dots,d_n)$ is in this subset.

As before, we can make a complete lattice out of the set of all such interpretations under the partial order of set inclusion. We also define a mapping, again denoted by T_P, from this lattice to itself as follows. Let I be such an interpretation. Then

$T_P(I) = \{p(d_1,\dots,d_n) : B \leftarrow B_1,\dots,B_n$ is a clause in P and there is some assignment of the variables in the clause to elements of D such that with this assignment B is $p(d_1,\dots,d_n)$ and $\{B_1,\dots B_n\} \subseteq I\}$

It is easy to see that T_P is monotonic.

The following proposition, whose proof is straightforward, is a major tool we employ in the proof of our theorem.

Proposition 3.1 Let I be an interpretation of P, let the predicate = be assigned the identity relation on the domain of I and suppose the equality axioms 1 to 8 are satisfied. Then I is a fixpoint of T_P implies that I, together with = assigned the identity relation, is a model for comp(P).

The domain D of our model, which we shall define later, will be a quotient of the set T of first order terms of P. We define T following Huet [3]. Let $x_1,y_1,z_1,\dots,x_2,y_2,z_2,\dots,\dots$ constitute a denumerable set of variables V, where x,y,z,... are the variables appearing in P. Let F denote the set of functions and constants in P. The set T is defined as the free F-algebra generated by V, that is, a term in T is either a variable in V or a constant in F or is of the form $f(t_1,\dots,t_n)$, for some n-ary function $f \in F$ and some terms $t_i \in T$, $1 \le i \le n$. We shall use the symbols s,t,u and v, possibly indexed, to denote terms.

Let N^* denote the set of all finite lists of positive integers, Λ the empty list and . the usual cons operator. Two lists i and j are independent if neither one is a prefix of the other. For any $t \in T$, we define the set of occurrences of t, $OCC(t) \subseteq N^*$, and, for any $i \in OCC(t)$, the subterm of t at i, t/i, as follows:

(a) If t is a variable or a constant, then (i) $OCC(t)=\{\Lambda\}$ and (ii) $t/\Lambda=t$.

(b) If $t=f(t_1,\dots,t_n)$, then (i) $OCC(t)=\{\Lambda\} \cup \{j.k : 1\le j\le n$ and $k \in OCC(t_j)\}$ and (ii) $t/\Lambda=t$ and $t/j.k=t_j/k$, for all $j.k \in OCC(t)$.

Next, for any $s,t \in T$ and $i \in OCC(s)$, we define the replacement of the subterm of s at i by t, $s[i\leftarrow t]$, as follows:

(a) $s[\Lambda\leftarrow t]=t$.

(b) $f(s_1,\dots,s_n)[j.k\leftarrow t]=f(s_1,\dots,s_j[k\leftarrow t],\dots,s_n)$

To obtain our quotient of T, we now define certain binary relations upon T.

Definition A rewrite X is of the form $\langle i,x,t\rangle$, where $i \in N^*$, $x \in V$ and $t \in T$. It defines a mapping from T into itself: given any $s \in T$,

$$s\langle i,x,t\rangle = \begin{cases} s[i\leftarrow t], & \text{if } i \in OCC(s) \text{ and } s/i=x \\ s, & \text{otherwise.}\end{cases}$$

Two rewrites $\langle i,x,s\rangle$ and $\langle j,y,t\rangle$, where x and y are not necessarily distinct, are independent if i and j are independent. A rewrite X is superfluous for a term t if tX=t.

While a rewrite may closely resemble a substitution, it is important to note that a rewrite may alter at most one instance of any variable in any term. We shall use the symbols X and Y, possibly indexed, to denote rewrites.

Definition Having fixed a set R of rewrites on T,

(a) $s<_n t$ if n is the smallest integer ≥ 0 such that $sX_1X_2\dots X_n=t$, for some $X_i \in R$, $1\le i\le n$.
$s<t$ if $\exists n\ge 0$ such that $s<_n t$.

(b) $s\uparrow_n t$ if n is the smallest integer ≥ 0 such that $uX_1\dots X_k=s$ and $uY_1\dots Y_m=t$, for some $u \in T$, $k+m=n$ and $X_i,Y_j \in R$, $1\le i\le k$, $1\le j\le m$.
$s\uparrow t$ if $\exists n\ge 0$ such that $s\uparrow_n t$.

(c) $s\downarrow_n t$ if n is the smallest integer ≥ 0 such

that $sX_1 \ldots X_k = tY_1 \ldots Y_m$, for some $k+m=n$ and $X_i, Y_j \in R$, $1 \leq i \leq k$, $1 \leq j \leq m$.
$s \updownarrow t$ if $\exists n \geq 0$ such that $s \updownarrow_n t$.

The following proposition is easily verified.

Proposition 3.2

(a) For any two distinct functions f and g and for any sequence of arguments $\bar{u}$ and $\bar{v}$ appropriate to f and g, resp., we have $\sim(f(\bar{u}) \updownarrow g(\bar{v}))$.

(b) For any n-ary function f, $f(s_1, \ldots, s_n) \updownarrow_m f(t_1, \ldots, t_n)$ iff, for all i such that $1 \leq i \leq n$, we have $s_i \updownarrow_{m_i} t_i$, where $m_1 + m_2 + \ldots + m_n = m$.

(c) If X and Y are independent, then $tXY = tYX$.

We now have all the tools needed for the proof of our

Theorem If A is not in the finite failure set of P, then $comp(P) \cup \{A\}$ has a model.

Proof By the results of section 2, we have that any fair SLD tree with root $\leftarrow A$ is not finitely failed. Select any non-failed branch BR in any such tree. Let $G_0 = \leftarrow A, G_1, G_2, \ldots$ denote the goals in BR and let $C_1, C_2, \ldots$ denote the corresponding input clauses. We assume that the variables in P are not indexed and that the variables in each C_i are renamed so that each has index i. Thus the sequence of mgu's $\theta_1, \theta_2, \ldots$, where G_{i+1} is derived from G_i and C_{i+1} using θ_{i+1}, are such that for any variable $x \in V$:

(a) If the bindings x/s and x/t appear in $\{\theta_i\}$, then $s=t$.

(b) For any sequence $\bar{\theta}$ of mgu's in $\{\theta_i\}$ such that $x\bar{\theta} \neq x$, $x\bar{\theta}$ does not contain x.

In other words, for any variable $x \in V$, (a) states that there is at most one binding for x and (b) states that once x has been substituted by any term $t \in T$, no further substitutions upon t can result in a term containing x. We shall say that the substitutions $\{\theta_i\}$ are univocal and acyclic because of (a) and (b), respectively.

We now define a set R of rewrites based on the collection $\{\theta_i\}$:

$R = \{\langle i,x,t \rangle : i \in N^*$ and the binding x/t appears in some $\theta_j\}$.

It is easy to see that, like the θ_i's, the rewrites in R are univocal and acyclic. That is,

(a) If $\langle i,x,s \rangle$ and $\langle j,x,t \rangle$ are in R, then $s=t$.

(b) For any sequence $\bar{X}$ of rewrites in R such that $x\bar{X} \neq x$, $x\bar{X}$ does not contain x.

The main relationship between the substitutions $\{\theta_i\}$ and the rewrites R which we employ is this: if $s\bar{\theta}=t$, for any terms s and t and any sequence of substitutions $\bar{\theta}$, then $s<t$.

We now show that $\updownarrow$ is an equivalence relation on T. That it is reflexive and symmetric is obvious. That it is transitive follows from the "Church-Rosser" property of <: $s \uparrow t$ implies $s \updownarrow t$. This is proved as follows.

Let $s \uparrow_{m+n} t$, say $uX_1 \ldots X_m = s$ and $uY_1 \ldots Y_n = t$, for some $u \in T$. Proceeding by induction on m, we have

(a) Basis: $m \leq 1$. The case $m=0$ is obvious. Now suppose $m=1$. If X_1 is not equal to any Y_i, then X_1 is independent of all the Y_i and, by n applications of proposition 3.2c, $sY_1 \ldots Y_n = tX_1$ and we are done. Otherwise, $X_1 = Y_j$, for some j where $1 \leq j \leq n$. By j-1 applications of proposition 3.2c, we have that $uY_1 \ldots Y_j = uY_jY_1 \ldots Y_{j-1}$. Thus $sY_1 \ldots Y_{j-1}Y_{j+1} \ldots Y_n = t$.

(b) Induction step: $m>1$. Let j satisfy $1 \leq j < m$ and let $s' = uX_1 \ldots X_j$. By the induction hypothesis, $s' \updownarrow t$, say $s' < v'$ and $t < v'$. (see Figure 1). Note that $s \uparrow v'$. Thus by again using the induction hypothesis, we have that $s \updownarrow v'$, say $s < v$ and $v' < v$. By the transitivity of <, we have $t < v$ and we are done.

The transitivity of $\updownarrow$ can be easily seen to follow from the Church-Rosser property (see Figure 2).

We now obtain the domain of our model by defining D to be $T/\updownarrow$, the set of all $\updownarrow$ equivalence classes on T. Next we give the interpretation of the functions in P. Let $[t]$ denote the $\downarrow$ equivalence class of t. For each n-ary function f in P, we assign to f the function from D^n into D defined by $([s_1],\ldots,[s_n]) \rightarrow [f(s_1,\ldots,s_n)]$. That this function is well-defined follows from proposition 3.2b.

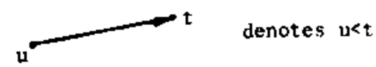
We assign each constant c in P to the equivalence class [c]. Note that if s and t are distinct ground terras, then [s]=[t]. Thus D contains an isomorphic copy of the usual Herbrand universe. This completes the definition of the domain of the model and the assignment of the functions and constants. It remains to give the assignments of the predicates. For this purpose, we are going to use the mapping T_p corresponding to this particular domain and assignment of functions and constants.

First, recall the branch BR in the fair SLD tree for $\leftarrow$A. We construct a set I_0 as follows: $I_0 = \{p([t_1],\ldots,[t_n]) : p(t_1,\ldots,t_n)$ appears in BR$\}$. Next we show that $I_0 \subseteq T_P(I_0)$. Let $p([t_1],\ldots,[t_n])$ be any element in I_0 such that $p(t_1,\ldots,t_n)$ appears in some goal G_i, $i \geq 0$. Because BR is from a fair SLD tree and BR is not failed, there exists a $j \geq 0$ such that $p(s_1,\ldots,s_n) = p(t_1,\ldots,t_n)\theta_{i+1}\theta_{i+2}\cdots\theta_{i+j}$ appears in the goal G_{i+j} and $p(s_1,\ldots,s_n)$ is the selected atom in G_{i+j}. Suppose C_{i+j+1} takes the form $p(u_1,\ldots,u_n) \leftarrow B_1,\ldots,B_m$. By the definition of T_P, $p([u_1\theta_{i+j+1}],\ldots,[u_n\theta_{i+j+1}]) \in T_P(I_0)$. Also, we note that by the abovementioned relationship between substitutions and rewrites and our definition of the $\downarrow$ equivalence classes,

$$\begin{aligned} & p([t_1],\ldots,[t_n]) \\ &= p([t_1\theta_{i+1}\theta_{i+2}\cdots\theta_{i+j}],\ldots,[t_n\theta_{i+1}\theta_{i+2}\cdots\theta_{i+j}]) \\ &= p([s_1],\ldots,[s_n]) \\ &= p([s_1\theta_{i+j+1}],\ldots,[s_n\theta_{i+j+1}]) \\ &= p([u_1\theta_{i+j+1}],\ldots,[u_n\theta_{i+j+1}]), \end{aligned}$$

so that $p([t_1],\ldots,[t_n]) \in T_P(I_0)$. Thus $I_0 \subseteq T_P(I_0)$.

The last step in the definition our model is as follows: using the above result and the



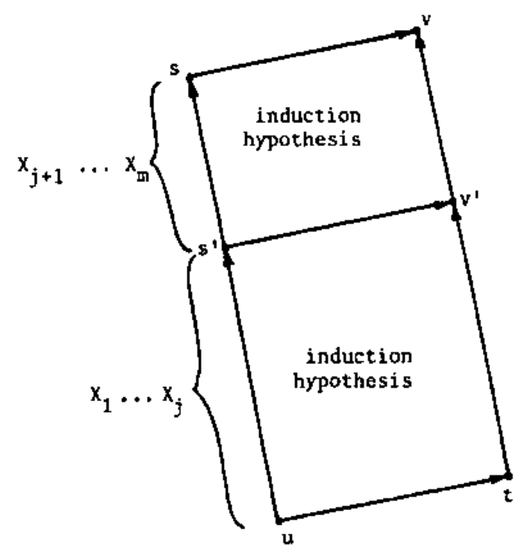


Figure 1.

s$\downarrow$t and t$\downarrow$u implies s$\downarrow$u

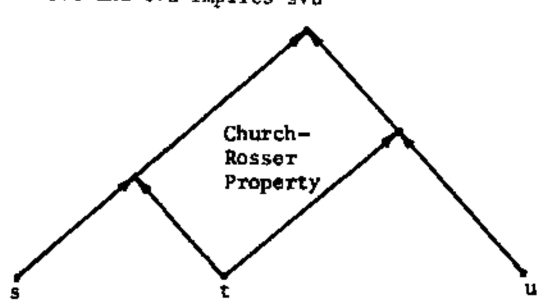


Figure 2.

Knaster-Tarski theorem about fixpoints for monotonic functions, there exists an I such that $I_0 \subset I$ and $I = T_P(I)$. Thus A is true in I.

We assign to ■ the identity relation on D. According to proposition 3-1, it only remains to check that equality axioms 1 to 8 are satisfied. Axioms 6 to 3 are obviously satisfied because - is assigned the identity relation. Axioms 1 and 4 are satisfied because every rewrite is superfluous for constants. Axiom 2 is satisfied by proposition 3-2a and axiom 3 by proposition 3.2b. Only axiom 5 requires some effort. We prove this as follows.

Let $s \subseteq t$ ($s \subset t$) denote that s is a (proper) subterm of t. We now prove that for all $k \geq 0$, $s \subset t$ implies $\neg(s \downarrow_k t)$. This is clearly true for $k=0$. For the induction step, we assume that the statement is false, that is, $s \downarrow_{m+n} t$, say $sX_1 \ldots X_m = tY_1 \ldots Y_n = u$, where $m+n \geq 1$, and obtain a contradiction. Let $s = t/i_0$, for some $i_0 \neq \Lambda$.

Case 1: s is a variable, say x. Clearly, $m \geq 1$. Thus X_1 must be of the form $\langle \Lambda, x, r \rangle$ and, because R is acyclic, u cannot contain x. Thus $n \geq 1$. Now since $t < u$, there exists some j, where $1 \leq j \leq n$ and Y_j is of the form $\langle i_0, x, r \rangle$. Note that the last component in this rewrite must coincide with the last component in X_1 because R is univocal. By j-1 applications of proposition 3.2c, we have that $tY_1 \ldots Y_j = tY_jY_1 \ldots Y_{j-1}$. Let $s' = sX_1$ and $t' = tY_j$, so that $s'X_2 \ldots X_m = t'Y_1 \ldots Y_{j-1}Y_{j+1} \ldots Y_n$. Noting that we have just shown that $s' \subset t'$ and $s' \downarrow_{m+n-2} t'$, we have the desired contradiction of our induction hypothesis.

Case 2: s is not a variable. If s does not contain a variable, then clearly we are finished since all rewrites on s are superfluous. So we assume otherwise and we show that, for some term $t' \subset t$ and some variable x appearing in both s and t', $x \downarrow_i t'$, for some $i \leq m+n$. This will bring us back into case 1.

We proceed by reducing the size (that is, the number of symbols) of s. Let s be of the form $f(s_1, \ldots, s_d)$, for some $d \geq 1$; by proposition 3.2a, t must take a "similar" form, say $f(t_1, \ldots, t_d)$. Since $s \subset t$, we have $s \subseteq t_j$, for some j such that $1 \leq j \leq d$; thus $s_j \subset t_j$ and using proposition 3.2b, $s_j \downarrow_i t_j$, for some $i \leq m+n$. We are now finished, since if s_j is a variable we are back in case 1 and, if otherwise, we apply the process again.

Thus axiom 5 is satisfied and the proof of the theorem is finished.

REFERENCES

[1] Apt, K.R. and van Emden, M.H., "Contributions to the Theory of Logic Programming", JACM, 29, 3(July 1982), 341-862.

[2] Clark, K.L., "Negation as Failure", in Logic and Databases, H. Gallaire and J. Winker (eds), Plenum Press, New York, 1973, 293-322.

[3] Huet, G., "Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems", JACM, 27, 4(Oct. 1980), 797-821.

[4] Lassez, J-L. and Maher, M.J., "Closures and Fairness in the Semantics of Programming Logic", Theoretical Computer Science, to appear.

[5] Lloyd, J.W., "Foundations of Logic Programming", TR 82/7, Department of Computer Science, University of Melbourne.