A LINEAR AXIOMATIZATION OF NEGATION AS FAILURE

SERENELLA CERRITO

▷ This paper is concerned with the axiomatization of success and failure in propositional logic programming. It deals with the actual implementation of SLDNF in PROLOG, as opposed to the general nondeterministic SLDNF evaluation method. Given any propositional program P, a linear theory $\mathbf{LT}_P$ is defined (the linear translation of P) and the following results are proved for any literal A: *soundness of PROLOG evaluation* (if the goal A PROLOG-succeeds on P, then $\mathbf{LT}_P \vdash_{\text{lin}} A$, and if A PROLOG-fails on P, then $\mathbf{LT}_P \vdash_{\text{lin}} A^{\perp}$), and *completeness of PROLOG evaluation* (if $\mathbf{LT}_P \vdash_{\text{lin}} A$, then the goal A PROLOG-succeeds on P, and if $\mathbf{LT}_P \vdash_{\text{lin}} A^{\perp}$, then A PROLOG-fails on P). Here $\vdash_{\text{lin}}$ means provability in linear logic, and $A^{\perp}$ is the linear negation of A. ◁

INTRODUCTION

This paper is concerned with the axiomatization of success and failure in propositional logic programming. One wants to prove something like

A succeeds iff A is provable,

A fails iff the negation of A is provable,

where "provable" means provable in a certain theory Th(P) depending on the program P that we are considering.

A basic choice has to be made: either

(i) we decide—as in most of the literature (see [11] for a general reference)—to axiomatize the general SLDNF method (or a variant of it, as for example the algorithm studied in [16]), or

(ii) we decide to treat the implemented version of SLDNF, i.e. PROLOG.

Address correspondence to Serenella Cerrito, Universitè de Paris-Sud, L.R.I., Bat. 490, Centre d'Orsay, 91405 Orsay Cedex, France.

Received June 1988; accepted October 1989.

655 Avenue of the Americas, New York, NY 10010 0743-1066/92/$3.50

Here we choose the second alternative. (The only works of this kind are [12] and [15], as far as we know.)

The fact that (i) and (ii) lead to different answers is illustrated by the different treatment of the notion of failure for a propositional conjunctive query A AND B:

A AND B fails w.r.t. SLDNF when either A fails or B fails;

A AND B fails w.r.t. PROLOG when either A (which is the first subgoal tested) fails, or A succeeds but B fails.

(Thus if A loops and B fails, the query A AND B fails w.r.t. SLDNF but does not fail w.r.t. PROLOG.)

There is a second choice to be made, namely between

(iii) a nonlogical axiomatization and

(iv) a logical one.

In the first case, what is given is a precise formal description of the evaluation algorithm considered, but no logical relation (in terms of connectives) between the several queries of a program is exhibited. This is the case, for example, of the formal calculus proposed in [12]. In the second case, one should use, if possible, a logical operation of negation to exchange success and failure, a logical operation of disjunction to express, for example, a statement like "either A fails or A succeeds," etc. Moreover, the axiomatization Th(P) proposed for a given program P should be as modular as possible, in the sense that a local modification of the program P should induce just a local modification of Th(P). For example, Clark's completion, which has been proposed as an axiomatization w.r.t. SLDNF, is modular in the sense that if one modifies just the clauses of head A of a given program P, only the component "definition of A" has to be modified in Comp(P) [2].

Here, we make the choice (iv). Now the question is: which logic may one use to formulate Th(P)? There are some difficulties to face:

(a) Classical logic cannot be used, since there are situations where we need a commutation of the connectives with provability which classical logic does not provide: typically, the classical provability of $A \vee B$ does not imply that either A is provable or B is provable (see Example 1 below).

(b) Intuitionistic logic obviates the above difficulty but is far from being free of defects. In contrast with classical logic, it lacks an involutive negation (i.e., the double negation of A is not equivalent to A); therefore it is not able to express the exchange of success and failure. Moreover, it joins classical logic in leading to some undesired situations which are illustrated by Example 2 below.

In both classical and intuitionistic logic there is a hidden principle, namely Gentzen's "contraction rule" (see [3], [6], or [8]), which seems to be responsible for some of the inadequacies of the axiomatizations so far proposed (for example Clark's completion: see the examples below). When we remove contraction (and also another rule, namely weakening), we obtain linear logic (see [5] and [7]). This last is the logic that we have chosen.

This paper proves the following result by using linear logic: given any propositional PROLOG program P, there is a theory LT_P such that for any propositional

literal A one has:

(a) *Soundness of propositional PROLOG evaluation.* If A PROLOG-succeeds on P, then A is a theorem of $\mathbf{LT}_P$, and if A PROLOG-fails on P, then the negation of A is a theorem of $\mathbf{LT}_P$.

(b) *Completeness of propositional PROLOG evaluation.* If A is a theorem of $\mathbf{LT}_P$, then A PROLOG-succeeds on P, and if the negation of A is a theorem of $\mathbf{LT}_P$, then A PROLOG-fails on P.

Now let us go into more detailed explanations. First of all, let us see some examples which show the kind of difficulties that classical logic and intuitionistic logic cannot solve.

Example 1. Let P_1 be the program

(1) $R :- P$

(2) $R :-$ NOT(P)

(3) $P :- P$

The query R PROLOG-loops on this program (and indeed, also SLDNF-loops: the specific search strategy used by PROLOG does not play an essential role here). Now, given the way in which negation is implemented (namely by the negation-as-failure rule), one may expect that an axiomatization via classical logic of this program will contain something like

$$R \leftrightarrow (P \vee \neg P) \quad \text{and} \quad P \leftrightarrow P$$

as theorems (these are actually the axioms of Clark's completion). Thus, R will be a theorem, since $P \vee \neg P$ is classically valid. The point is that PROLOG's internal logic is such that a disjunction $A \vee B$ is "provable" if and only if either A is "provable" or B is "provable", and classical disjunction does not behave like that. Notice that any proof of the excluded-middle principle in the Gentzen formalization of classical logic does essentially use the contraction rule.

Example 2. Let P_2 be the trivial program whose only clause is $A :-$ NOT(A). The query A PROLOG-loops for this program (and, indeed, SLDNF-loops too). However, it is reasonable to ask for $A \leftrightarrow \neg A$ to be a theorem of any logical axiomatization of this program (once again, this is the case for Clark's completion), so that if the underlying logic is classical or intuitionistic, we will get an inconsistent theory which trivially proves A. Notice that in the absence of the contraction rule the provability of

$$A \rightarrow \neg A \quad \text{and} \quad \neg A \rightarrow A$$

does not lead to the provability of every formula (see Section III for a more detailed discussion).

So far, we have seen two difficulties which haunt the axiomatization of some programs where the specific PROLOG search strategy does not play any essential role (which therefore also exploit the possibility of fully axiomatizing SLDNF via classical or intuitionistic logic). But there are also problems which are specific to PROLOG and which are illustrated by the following example.

Example 3. Let P_3 be the program whose only clause is $A :- A, B$. Given the selection rule of standard PROLOG, which treats literals in the body of a clause from left to right, we have that the query A loops on P_3. (Of course, if another selection rule were chosen, A would fail.) A fair axiomatization of this PROLOG program should be able to express the following facts:

(1) A fails if either A fails or A succeeds and B fails,

(2) B fails.

Now, a translation of (1) and (2) into logic language will give us

(1) $\neg A \vee (A \wedge \neg B) \rightarrow \neg A$,

(2) $\neg B$,

and if one uses classic or intuitionistic logic, one will prove $\neg A$.

The above examples should clarify why we have chosen linear logic for our axiomatization rather than classical or intuitionistic logic. The kind of proof theory used is linear sequent calculus; it is described in this paper, but some familiarity with Gentzen classical and intuitionistic calculi is assumed (see [3], [6], or [8]).

We will use the sequent notation

$$A_1, \ldots, A_n \vdash B$$

rather than the standard expression $B :- A_1, \ldots, A_n$ to denote a clause whose head is B and whose body is $A_1, \ldots, A_n$.

Plan of the Paper

This paper is organized as follows:

Section I: Description of the tools that we use (quick description of linear logic and definition of the system LL).

Section II: Formulation and proof of our results.

Section III: Discussion.

I. THE TOOLS THAT WE USE: LINEAR LOGIC

We give here a quick survey of the basic ideas of linear logic, and we introduce the formal system that we will use. For a more precise account of linear logic see [5] and [7].

Linear sequent calculus—as well as the better-known intuitionistic sequent calculus—is essentially a modification of the Gentzen classical sequent calculus. A classical sequent is an expression of the form $\Gamma \vdash \Delta$, where Γ and Δ are finite sequences of formulas. A classical sequent

$$G_1, \ldots, G_n \vdash D_1, \ldots, D_m$$

is true when the implication

$$G_1 \wedge \cdots \wedge G_n \rightarrow D_1 \vee \cdots \vee D_m$$

is true. In an intuitionistic sequent Δ contains at most one formula. This restriction has an immediate effect on the inference rules; for example, in the intuitionistic calculus one no longer has the right-contraction rule. The modifications brought about by the linear approach to the classical calculus are more dramatic: basically the sequents keep the same form as in the classical case, but the structural rules of weakening and contraction are eliminated from the calculus, and this seemingly local modification has far reaching consequences for the structure of the whole calculus and for the very nature of the logical connectives.

But let us go to the core of the matter. In the presence of the structural rules of weakening,

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}(rW),$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}(lW),$$

and contraction,

$$\frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}(rC),$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}(lC),$$

there are two equivalent formulations for the right rule for conjunction, namely the *multiplicative* formulation

$$\frac{\Gamma \vdash A, \Delta \quad \Lambda \vdash B, \Pi}{\Gamma, \Lambda \vdash A \wedge B, \Delta, \Pi}(r\wedge)$$

and the *additive* formulation:

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}(r\wedge^*).$$

However, once weakening and contraction are dropped, $(r\wedge)$ and $(r\wedge^*)$ are no longer equivalent; they actually correspond to two different connectives (the linear conjunctions), namely

$\otimes$ (read "times"),

$\&$ (read "with").

Of course, these two distinct conjunctions do not represent truth functions, because it is well known that there are exactly 16 binary boolean operators and that all of them are definable in terms of standard negation, conjunction, and disjunction. Evidently, the semantics of the linear connectives $\otimes$ and $\&$ is not the usual Tarski one (see [5] for a formal definition of such semantics). However, it is not so surprising that there are propositional connectives which are not truth-functional: the meaning of intuitionistic connectives cannot be explained in terms of truth either, but rather via Heyting's semantics of proofs (see [8]).

The difference between the two linear conjunctions may be roughly explained as follows. Linear logic is sensitive to how many times a formula is used as hypothesis in a proof; thus, using a formula C twice as hypothesis in a linear proof of a

formula A from some hypotheses Γ actually counts as using *two* hypotheses of the form C. In other words, a set of hypotheses Γ should be seen as a set of *occurrences* of formulas rather then as a set of formulas. Now, in order to have a proof $\mathscr{D}$ of $A \otimes B$ from some hypotheses Γ, a proof $\mathscr{D}_1$ of A and a proof $\mathscr{D}_2$ of B have to be *joined*; Γ will be the set of occurrences of formulas obtained by taking the union of the hypotheses used in $\mathscr{D}_1$ and the hypotheses used in $\mathscr{D}_2$. For example, if we have a proof of A which uses the hypothesis C exactly once and a proof of B which uses the hypothesis C exactly once:

$$\begin{array}{cc} C & C \\ \vdots\,\mathscr{D}_1 & \vdots\,\mathscr{D}_2 \\ A & B \end{array}$$

then we have a proof of $C, C \vdash A \otimes B$, but we do not have a proof of $C \vdash A \otimes B$, because C is used twice in the pair $(\mathscr{D}_1, \mathscr{D}_2)$. On the other hand, in order to have a proof $\mathscr{D}$ of $A \,\&\, B$ from Γ, we must have a proof $\mathscr{D}_1$ of A and a proof $\mathscr{D}_2$ of B where each one of the two proofs uses exactly Γ as set of hypotheses; $\mathscr{D}$ "proposes a choice" between $\mathscr{D}_1$ and $\mathscr{D}_2$. Thus, if we take $\mathscr{D}_1$ and $\mathscr{D}_2$ as in the above example, we have a proof of $C \vdash A \,\&\, B$, but we do not have a proof of $C, C \vdash A \,\&\, B$, because C is used once in $\mathscr{D}_1$ and the same happens with $\mathscr{D}_2$. In other terms, $C \vdash A \,\&\, B$ means "using C once, you can get either A or B, as you wish".

In [7] Girard suggests the following story to give some intuition about the meaning of the two linear conjunctions: take C to be "I spend one dollar", A to be "I can get a packet of Marlboros", B to be "I can get a packet of Camels"; then $C \vdash A \otimes B$ may be read as "For one dollar I can get a packet of Marlboros and a packet of Camels" and is false, where $C, C \vdash A \otimes B$ may be read as "For two dollars I can get a packet of Marlboros and a packet of Camels". On the other hand, $C \vdash A \,\&\, B$ says "For one dollar I can get a packet of Marlboros; for one dollar I can also get a packet of Camels; however, I must make a choice between the two possibilities because one dollar suffices just for one packet of cigarettes". Notice that & is not a disjunction: in order to prove $A \,\&\, B$ both a proof of A and a proof of B must be available, and the following linear sequents are valid: $A \,\&\, B \vdash A$ and $A \,\&\, B \vdash B$.

Linear negation is denoted by "$^\perp$" (read "orthogonal") and has the property for any formula A, $A^{\perp\perp}$ means exactly the same thing as A (from this point of view, it has a rather classical flavor).

Classical disjunction obeys De Morgan's rule:

$A \vee B$ is logically equivalent to $\neg(\neg A \wedge \neg B)$.

Corresponding to the two linear conjunctions $\otimes$ and & we have two linear disjunctions, namely:

$\parr$ (read "par")

$\oplus$ (read "plus")

which satisfy de Morgan's rules:

$A \parr B$ is logically equivalent to $(A^\perp \otimes B^\perp)^\perp$;

$A \oplus B$ is logically equivalent to $(A^\perp \,\&\, B^\perp)^\perp$.

We have two right rules for the "plus" disjunction:

$$\frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ (r\oplus 1),$$

$$\frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \oplus B, \Delta}\ (r\oplus 2).$$

It is easy to see from the above rules that $\oplus$ behaves as an intuitionistic disjunction: a proof $\mathscr{D}$ of $A \oplus B$ from the empty set of hypotheses is either a proof of A or a proof of B.

On the other hand, the "par" disjunction has a rather classical flavor: a proof of $A \parr B$ tells us that either A holds or B holds but does not tell us which one. Its right rule is the following:

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \parr B, \Delta}\ (r\parr)$$

A proof $\mathscr{D}$ of $A \parr B$ from Γ behaves in such a way that if a proof of $A^{\perp}$ from Γ' is provided, then a proof of B from Γ, Γ' is obtained, and if a proof of $B^{\perp}$ from Γ'' is provided, then a proof of A from Γ, Γ'' is obtained.

The connectives $\otimes, \parr$ are called *multiplicative* connectives, while $\oplus$ and $\&$ are the *additive* ones. Another multiplicative connective is linear implication, which is defined by

$A \multimap B = A^{\perp} \parr B$ (read "A linearly implies B").

The intended meaning of a linear sequent

$$G_1, \ldots, G_n \vdash D_1, \ldots, D_m$$

is

$$G_1 \otimes \cdots \otimes G_n \multimap D_1 \parr \cdots \parr D_m.$$

So far we have seen the propositional linear connectives. With linear logic one wants to be able to make finer distinctions than in classical or intuitionistic logic (for example between $\otimes$ and $\parr$); however, one does not want to lose the expressive power of intuitionistic calculus. This is why the so-called *exponential* modalities:

! (read "of course")

? (read "why not?")

are introduced: by their means it is possible to translate intuitionistic logic into linear logic (see [5]). Intuitively, "$!A$" means that we can have as many occurrences of type A as we want (a more suitable name for this modality might be *ad libitum*). The modality ? is the dual of !. Finally, there are the linear quantifiers, which have more or less the standard meaning. However, in this paper we will work just with the finite propositional fragment of linear logic, which contains neither exponentials nor quanitifers.

Linear logic has two proof systems, namely *linear sequent calculus* and *proof nets*; here, we use the first one.

After this quick description of linear logic, we can now give the definition of the formal system which we use. Since any linear sequent

$$G_1, \ldots, G_n \vdash D_1, \ldots, D_m$$

has the same meaning as the right-handed sequent

$$\vdash G_1^{\perp}, \ldots, G_n^{\perp}, D_1, \ldots, D_m,$$

we will write just right-handed sequents. By using this trick, we can formulate the calculus writing just the right rules.

I.1. The Language $\mathbb{L}$

Let $P_1, \ldots, P_i, \ldots$ and $P_1^{\perp}, \ldots, P_i^{\perp}, \ldots$ be infinite sequences of propositional variables. The language $\mathbb{L}$ contains the formulas built out of such propositional variables by using the following binary connectives:

multiplicative connectives: $\otimes, \parr$;

additive connectives: $\with$, $\oplus$.

The linear negation is a defined connective; if $\mathscr{F}$ is a formula of $\mathbb{L}$, its linear negation $\mathscr{F}^{\perp}$ is defined as follows:

for propositional variables, we have two distinct kinds of variables in the syntax, i.e. P_i (positive literal) and $P_i^{\perp}$ (negative literal);

$(P_i^{\perp})^{\perp} = P_i$;

$(\mathscr{F} \otimes \mathscr{F}')^{\perp} = \mathscr{F}^{\perp} \parr \mathscr{F}'^{\perp}$;

$(\mathscr{F} \parr \mathscr{F}')^{\perp} = \mathscr{F}^{\perp} \otimes \mathscr{F}'^{\perp}$;

$(\mathscr{F} \oplus \mathscr{F}')^{\perp} = \mathscr{F}^{\perp} \with \mathscr{F}'^{\perp}$;

$(\mathscr{F} \with \mathscr{F}')^{\perp} = \mathscr{F}^{\perp} \oplus \mathscr{F}'^{\perp}$.

Also linear implication is a defined symbol:

$$\mathscr{F} \multimap \mathscr{F}' = \mathscr{F}^{\perp} \parr \mathscr{F}'.$$

I.2. The System LL

A (right-handed) linear sequent of $\mathbb{L}$ is an expression of the form $\vdash \Gamma$, where Γ is a finite sequence of formulas $G_1, \ldots, G_n$ of $\mathbb{L}$; the implicitly defined meaning of the linear sequent $\vdash \Gamma$ is $G_1 \parr G_2 \cdots G_{n-1} \parr G_n$.

Logical axioms:

$$\vdash A, A^{\perp}.$$

Cut rule:

$$\frac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta} \ (\text{CUT}).$$

Exchange rule:

$$\frac{\vdash \Gamma}{\vdash \Gamma'}\ (\text{EXCH}),$$

where Γ' is a permutation of Γ.

Additive rules:

$$\frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \parr B}\ (\parr),$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\ (\oplus 1),$$

$$\frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ (\oplus 2).$$

Multiplicative rules:

$$\frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\ (\otimes),$$

$$\frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, A \parr B, \Delta}\ (\parr).$$

I.3. Some Properties of the System LL

Proposition 1 (Cut-elimination theorem for LL*). Let $\vdash \Delta$ be a linear sequent, and H be a set of linear sequents of* $\mathbb{L}$. *If $\vdash \Delta$ is* LL*-derivable from the set of nonlogical axioms H, then there is a proof $\mathscr{D}$ of $\vdash \Delta$ from H such that if*

$$\frac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Sigma}{\vdash \Gamma, \Sigma}$$

is a cut in $\mathscr{D}$, then either A or $A^{\perp}$ is a formula of a sequent in H.

PROOF OF PROPOSITION 1. By a straightforward adaptation of the standard proof of cut elimination for Gentzen classical calculus (see [6] or [8]). □

Proposition 2 (Subformula property for LL*). Let $\vdash \Delta$ be a linear sequent, and H be a set of linear sequents of* $\mathbb{L}$. *If $\vdash \Delta$ is* LL*-derivable from the set of nonlogical axioms H, then there is a proof $\mathscr{D}$ of $\vdash \Delta$ from H such that each formula $\mathscr{F}$ occurring in $\mathscr{D}$ is either*:

(i) *a subformula of a formula in Δ, or*

(ii) *a formula in a sequent of H or*

(iii) *the linear negation of a subformula of a formula in a sequent of H.*

PROOF OF PROPOSITION 2. As for Gentzen's calculus, the subformula property is a corollary of the cut-elimination theorem. Let $\mathscr{D}$ be the proof of $\vdash \Delta$ from H given by Proposition 1. A rather straightforward induction on the height h of $\mathscr{D}$ shows that $\mathscr{D}$ satisfies the conditions of Proposition 3. □

Proposition 3 (Distributivity properties).

(i) $\otimes$ *is distributive w.r.t.* $\oplus$, *i.e., any formula of the form*

$$A \otimes (B \oplus C)$$

is logically equivalent to

$$(A \otimes B) \oplus (A \otimes C).$$

(ii) $\parr$ *is distributive w.r.t.* &, *i.e., any formula of the form*

$$A \parr (B \,\&\, C)$$

is logically equivalent to

$$(A \parr B) \,\&\, (A \parr C).$$

(iii) $\otimes$ *is semidistributive w.r.t.* &, *in the sense that any formula of the form*

$$A \otimes (B \parr C)$$

logically implies the formula

$$(A \otimes B) \parr (A \otimes C)$$

(*the converse is not generally true*).

(iv) $\parr$ *is semidistributive w.r.t.* $\oplus$, *in the sense that any formula of the form*

$$(A \parr B) \oplus (A \parr C)$$

logically implies

$$A \parr (B \oplus C)$$

(*the converse is not generally true*).

PROOF OF PROPOSITION 3. The reader can find semantical proofs of the above properties in [5]; here we give syntactical proofs just of the first two properties.

In order to prove property (i), we must prove that:

(a) there is a linear sequent deduction of $(A \otimes B) \oplus (A \otimes C)$ from the hypothesis $A \otimes (B \oplus C)$, and

(b) there is a linear sequent deduction of $A \otimes (B \oplus C)$ from the hypothesis $(A \otimes B) \oplus (A \otimes C)$.

A proof of (a) is just an LL proof of the right-handed sequent

$$\vdash (A \otimes (B \oplus C))^{\perp}, (A \otimes B) \oplus (A \otimes C),$$

i.e. of the sequent

$$\vdash (A^{\perp} \parr (B^{\perp} \,\&\, C^{\perp})), (A \otimes B) \oplus (A \otimes C). \qquad (\alpha)$$

A proof of (b) is just a LL-proof of the right-handed sequent

$$\vdash ((A \otimes B) \oplus (A \otimes C))^{\perp}, A \otimes (B \oplus C),$$

i.e. of the sequent

$$\vdash (A^{\perp} \parr B^{\perp}) \,\&\, (A^{\perp} \parr C^{\perp}), A \otimes (B \oplus C). \qquad (\beta)$$

Proof of (α):

$$\dfrac{\dfrac{\dfrac{\vdash A^{\perp}, A \quad \vdash B^{\perp}, B}{\vdash A^{\perp}, A \otimes B, B^{\perp}}(\otimes)}{\vdash A^{\perp}, B^{\perp}, A \otimes B}(\text{EXCH})}{\vdash A^{\perp}, B^{\perp}(A \otimes B) \oplus (A \otimes C)}(\oplus 1)}{\vdash A^{\perp}, (A \otimes B) \oplus (A \otimes C), B^{\perp}}(\text{EXCH}) \qquad \dfrac{\dfrac{\dfrac{\vdash A, A^{\perp} \quad \vdash C, C^{\perp}}{\vdash A^{\perp}, A \otimes C, C^{\perp}}(\otimes)}{\vdash A^{\perp}, C^{\perp}, A \otimes C}(\text{EXCH})}{\vdash A^{\perp}, C^{\perp}, (A \otimes B) \oplus (A \otimes C)}(\oplus 2)}{\vdash A^{\perp}, (A \otimes B) \oplus (A \otimes C), C^{\perp}}(\text{EXCH})$$

$$\dfrac{\dfrac{\vdash A^{\perp}, (A \otimes B) \oplus (A \otimes C), B^{\perp} \& C^{\perp}}{\vdash A^{\perp}, B^{\perp} \& C^{\perp}, (A \otimes B) \oplus (A \otimes C)}(\text{EXCH})}{\vdash A^{\perp} \& (B^{\perp} \parr C^{\perp}), (A \otimes B) \oplus (A \otimes C)}(\parr) \qquad (\&)$$

Proof of (β):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash B^{\perp}, B}{\vdash B^{\perp}, B \oplus C}(\oplus 1)}{\vdash A, A^{\perp} \quad \vdash B \oplus C, B^{\perp}}(\text{EXCH})}{\vdash A^{\perp}, A \otimes (B \oplus C), B^{\perp}}(\otimes)}{\vdash A \otimes (B \oplus C), A^{\perp}, B^{\perp}}(\text{EXCH})}{\vdash A \otimes (B \oplus C), A^{\perp} \parr B^{\perp}}(\parr) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash C^{\perp}, C}{\vdash C^{\perp}, B \oplus C}(\oplus 2)}{\vdash A, A^{\perp} \quad \vdash B \oplus C, C^{\perp}}(\text{EXCH})}{\vdash A^{\perp}, A \otimes (B \oplus C), C^{\perp}}(\otimes)}{\vdash A \otimes (B \oplus C), A^{\perp}, C^{\perp}}(\text{EXCH})}{\vdash A \otimes (B \oplus C), A^{\perp} \parr C^{\perp}}(\parr)$$

$$\dfrac{\dfrac{\vdash A \otimes (B \oplus C), (A^{\perp} \parr B^{\perp}) \& (A^{\perp} \parr C^{\perp})}{\vdash (A^{\perp} \parr B^{\perp}) \& (A^{\perp} \parr C^{\perp}), A \otimes (B \oplus C)}(\text{EXCH})}{}(\&)$$

In order to prove (ii) we must prove the sequents

$$\vdash (A^{\perp} \otimes (B^{\perp} \oplus C^{\perp})), (A \parr B) \& (A \& C), \qquad (\gamma)$$

$$\vdash (A^{\perp} \otimes B^{\perp}) \oplus (A^{\perp} \otimes C^{\perp}), A \parr (B \& C). \qquad (\delta)$$

It suffices to remark that modulo the replacement of the formulas A, B, C with their duals $A^{\perp}, B^{\perp}, C^{\perp}$, the sequent (γ) is just a permutation of the sequent (β), and the sequent (δ) is just a permutation of the sequent (α).

II. SOUNDNESS AND COMPLETENESS OF PROLOG

II.1. Translating a PROLOG Program P into $\mathbb{L}$

Let P be a normal PROLOG program. Our goal is to provide a "translation" $\mathbf{LT}_P$ of a PROLOG program P in $\mathbb{L}$ in such a way that $\mathbf{LT}_P$ "expresses" what it means for an atomic goal A to PROLOG-succeed with respect to P and what it means for A to PROLOG-fail. We recall that the search strategy over the clauses that we consider is the standard PROLOG one, namely, the clauses are tried in the order in which they are written, and that to evaluate a goal B the atoms in the body of the clause

$$A_1, \ldots, A_n \vdash B$$

are evaluated in the order $A_1, \ldots, A_n$.

Before giving the formal definition of such a translation, let us consider some examples.

Example 4. Let P_4 be a PROLOG program whose clauses with head C are the following (in their order of appearance in P):

(1) $A, B \vdash C$,

(2) $D, E \vdash C$,

(3) $\vdash C$,

(4) $F \vdash C$.

Let C be a goal for P_4. The situation in which C succeeds at the first attempt (namely, when the first clause with head C is tried) can be described by the linear formula:

$$A \otimes B \multimap C \tag{a}$$

because the success of C is caused by the success of A followed by the success of B. When C succeeds at the second attempt (after failure of the first one), we have that either A has failed or A has been successful but B has failed (failure w.r.t. the first clause), and both D and E have been successful, namely,

$$(A^{\perp} \oplus (A \otimes B^{\perp})) \otimes (D \otimes E) \multimap C. \tag{b}$$

When C is successful at the third attempt, we have

$$(A^{\perp} \oplus (A \otimes B^{\perp})) \otimes (D^{\perp} \oplus (D \otimes E^{\perp})) \multimap C. \tag{c}$$

Clearly, PROLOG never tests the last clause.

We define the *success set* of the atom C as the set

$$\mathbf{S}_C = \{a, b, c\}.$$

The program P_4 will halt with failure of C if *all* the clauses of P_4 fail; here, such a situation cannot arise [thanks to the presence of the clause (3)]; thus, the *failure set* of the atom C is empty:

$$\mathbf{F}_C = \varnothing$$

If we apply the distributivity of $\otimes$ with respect to $\oplus$, we can rewrite $\mathbf{S}_C$ as the set

of formulas whose elements are

$A \otimes B \multimap C$,

$A^{\perp} \otimes (D \otimes E) \multimap C$,

$(A \otimes B^{\perp}) \otimes (D \otimes E) \multimap C$,

$A^{\perp} \otimes D^{\perp} \multimap C$,

$A^{\perp} \otimes (D \otimes E^{\perp}) \multimap C$,

$(A \otimes B^{\perp}) \otimes D^{\perp} \multimap C$,

$(A \otimes B^{\perp}) \otimes (D \otimes E^{\perp}) \multimap C$.

Now, we can express each formula in this set as a right-handed atomic linear sequent, thus getting the set $\mathbf{S}_C^{\text{seq}}$ whose elements are

$\vdash A^{\perp}, B^{\perp}, C$,

$\vdash A, D^{\perp}, E^{\perp}, C$,

$\vdash A^{\perp}, B, D^{\perp}, E^{\perp}, C$,

$\vdash A, D, C$,

$\vdash A, D^{\perp}, E, C$,

$\vdash A^{\perp}, B, D, C$,

$\vdash A^{\perp}, B, D^{\perp}, E, C$.

(Notice that formulas in the antecedent of an implication change sign when they are put on the right of a right-handed sequent: $A^{\perp}$ becomes A, and A becomes $A^{\perp}$; also remember that $A^{\perp\perp} = A$.)

Clearly

$\mathbf{F}_C^{\text{seq}} = \varnothing$.

Example 5. Let P_5 be such that the only clauses with head C are:

(1) $A, \text{NOT}(B) \vdash C$,

(2) $D, \text{NOT}(E) \vdash C$.

The success of the negative goal $\text{NOT}(B)$ is the same thing as the failure of B, and its failure is the same thing as the success of B (likewise for E). Thus the success set for C is given by

$$\mathbf{S}_C = \{A \otimes B^{\perp} \multimap C, (A^{\perp} \oplus (A \otimes B)) \otimes (D \otimes E^{\perp}) \multimap C\}.$$

Here we have

$$\mathbf{F}_C = \{(A^{\perp} \oplus (A \otimes B)) \otimes (D^{\perp} \oplus (D \otimes E)) \multimap C^{\perp}\},$$

because P_5 halts with failure of C when all the clauses of P_5 with head C fail. The set $\mathbf{S}_C^{\text{seq}}$ now has as elements

$\vdash A^{\perp}, B, C$,

$\vdash A, D^{\perp}, E, C$,

$\vdash A^{\perp}, B^{\perp}, D^{\perp}, E, C$,

while $\mathbf{F}_C^{\text{seq}}$ has the elements

$\vdash A, D, C^{\perp}$,

$\vdash A, D^{\perp}, E^{\perp}, C^{\perp}$,

$\vdash A^{\perp}, B^{\perp}, D, C^{\perp}$,

$\vdash A^{\perp}, B^{\perp}, D^{\perp}, E^{\perp}, C^{\perp}$.

Let P be a normal PROLOG program whose atoms are all propositional letters, as in the examples above. Let A be one such atom, and let $\mathbf{S}_A^{\text{seq}}$ and $\mathbf{F}_A^{\text{seq}}$ be defined as the above examples suggest (see below for the formal definition). We define the *definition of* A, for the given program P, as the following set of linear sequents:

$$\mathbf{D}_A = \mathbf{S}_A^{\text{seq}} \cup \mathbf{F}_A^{\text{seq}}$$

Let $A_1, \ldots, A_n$ be the atoms in P. The linear translation $\mathbf{LT}_P$ of the program P will be defined by

$$\mathbf{LT}_P = \bigcup_{i=1,\ldots,n} D_{A_i}.$$

One may observe that it would be possible to formulate $\mathbf{LT}_P$ in such a way to have a much smaller number of axioms. For example, in the case of the program P_5 above, one might choose the sequents

$\vdash A \otimes B^{\perp} \multimap C$,

$\vdash (A^{\perp} \oplus (A \otimes B)) \otimes (D \otimes E^{\perp}) \multimap C$

as components of the set $\mathbf{S}_C^{\text{seq}}$, and the sequent

$\vdash (A^{\perp} \oplus (A \otimes B)) \otimes (D^{\perp} \oplus (D \otimes E)) \multimap C^{\perp})$

as the unique component of the set $\mathbf{F}_C^{\text{seq}}$. The advantage of having just sequents which do not contain any connective as nonlogical axioms of the linear theory $\mathbf{LT}_P$ is that it is particularly easy to study the structure of sequent proofs where all the axioms have such a form.

The discussion of the examples given above should suffice to motivate the following definitions. Let $\mathbb{AT}(P)$ be the set of the atoms of a program P, and let A be an element of $\mathbb{AT}(P)$. Let Clauses(A) be the sequence of clauses of P whose head is A, taken in their order of appearance in the program P.

Let $B_1, \ldots, B_m \vdash A$ be the jth element of Clauses(A). Let $(B_i)^{\circ}$ be F if B_i is a positive literal F, and $F^{\perp}$ if B_i is NOT(F).

Definition 1. If m is different from 0, then for $1 \leq z \leq m$

$$\Gamma(A)_z^j = (B_1)^{\circ\perp}, \ldots, (B_{z-1})^{\circ\perp}, (B_z)^{\circ},$$

and for $z = 0$

$$\Gamma(A)_0^j = (B_1)^{\circ\perp}, \ldots, (B_m)^{\circ\perp}.$$

When $m = 0$, $\Gamma(A)_n^j$ is the empty sequence for any positive integer n.

Thus, for example, if the first clause in Clauses(C) is A, NOT(B) $\vdash C$, then

$\Gamma(C)_0^1 = A^\perp, B,$

$\Gamma(C)_1^1 = A,$

$\Gamma(C)_2^1 = A^\perp, B^\perp.$

Definition 2.

(a) If Clauses(A) is empty, then $\mathbf{S}_A^{\text{seq}}$ is empty,
else let k be the length of Clauses(A), and for $1 \le j \le k$, define

$$\mathbf{S}_A^{\text{seq}}(j) = \{\vdash \Gamma(A)^1_{z(1)}, \ldots, \Gamma(A)^{j-1}_{z(j-1)}, \Gamma(A)^j_0, A : 1 \le z(i) \le m(i) \text{ for } i = 1, \ldots, j-1\}.$$

The set of sequents $\mathbf{S}_A^{\text{seq}}(j)$ expresses what it means for A to succeed via the jth clause in Clauses(A). For example, if the first clause of Clauses(A) is

B, NOT(C) $\vdash A$

and the second is

D, NOT(E) $\vdash A$,

then $\mathbf{S}_A^{\text{seq}}(2)$ contains the sequents

$\vdash \Gamma(A)^1_1, \Gamma(A)^2_0, A,$

$\vdash \Gamma(A)^1_2, \Gamma(A)^2_0, A,$

namely the sequents

$\vdash B, D^\perp, E, A,$

$\vdash B^\perp, C^\perp, D^\perp, E, A.$

Now we can define

$$\mathbf{S}_A^{\text{seq}} = \bigcup_{j=1,\ldots,k} \mathbf{S}_A^{\text{seq}}(j).$$

(b) If Clauses(A) is empty, then $\mathbf{F}_A^{\text{seq}} = \{\vdash A^\perp\}$;
else,
if there is a clause in Clauses(A) whose body is empty, then $\mathbf{F}_A^{\text{seq}}$ is empty,
else let k be the length of Clauses(A):

$$\mathbf{F}_A^{\text{seq}} = \{\vdash \Gamma(A)^i_{z(1)}, \ldots, \Gamma(A)^{k-1}_{z(k-1)}, \Gamma(A)^k_{z(k)}, A^\perp : 1 \le z(j) \le m(j) \text{ for } j = 1, \ldots, k\}.$$

Intuitively, a sequent in $\mathbf{F}_A^{\text{seq}}$ says that *if*:

(1) the atoms $B_1, \ldots, B_{z-1}$ in the body of the first clause whose head is A all succeed but B_z fails, and

(2) the atoms $B_1, \ldots, B_{z'-1}$ in the body of the second clause whose head is A all succeed but $B_{z'}$ fails, and

⋮

(k) the atoms $B_1, \ldots, B_{z''-1}$ in the body of the last clause whose head is A all succeed but $B_{z''}$ fails,

then A fails.

For example, if the first clause of Clauses(A) is $B, \text{NOT}(C) \vdash A$ and the second is $D, \text{NOT}(E) \vdash A$, and there are no other clauses with head A, then $\mathbf{F}_A^{\text{seq}}$ contains the sequents

$\vdash \Gamma(A)_1^1, \Gamma(A)_1^2, A^\perp,$

$\vdash \Gamma(A)_2^1, \Gamma(A)_1^2, A^\perp,$

$\vdash \Gamma(A)_1^1, \Gamma(A)_2^2, A^\perp,$

$\vdash \Gamma(A)_2^1, \Gamma(A)_2^2, A^\perp,$

namely, the sequents

$\vdash B, D, A^\perp,$

$\vdash B^\perp, C^\perp, D, A^\perp,$

$\vdash B, D^\perp, E^\perp, A^\perp,$

$\vdash B^\perp, C^\perp, D^\perp, E^\perp, A^\perp.$

Definition 3.

$$\mathbf{D}_A = \mathbf{F}_A^{\text{seq}} \cup \mathbf{S}_A^{\text{seq}},$$

$$\mathbf{LT}_P = \bigcup_{A \in \mathbb{AT}(P)} \mathbf{D}_A.$$

Example 6. Let P_6 be

$C_1 = B, A \vdash E,$

$C_2 = D, C \vdash E.$

The elements of $\mathbf{LT}_{P_6}$ are

$\vdash B^\perp, A^\perp, E,$

$\vdash B^\perp, A, D^\perp, C^\perp, E,$

$\vdash B, D^\perp, C^\perp, E,$

$\vdash B, D^\perp, C, E^\perp,$

$\vdash B, D, E^\perp,$

$\vdash B^\perp, A, D^\perp, C, E^\perp,$

$\vdash B^\perp, A, D, E^\perp,$

$\vdash A^\perp,$

$\vdash B^\perp,$

$\vdash C^\perp,$

$\vdash D^\perp.$

Notice that the translation $\mathbf{LT}_P$ is modular, i.e., if just those clauses of P whose head is A are modified, only the component $\mathbf{D}_A$ of $\mathbf{LT}_P$ will be modified.

II.2. Soundness and Completeness of PROLOG

Let P be a normal propositional PROLOG program and A a literal of P. We get the following theorems:

Theorem 1 (Completeness of PROLOG).

(a) *If* $\vdash A$ *is* LL*-provable from* $\mathbf{LT}_P$, *then the goal* A *PROLOG-succeeds on* P.

(b) *If* $\vdash A^{\perp}$ *is* LL*-provable from* $\mathbf{LT}_P$, *then the goal* A *PROLOG-fails on* P.

Theorem 2 (Soundness of PROLOG).

(a) *If* A PROLOG*-succeeds on* P, *then* $\vdash A$ *is* LL*-provable from* $\mathbf{LT}_P$.

(b) *If* A PROLOG*-fails on* P, *then* $\vdash A^{\perp}$ *is* LL*-provable from* $\mathbf{LT}_P$.

(In the above formulation of our theorems, we identify the NOT operator occurring in a literal A of P with the linear negation operator $^{\perp}$.)

These two theorems may be seen respectively as a completeness result and a soundness result for the PROLOG evaluation algorithm with respect to the notion of linear logical consequence of the set of nonlogical axioms of $\mathbf{LT}_P$; the theory $\mathbf{LT}_P$ gives a semantics to the program P and plays a role w.r.t. PROLOG somewhat similar to the one that Clark's completion is intended to play w.r.t. SLDNF.

To prove Theorem 1 we need a lemma.

Lemma 1. Let P *be a propositional PROLOG program. Let* L *be an expression of the form* F *or* $F^{\perp}$, *where* F *is an element of* $\mathbb{AT}(P)$, *and let* L^* *be the sentence defined as follows*:

if L *is a positive literal* F, *then* L^* *is "*F *PROLOG-succeeds on* P*"*;

if L *is a negative literal* $F^{\perp}$, *then* L^* *is "*F *PROLOG-fails on* P*"*.

For any sequent S *given by*

$$A_1, \ldots, A_n,$$

and any literal A_i *in* S, *let* $S^*(A_i)$ *be the sentence*

"*If* $(A_1^{\perp})^*$ *and* ... *and* $(A_{i-1}^{\perp})^*$ *and* $(A_{i+1}^{\perp})^*$ *and* ... *and* $(A_n^{\perp})^*$ *then* A_1^**"*;

let S^* *be the sentence*

"$S^*(A_1)$ *and* $S^*(A_2)$ *and* ... *and* $S^*(A_n)$".

(a) *Given any sequent* S *in* $\mathbf{LT}_P$, *the sentence* S^* *is true.*

(b) *Given any logical axiom* S, *the sentence* S^* *is true*; *moreover, the truth of* S^* *is stable under application of cut and exchange.*

REMARKS ON LEMMA 1. The intuitive meaning of part (a) of this lemma is the following. For the sake of explanation, let us suppose that our sequent S is such that $A_1, \ldots, A_{n-1}$ are all negative literals and that A_n is positive; for example, let us take $\vdash A^{\perp}, B^{\perp}, C$ as S. Such a sequent belongs to $\mathbf{LT}_P$ for any program P

whose first clause with head C is

$$A, B \vdash C. \tag{1}$$

Let us pick C as the literal of S. The right sequent $\vdash A^{\perp}, B^{\perp}, C$ has exactly the same meaning as the two-sided sequent

$$A, B \vdash C.$$

This remark suggests the following reading for S:

If A PROLOG-succeeds on P and B PROLOG-succeeds on P, then C PROLOG-succeeds on P.

Lemma 1(a) tells us that such a reading is correct. Now let us select the literal $B^{\perp}$ in S. We can write S as

$$A, C^{\perp} \vdash B^{\perp}$$

and read it as saying:

If A PROLOG-succeeds on P and C PROLOG-fails on P, then B PROLOG-fails on P,

which is, once again, correct. Finally, by selecting the literal A we get

$$B, C^{\perp} \vdash A^{\perp}$$

and the reading

If B PROLOG-succeeds and C PROLOG-fails on P, then A PROLOG-fails on P,

which is also correct. In other words, Lemma 1(a) tells us that one can read any sequent in $\mathbf{LT}_P$ as expressing the relations of success and failure between its atoms. In particular, when the sequent S has the form $\vdash A$, then it may be read as "A PROLOG-succeeds", and when S has the form $\vdash A^{\perp}$, then it may be read as "A PROLOG-fails".

Part (b) of the lemma essentially says that if the reading above suggested is correct for the sequents which are the premises of a cut rule or an exchange rule, then it is also correct for the sequent which is the conclusion of such a rule. It is interesting to notice that (b) *would no longer be true if we considered also the contraction rule*. For example, consider the program P whose only clause is $\text{NOT}(A) \vdash A$. The sequent $\vdash A, A$ is an element of $\mathbf{LT}_P$ and may be correctly read as "if A PROLOG-fails on P then A PROLOG-succeeds on P". However, if we contract such a sequent to $\vdash A$, the reading "A PROLOG-succeeds on P" is false. This suggests that an attempt to construct $\mathbf{LT}_P$ as a set of classical or linear sequents would not work (see below how we use Lemma 1 to get a proof of Theorem 1, and see also the discussion of some specific examples in Section III).

PROOF OF LEMMA 1. (a) is almost evident by the construction of $\mathbf{LT}_P$; (b) is also straightforward. □

PROOF OF THEOREM 1 (Completeness of PROLOG). Since a query $\text{NOT}(A)$ succeeds when A fails and fails when A succeeds, and since the linear formula $A^{\perp\perp}$ is the same thing as A, it suffices to prove the theorem for any positive literal A of

P. Thus, let A be a positive literal such that $\vdash A$ is a theorem of $\mathbf{LT}_P$. By Proposition 2, we get that there is a proof $\mathscr{D}$ of $\vdash A$ such that any formula occurring in $\mathscr{D}$ is either A, or a formula occurring in a sequent of $\mathbf{LT}_P$, or the linear negation of a subformula of a formula occurring in a sequent of $\mathbf{LT}_P$. Thus, since sequents of $\mathbf{LT}_P$ are sequents of literals and for any propositional letter P the linear negation of $P^\perp$ is just P, we have that each formula occurring in $\mathscr{D}$ is a literal. It follows that the only rules used in $\mathscr{D}$ are cut and exchange. Thus, by Lemma 1, the sentence A^* is true, that is, A PROLOG-succeeds on P. The same argument shows that if $\vdash A^\perp$ is a theorem of $\mathbf{LT}_P$, then A PROLOG-fails on P. □

PROOF OF THEOREM 2 (Soundness of PROLOG). Let P be a PROLOG propositional program, and L be a literal of P; let $w(L)$ be the number of calls to atomic subgoals in the PROLOG evaluation of L w.r.t. the program P. [For example, if our program contains just the clauses NOT(A), $B \vdash C$, and $\vdash B$, then $w(C) = 2$, because in the evaluation of C we have a call to A and a call to B, while if our program has just NOT(A) $\vdash A$ as a clause, then $w(A)$ and w(NOT(A)) are infinite.]

Once again, since a query NOT(A) succeeds when A fails and fails when A succeeds, and since the linear formula $A^{\perp\perp}$ is the same thing as A, it suffices to prove the theorem for any positive literal A of P. Thus, suppose that A is an atomic query which succeeds or fails; clearly $w(A)$ is finite in both cases. We prove by induction on $n = w(A)$ that if A PROLOG-succeeds then $\vdash A$ is a theorem of $\mathbf{LT}_P$, and if A PROLOG-fails then $\vdash A^\perp$ is a theorem of $\mathbf{LT}_P$.

Base: $n = 0$. If A succeeds without any call to atomic subgoals, it must be the case that the first clause in Clauses(A) is $\vdash A$. Then by construction of $\mathbf{LT}_P$ we have that $\vdash A$ is a sequent in $\mathbf{LT}_P$. If A fails without any call to atomic subgoals, it must be the case that Clauses(A) is empty. Then by construction of $\mathbf{LT}_P$ we have that $\vdash A^\perp$ is a sequent in $\mathbf{LT}_P$.

Inductive step: $n > 0$. *Case 1: A PROLOG-succeeds.* Suppose that A succeeds at the ith attempt, i.e., the ith clause in Clauses(A) causes the success of A. Let the following be the first i clauses in Clauses(A):

$$
\begin{aligned}
&(1) \quad B_1^1, \ldots, B_{m(1)}^1 \vdash A,\\
&\vdots\\
&(i-1) \quad B_1^{i-1}, \ldots, B_{m(i-1)}^{i-1} \vdash A,\\
&(i) \quad B_1^i, \ldots, B_{m(i)}^i \vdash A.
\end{aligned}
$$

For each p such that $1 \leq p \leq i-1$ there is a $z(p)$ such that $B_1^p, \ldots, B_{z(p)-1}^p$ all succeed but $B_{z(p)}^p$ fails, while $B_1^i, \ldots, B_{m(i)}^i$ all succeed. Clearly the evaluation of each one of these queries B_r^k calls a number of atomic subgoals strictly inferior to n. [Notice that some of these queries may be negative literals rather than atoms, so that at first sight it seems we cannot apply the inductive hypothesis. However, a negative query NOT(F) succeeds iff the atom F fails and fails if the atom F succeeds; moreover w(NOT(F)) = $w(F)$.] Thus we can apply the inductive hypothesis to get $\vdash L^+$ for any literal L which belongs to the list

$$B_1^1, \ldots, B_{z(1)}^1, \ldots, B_1^{i-1}, \ldots, B_{z(i-1)}^{i-1}, B_1^i, \ldots, B_{m(1)}^i,$$

where L^+ is defined as follows:

$$L^+ = \begin{cases} F & \text{if } L \text{ is the atom } F \text{ and } L \text{ succeeds,} \\ F^\perp & \text{if } L \text{ is the atom } F \text{ and } L \text{ fails,} \\ F^\perp & \text{if } L \text{ is NOT}(F) \text{ and } L \text{ succeeds,} \\ F & \text{if } L \text{ is NOT}(F) \text{ and } L \text{ fails.} \end{cases}$$

By construction of $\mathbf{LT}_P$ we get that the following is a sequent of $\mathbf{LT}_P$:

$$\vdash \Gamma(A)^1_{z(1)}, \ldots, \Gamma(A)^{i-1}_{z(i-1)}, \Gamma(A)^i_0, A. \qquad (\alpha)$$

[Recall the definitions of Section II.1 to grasp the form of such a sequent, which is an element of the set of sequents $\mathbf{S}^{\text{seq}}_A$ (clause (i).] Thus by a series of cuts we get a proof $\mathscr{D}$ of $\vdash A$ from $\mathbf{LT}_P$.

Let us see an example. Suppose that A succeeds at the second attempt, where the first two clauses in Clauses(A) are

(1) NOT(B), $C \vdash A$,

(2) NOT(E), $D \vdash A$.

Suppose also that $z(1) = 2$, so that NOT(B) succeeds and C fails, while NOT(E) succeeds and D succeeds. Here our sequent α is

$$\vdash B, C, E, D^\perp, A,$$

and our proof $\mathscr{D}$ of $\vdash A$ is obtained by using the sequents

$$\vdash B^\perp,$$

$$\vdash C^\perp,$$

$$\vdash E^\perp,$$

$$\vdash D$$

already obtained by the induction hypothesis to make successive cuts which eliminate the formulas $B, C, E, D^\perp$ from α.

Case 2: A PROLOG-fails. The reasoning is very similar to that of the previous case. Here the role of the nonlogical axiom (α) is played by the sequent:

$$\vdash \Gamma(A)^1_{z(1)}, \ldots, \Gamma(A)^{k-1}_{z(k-1)}, \Gamma(A)^k_{z(k)}, A^\perp, \qquad (\alpha')$$

where k is the length of Clauses(A), and the values of $z(1), \ldots, z(k)$ are given by the actual evaluation of A. Notice that the sequent (α') is a member of the set of sequents $\mathbf{F}^{\text{seq}}_A$ defined in Section II.1. □

III. DISCUSSION AND CONCLUDING REMARKS

It is clear that the notion of success in the standard formulations of SLDNF completeness and soundness is different from our notion of PROLOG success; there, "A succeeds" means that there is a way of performing a search which leads to the answer TRUE for A, while the sense in which the term "success" is used in our theorems is linked to a fixed search procedure, the standard PROLOG one. Our translation of a given PROLOG program P is done in such a way as to build in not only the declarative meaning of the clauses of P, but also a logical

description of such a procedure. Notice that if one wanted to construct different linear translations for propositional PROLOG programs which use different search strategies, in principle this could be done.

Now let us see how our approach deals with the "difficult" examples given in the introduction.

First, we have considered the program P_1:

(1) $P \vdash R$

(2) $\text{NOT}(P) \vdash R$

(3) $P \vdash P$

and we have seen that a good axiomatization of such a program should not have R as a theorem. Now we have that the elements of $\mathbf{LT}_{P_1}$ are

$\vdash P^{\perp}, R,$

$\vdash P, P, R,$

$\vdash P, P^{\perp}, R^{\perp},$

$\vdash P^{\perp}, P$

$\vdash P, P^{\perp},$

and $\vdash R$ is not LL-provable from $\mathbf{LT}_{P_1}$.

Our second example was the program P_2, whose only clause is $\text{NOT}(A) \vdash A$. Here $\mathbf{LT}_{P_2}$ is

$\vdash A, A,$

$\vdash A^{\perp}, A^{\perp}.$

Neither $\vdash A^{\perp}$ nor $\vdash A^{\perp}$ is a theorem of such a theory (contraction is forbidden). As a matter of fact, unlike Clark's completion of P_2, $\mathbf{LT}_{P_2}$ is consistent and admits nontrivial models (see [5] for the model theory of linear logic).

We considered also the program P_3, whose only clause is $A, B \vdash A$. There the problem was that a good axiomatization of such a program should not have the negation of A as a theorem. Our set of sequents $\mathbf{LT}_{P_3}$ has the following elements:

$\vdash A^{\perp}, B^{\perp}, A,$

$\vdash A, A^{\perp},$

$\vdash A^{\perp}, B, A^{\perp},$

$\vdash B^{\perp},$

and the sequent $\vdash A^{\perp}$ is not provable (the sequent $\vdash A^{\perp}, A^{\perp}$ is indeed provable, but we cannot contract such a sequent).

Now let us play the devil's advocate. A false impression that could arise from a quick reading of our work is that the ability of our translation $\mathbf{LT}_P$ to reflect PROLOG success and failure does not depend on the specificity of linear logic: $\mathbf{LT}_P$ is just a paraphrase of how (standard) PROLOG searches through the clauses of P, and if one were replacing linear connectives by classical ones, one would obtain a translation $\mathbf{CT}_P$ which would work as well as $\mathbf{LT}_P$ to get Theorem 1 and Theorem 2. Discussing the meaning of Lemma 1, we have already observed that

the absence of contraction, which is specific to linear logic, does play an essential role in the proof of our results. Moreover we have seen that if we were handling the axioms of $\mathbf{LT}_{P_2}$ as classical or intuitionists sequents, we would get an inconsistent theory. That example was quite interesting because it clearly showed the specific role played by linear logic in our translation, independently of the specific way in which the PROLOG strategy has been axiomatized. Below, we give two additional examples to show once again that if our axioms were treated as classical or intuitionistic sequents, then we would get into trouble; both these examples consider programs where the PROLOG search strategy does play an important role.

Example 7. Let P_7 be the following program:

(1) $A \vdash A$

(2) $A \vdash B$

(3) $\vdash B$

We have $\mathbf{LT}_{P_7} = \{\vdash A^{\perp}, A,\ \vdash A, A^{\perp},\ \vdash A^{\perp}, B,\ \vdash A, B\}$. The corresponding set of classical sequents looks almost the same as $\mathbf{LT}_{P_7}$:

$\mathbf{CT}_{P_7} = \{\vdash \neg A, A,\ \vdash A, \neg A,\ \vdash \neg A, B\ \vdash A, B\}$,

but clearly now these sequents must be handled via classical sequent rules. Although B PROLOG-loops for the above program, we do get a classical proof of the sequent $\vdash B$ from the set of hypotheses $\mathbf{CT}_{P_7}$, namely

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash A, \neg A}{\vdash A \vee \neg A, \neg A}(\vee 2)}{\vdash \neg A, A \vee \neg A}(\text{EXCH})}{\vdash A \vee \neg A, A \vee \neg A}(\vee 2)}{\vdash A \vee \neg A}(\text{CONTR}) \qquad \dfrac{\dfrac{\vdash A, B \quad \vdash \neg A, B}{\vdash A \wedge \neg A, B, B}(\wedge)}{\vdash A \wedge \neg A, B}(\text{CONTR})}{\vdash B}(\text{CUT})$$

Notice that the use of the contraction rule plays an essential role in the above classical proof. The next example shows, once again, that we cannot treat the sequents of our axiomatization as intuitionistic sequents either.

Example 3 Revisited. Let us consider once again the program P_3 seen above, whose only clause is $A, B \vdash A$.

We have already seen that once PROLOG tries to answer the question A, a loop occurs, and that the sequent $\vdash A^{\perp}$ is not LL-derivable from the linear

translation of P_3. (By the way, observe that if the atoms B and A appear in reverse order in the body of the clause, the goal A finitely fails and $\mathbf{LT}_{P_3}$ is modified in such a way that $\vdash A^{\perp}$ is LL-derivable.) Now consider the corresponding intuitionistic translation for P_3:

$$\mathbf{IT}_{P_3} = \{B, A \vdash A, \neg A \vdash \neg A, A, \neg B \vdash \neg A, \vdash \neg B\}.$$

Here is an intuitionistic proof of $\vdash \neg A$ from $\mathbf{IT}_{P_3}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \neg B \quad A, \neg B \vdash \neg A}{A \vdash \neg A}}{A, \neg\neg A \vdash}}{\neg\neg A \vdash \neg A}}{\neg\neg A, \neg\neg A \vdash}}{\neg\neg A \vdash}\,(\text{CONTR})}{\vdash \neg\neg\neg\neg A} \qquad \dfrac{\dfrac{\dfrac{\dfrac{A \vdash A}{A, \neg A \vdash}}{A \vdash \neg\neg A}}{\neg\neg\neg A, A \vdash}}{\neg\neg\neg A \vdash \neg A}}{\vdash \neg A}$$

Let us conclude this discussion with a few words of comparison between our work and some related results.

In [12] and [15] Mints proposes an axiomatization for PROLOG evaluation which applies to the general class of first-order logic programs; PROLOG is sound and complete with respect to such an axiomatization. As we already said, the main difference between Mints's approach and our approach is that Mints's axiomatization is a *formal calculus* which provides a paraphrase of PROLOG evaluation, rather than a *logical theory* which analyses such an evaluation by means of logical operators of negation, conjunction, disjunction, etc. The advantage of Mints's axiomatization is that it is not limited to propositional programs, while its weakness—in our opinion—lies in its rather *ad hoc* nature.

In [10] it is shown that SLDNF evaluation on *allowed* programs (and queries) is sound and complete with respect to Clark's completion if one takes the underlying logic to be a specific version of three-valued logic rather than classical logic. In particular, SLDNF is sound and complete for propositional programs. The obvious difference between Kunen's approach and the approach of the present paper lies in the different aims: here we are interested in the axiomatization of a *specific implementation* of SLDNF, namely PROLOG, rather than in the general SLDNF evaluation. However, in [1] we indeed propose a linear version of Clark's completion as semantics for the general SLDNF nondeterministic evaluation, and we prove the soundness and completeness of SLDNF with respect such a semantics under the same hypothesis on programs and queries as Kunen's, namely allowedness. In that paper we compare extensively the linear-completion approach and the three-valued-completion approach with respect to the problem of finding a good logical axiomatization of SLDNF.

The author would like to thank Nicole Bidoit, who patiently read the first draft of this paper; her remarks lead to substantial improvements. Also, the author wishes to express here indebtedness to Jean-Yves Girard; discussion with him has been essential to the development of the paper.

REFERENCES

1. Cerrito, S., Linear Completion, Rapport de Recherche L.R.I., Univ. Paris XI, 1989.
2. Clark, K. L., Negation as Failure, in H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, 1978, pp. 293–322.
3. Gallier, J. H., *Logic for Computer Science*, Harper & Row, 1986.
4. Gentzen, G., *Collected Works*, Szabo (ed.), North Holland, Amsterdam, 1969.
5. Girard, J. Y., Linear Logic, *Theoret. Comput. Sci.* 50 (1987).
6. Girard, J. Y., *Proof Theory and Logical Complexity*, Vol. 1, Bibliopolis, 1987.
7. Girard, J. Y., Towards a Theory of Geometry of Interaction, in Gray and Scedrov (eds.), *Categories in Logic and Computer Science*, Contemp. Math. 92, AMS, Providence, 1989.
8. Girard, J. Y., Lafont, Y., and Taylor, P., *Proofs and Types*, Cambridge Tracts Theoret. Comput. Sci., Cambridge U.P., Cambridge, 1989.
9. Kunen, K., Negation in Logic Programming, *J. Logic Programming* 4:289–308 (1987).
10. Kunen, K., Signed Data Dependencies in Logic Programs, *J. Logic Programming* 7:231–245 (1989).
11. Lloyd, J. W., *Foundations of Logic Programming*, Springer-Verlag, 1987.
12. Mints, G., Complete Calculus for Pure Prolog (in Russian), *Proc. Acad. Sci. Estonian SSR* 35:367–380 (1986).
13. Shepherdson, J. C., Negation as Failure, *J. Logic Programming* 1:51–79 (1984).
14. Shepherdson, J. C., Negation as Failure II, *J. Logic Programming* 2:185–202 (1985).
15. Shepherdson, J. C., Mints Type Calculi for Logic Programming, Report PM-88-01, School of Mathematics, Univ. of Bristol, 1988.
16. Shepherdson, J. C., A Sound and Complete Semantics for a Version of Negation as Failure, *Theoret. Comput. Sci.* 65(3):343–371 (1989).