Theoretical Computer Science 78 (1991) 189-216
North-Holland

Programming in three-valued logic

J.P. Delahaye and V. Thibau

L.I.F.L., U.A.-C.N.R.S. No. 369, BAT. M3, Université des Sciences et Techniques de Lille Flandres Artois, 59655 Villeneuve d'Ascq Cedex, France

Abstract

Delahaye, J.P and V. Thibau, Programming in three-valued logic, Theoretical Computer Science 78 (1991) 189-216.

The aim of this paper is to propose a logical and algebraic theory which seems well-suited to logic programs with negation and deductive databases. This theory has similar properties to those of Prolog theory limited to programs with Horn clauses and thus can be considered as an extension of the usual theory. This parallel with logic programming without negation lies in the introduction of a third truth value (Indefinite) and of a new non-monotonic implication connective. Our proposition is different from the other ways of introducing a third truth value already used in Logic Programming and databases but it is somehow related to some of them, especially to Fitting's theory. We introduce a "consequence" operator associated with a logic program with negation which extends the operator of Apt and Van Emden. In the case of a consistent program, the post-fixpoints of this operator are the models of the program as they are usually. This operator is related to Fitting's one, the relation being obtained by completing the program. We finally give an operational semantics for a program with negation by the obtention of a three-valued interpreter from a bivalued one.

Introduction

When treating negation in logic programming, many problems must be faced and attempts to solve them may be sorted out more or less according to their faithfulness to negation as failure, which is the negation really used in Prolog [4, 10, 14, 22, 23]. It does not seem possible to stay close to Prolog while having an easy axiomatical semantics: we are not sure to find a sound and complete semantics for negation as failure. A negation other than negation as failure is also used in expert systems working in forwards chaining.

The theory we propose here is quite general and can be applied to both Logic Programming and this negation. It has good theoretical and algebraic properties similar to those obtained in logic programming with negation. The parallel lies in the introduction of a third truth value, Indefinite and a new non-monotonic implication connective denoted by →. We thus obtain a very simple way of treating negation

and a purely axiomatic reading of a program is then possible: we are now able to program by writing axioms and the *specification* of a program becomes the program itself.

Although the deductions obtained in three-valued logic may be too weak (we do not sometimes deduce enough logical consequences of a program), this logic is natural for cases of partial information and it has already been used for a long time in Artificial Intelligence [3, 26]. We have concentrated on the *denotational semantics* of a logic program with negation by introducing an operator associated with logic programs with negation, whose post-fixpoints are exactly the models of the program itself when it is consistent. The least fixpoint of this operator will then be the least model of the program. We also study the relations between our operator, Fitting's one and Apt and Van Emden's one for the case of programs without negation.

Finally, we try to study the *operational semantics* of a program with negation. First, we have the result that all the ground literals being the logical three-valued consequences of a program without negation are exactly the result of the computation of forward chaining. We also give algorithms which compute the smallest model of a program by obtaining a three-valued interpreter from a bivalued one.

In this paper, we restrict ourselves to Herbrand interpretations and we do not study completely all the properties of this three-valued logic. For a more general theory, and particularly the completeness of this three-valued logic, and the expressivity of the connectives of the language, see [24].

The introduction of a third truth value and a new non-monotonic connective can be justified as follows: it is used to give another expressivity to a rule with negation: as soon as a negation is introduced in the body of a rule, we obtain

$$A \leftarrow A_1, \ldots, A_k, \neg B_1, \ldots, \neg B_q,$$

which is logically equivalent to any clause

$$A, B_1, \ldots, B_q \leftarrow A_1, \ldots, A_k.$$

As we know that every first order axioms system may be written with a set of clauses, the question of writing a program as a set of rules is then raised. Two answers may be given. The first one is to keep a set of rules to generalize the SLD-resolution to the SLDNF-resolution, since it is an efficient mechanism. The completeness of the SLDNF-resolution, far from being easy, has already been obtained for special kinds of programs, for instance the hierarchical programs. The second one, which is our answer, is to give a different expressivity to such a formula $(A \leftarrow A_1, \ldots, A_k, \neg B_1, \ldots, \neg B_q)$ by the introduction of a third truth value and a new non-monotonic connective.

Many studies have used this approach and in the next section, we attempt to give a general survey of the research in order to compare it with our own research. It will enable us to give the relations between our implication connective and operator and Fitting's equivalence connective and operator.

Survey of research

Fitting [7, 8, 9] uses three-valued logic for logic programming, which actually means not giving a truth value to any ground atom. In his paper [7], Fitting introduces a third truth value I (Indefinite). He is interested only in programs with positive rules like

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

where A is an atom and B_i are formulas using the symbols $(\wedge, \vee, \neg)$. He defines the truth values of $\neg F$, $F \wedge G$, $F \vee G$, $\exists x\, F$, $\forall x\, G$, in the same way as they are defined here. There is no implication connective, but only an equivalence connective $\cong$ defined by:

$X \cong Y$ is True iff X and Y have the same truth value, False otherwise.

His connective $\cong$ will correspond to our strong equivalence connective, $\leftrightarrow$. A model of a P program is a Herbrand model of its completion Comp(P) in which equality is interpreted by syntactical equality. The completion of a program is obtained normally [4, 14] by replacing each rule

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

by

$$p(x_1, \ldots, x_n) \leftarrow \exists y_1 \cdots \exists y_k \quad (x_1 = t_1 \wedge \cdots \wedge x_n = t_n \wedge B_1 \wedge \cdots \wedge B_m)$$

where $y_1, \ldots, y_k$ are the variables of $t_1, \ldots, t_n$ and of $B_1, \ldots, B_m$.

If D_i denotes the formula: $\exists y_1 \cdots \exists y_k\ (x_1 = t_1 \wedge \cdots \wedge x_n = t_n \wedge B_1 \wedge \cdots \wedge B_m)$, the completion of P is obtained first by putting all the rules having their heads built on the same predicate symbol together, in the following way:

$$p(x_1, \ldots, x_n) \cong D_1 \vee \cdots \vee D_p$$

and secondly by adding to the P program, $\neg p(x_1, \ldots, x_n)$ where p is a predicate symbol of P without being in any head of its rules. Fitting is interested in the semantical interpretation of a program as fixpoints of operators. His order on the set of three-valued Herbrand interpretations is the set inclusion which corresponds to the order $I \leqslant F$, $I \leqslant T$ on $\{T, F, I\}$. With each three-valued Herbrand interpretation i, is associated the mapping tv_i, defined from the set of the language formulas to the set $\{T, F, I\}$, its image on a formula being the truth value of the formula with respect to i. His operator F_{Pr} is defined by its image on a three-valued Herbrand interpretation by

$$F_{\mathrm{Pr}}(i) = \{\mathrm{ato} \mid \text{there is a ground instance ato} \leftarrow B_1, \ldots, B_m \text{ of a rule of Pr such that } \mathrm{tv}_i(B_1, \ldots, B_m) = T\}$$

$$\cup \{\neg\mathrm{ato} \mid \text{for all ground instance ato} \leftarrow B_1, \ldots, B_m \text{ of Pr we have } \mathrm{tv}_i(B_1, \ldots, B_m) = F\}.$$

This operator is monotonic and the fixpoints of F_{Pr} are exactly the models of the completion of Pr. As a union of partial models is not necessarily a three-valued interpretation, the set of the three-valued interpretations is no longer a complete lattice but only a semi-complete one: the intersection of two three-valued Herbrand interpretations is still a three-valued Herbrand interpretation and a consistent union of two three-valued Herbrand interpretations is a three-valued Herbrand interpretation. The mapping F_{Pr} has a least fixpoint which is also the least model of comp(P) and does not have any greatest fixpoint but only several *maximal* ones. There are thus two ways of defining the *semantics* of a logic program:

- the least fixpoint of F_{Pr},
- the optimal fixpoint of F_{Pr} which is the greatest fixpoint less or equal to the intersection of all the maximal fixpoints and which is also the greatest consistent fixpoint.

Our theory is close to Fitting's but we give a new interpretation of the connective $\rightarrow$ and define the notion of Herbrand model of Pr without using its completion: the *specification* of a program will thus be the program itself. We can express its connective $\cong$ with our implication connective, the negation and the conjunction symbol; we can obtain F_{Pr} from our mapping T_{Pr} by completing our program.

Lassez [13] also uses a three-valued logic. The programs considered have only Horn clauses: $A \leftarrow A_1, \ldots, A_n$. His implication connective is Lukasiewicz's one, and has the following truth table:

$A \rightarrow B$ is not True
- if A gets the truth value T and B gets the truth value I ($A \rightarrow B$ gets the truth value I),
- if A gets the truth value T and B gets the truth value F ($A \rightarrow B$ gets the truth value F),
- If A gets the truth value I and B gets the truth value F ($A \rightarrow B$ gets the truth value I).

$A \rightarrow B$ gets the truth value T otherwise.

The formula $P \rightarrow Q$ is not logically equivalent to the formula $\neg P \vee Q$, if P and Q have the truth value I for instance. His order on $\{T, F, I\}$ corresponds to the set inclusion order on the set of the three-valued Herbrand interpretations ($I \leqslant F$, $I \leqslant T$).

The operator τ associated with a logic program P is defined in the following way (a Herbrand interpretation being considered as an application from the set of ground formulas to $\{T, F, I\}$):

- if there is a ground instance of a rule of P with A in its head, $A \leftarrow B_1, \ldots, B_n$, such that $f(B_1 \wedge \cdots \wedge B_n) = T$, then $\tau(f)(A) = T$,
- if, for every ground instance of P with A in its head, $A \leftarrow B_1, \ldots, B_n, f(B_1 \wedge \cdots \wedge B_n) = F$, then $\tau(f)(A) = f(A)$,
- if, for every ground instance of P with A in its head, $A \leftarrow B_1, \ldots, B_n, f(B_1 \wedge \cdots \wedge B_n) \neq V$, and if there is a ground instance of a rule of P with A in its head, $A \leftarrow B_1, \ldots, B_n$, such that $f(B_1 \wedge \cdots \wedge B_n) = I$, then $\tau(f)(A) = \text{glb}(T, f(A))$.

The fixpoints of τ are exactly the models of P. For this operator, the two notions of optimal fixpoint and least fixpoint are the same. This especially comes from the fact that the interpretation constantly equal to True, is always a model of a program with Horn clauses. It follows that neither the least model nor the optimal model of a program (which are both included in the maximal interpretation constantly equal to True), take the truth value False. The negative information, which results from the completed program, is not really represented in this semantics model. So this semantics does not seem well-suited to logic programming with negation.

Mycroft [18] also uses three-valued logic. However, the clauses considered are definite program clauses, in order to show that a many valued logic is better suited to the SLD-resolution. He introduces the same operator as Apt and Van Emden [28]. The truth tables of the connectives $\wedge$, and $\vee$ are the same as the ones we use. He does not introduce any implication connective. The three-valued part of his paper is essentially contained in Fitting's paper.

Kunen [11] uses three-valued logic only for programs with positive rules. His work is very close to Fitting's because he considers the completion of a program and has the same operator. According to him, the semantics of a logic P program, is not the least fixpoint of the operator T but $T\uparrow\omega$, which is not always a model of P. From our point of view, a good semantics should be expressed in terms of axioms and models.

Finally, Przymusinski [19, 20], has a new vision on three-valued logic. In [19], he introduces three truth values and an order on $\{T, F, I\}$, $F \leqslant I \leqslant T$ which does not correspond to the set inclusion order on the set of the Herbrand three-valued interpretations. A Herbrand interpretation $\mathcal{I}$ is given by two sets of ground atoms: $\mathcal{T}$ and $\mathcal{F}$, without saying that $\mathcal{T} \cap \mathcal{F} = \emptyset$, which allows a ground atom to have two truth values. In this context, *negation is no longer monotonic* while in our theory, the monotonicity of negation enables us to extend the monotony property, which has many consequences. His implication connective, although it never gets the truth value Indefinite (I), is not the same as ours. It is defined by

$$\mathrm{tv}_i(A \to B) = T \text{ iff } \mathrm{tv}_i(B) \geqslant \mathrm{tv}_i(A), \qquad \mathrm{tv}_i(A \to B) = F \text{ otherwise.}$$

Our implication connective gets the truth value True more often than this one and it follows that we will have more models in our theory. The logic programs he considered are definite ones with only positive rules ($A \leftarrow L_1, \ldots, L_n$, A is an atom and L_i are literals). The operator associated with each program is built on two levels; this operator is monotonic and its least fixpoint is the least model of P. Our operator is built in a much more simple way, as an extension of Apt and Van Emden's. However, the models obtained with his implication connective are models for us too.

Manna and Shamir [16, 17], have developed a theory of optimal fixpoints for any monotonic functional defined on a set of partial functions. This theory may interest us because the three-valued Herbrand interpretations we consider, may be

considered as partial functions from the set of the ground atoms to the set $\{T, F\}$ (not being defined where they get the truth value Indefinite). We adapt what could have been done for partial functions to three-valued logic. However, we do not develop this notion in this paper (see [24]).

Another development of three-valued logic for logic programming seems possible [5, 24]. We try to present some aspects of it in this paper.

1. An extension of the bivalued logic for programs with negation

We first recall the properties obtained for logic programs without negation.

L is a logical language with the connectives $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\neg$, and the quantifiers $\forall$, $\exists$. The set Her(L) denotes the set of all the ground literals (atoms or negations of an atom) not containing the equality predicate, her(L) being the set of the positive ground literals and $\neg$her(L) the negative ones, the set UNI(L) is the set of all the ground terms. These notations are available for every language L, even when the language is extended.

When studying programs without negation with rules like ato$\leftarrow$for where the formula for uses the symbols $\forall$, $\exists$, $\wedge$, $\vee$, we use the notion of bivalued Herbrand interpretations. The set $2^{\text{her}(L)}$ of all the bivalued Herbrand interpretations, which is the set of all the subsets of her(L), is ordered by the set inclusion. This order corresponds to the order induced by $F \leqslant T$ on the set of the functions from For(L) to $\{T, F\}$, with the correspondence $\psi : 2^{\text{her}(L)} \rightarrow \{\text{Functions from For}(L) \text{ to } \{T, F\}\}$, such that $\psi(i) = \text{tv}_i$ (where $\text{tv}_i(\text{for})$ denotes the truth value of a formula for of For(L), defined for an atom ato by $\text{tv}_i(\text{ato}) = \text{True}$ if ato $\in i$. False otherwise, and extended as usual for any formula for).

The first important property we will have to extend is a *monotony property*. If $F(L)$ denotes the set of formulas making the body of a rule of a program without negation (using the symbols $\forall$, $\exists$, $\wedge$, $\vee$), the mapping $\phi : 2^{\text{her}(L)} \rightarrow \{\text{Functions from } F(L) \text{ to } \{T, F\}\}$, such that $\phi(i) = \text{tv}_i / F(L)$ is monotonic for the set inclusion order on $2^{\text{her}(L)}$ and the order induced by $F \leqslant T$ on the functions from $F(L)$ to $\{T, F\}$. It means that if the formula for makes the body of a rule of a program without negation, i.e. is a formula using the symbols $\forall$, $\exists$, $\wedge$, $\vee$, then

$$i \subseteq j \text{ implies that } \text{tv}_i(\text{for}) \leqslant \text{tv}_j(\text{for}).$$

This monotony property is important because of the following consequences:

- From this monotony property and the truth table of the implication connective ($\Rightarrow$), follows the intersection model property: an intersection of Herbrand models of a program is still a model of this program.
- From this monotony property, it follows that the Apt and Van Emden "consequence" operator denoted by B_{Pr}, which associates the interpretation $B_{\text{Pr}}(i) = \{\text{ato} \in \text{her}(L) \mid \text{there is a ground instance of a rule of Pr, ato} \leftarrow \text{for such that } \text{tv}_i(\text{for}) = \text{True}\}$ with each bivalued interpretation i, is monotonic. The operator

B_{Pr} being monotonic, it admits a least fixpoint according to the Birkhoff-Tarski theorem, which is equal to $B_{Pr}\uparrow\alpha$ for some α ordinal [[1]] (where $B_{Pr}\uparrow\alpha$ is defined as usual by transfinite induction with $B_{Pr}\uparrow 0 = \emptyset$).

- As by the truth table of the implication connective ($\Rightarrow$), the postfixpoints of this operator are exactly the models of a program [[2]], with the monotony of B_{Pr} and [[2]], we obtain that $B_{Pr}\uparrow\alpha \subseteq i$, for each i model of a Pr program, by transfinite induction [[3]].
- From [[1]] and [[3]], we are now able to deduce that the least fixpoint of B_{Pr} is the least model of Pr [14]. We thus have a denotational semantics for a program without negation which is the least fixpoint of the operator being associated with it. Moreover this least fixpoint is $B_{Pr}\uparrow\omega$ when the program is definite.

As soon as we introduce negation, even only in the body of a rule, the intersection model property is no longer true. For example, if Pr is the following program, where p, q, r are predicate symbols and f a function symbol:

$$q(x) \leftarrow p(x), r(x)$$

$$q(f(x)) \leftarrow p(x), \neg r(x)$$

$$p(x) \leftarrow .$$

If we want to study programs having rules with negation, like lit$\leftarrow$ for where lit is a literal and the formula for uses the symbols $\forall$, $\exists$, $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, we would like to extend the monotony property to the formulas making the body of a rule of such programs. This is why we introduce a third truth value *Indefinite* and we associate the notion of three-valued interpretation with this truth value. To make the parallel easier with the bivalued case, we will define a bivalued Herbrand interpretation as a peculiar case of a three-valued one.

Definition 1.1. A three-valued interpretation is a consistent subset i of the set Her(L), that is:

$$\forall \text{ato} \in \text{her}(L), \quad \text{ato} \in i \text{ implies that } \neg\text{ato} \notin i.$$

The set of all the three-valued Herbrand interpretations is denoted by IHT(L).

A bivalued Herbrand interpretation is a three-valued complete one, that is:

$$\forall \text{ato} \in \text{her}(L), \quad \text{ato} \in i \text{ or } \neg\text{ato} \in i.$$

The set of all the bivalued Herbrand interpretations is denoted by IHB(L).

To come back to the usual definition, we use the mapping $\text{pos}: \text{IHB}(L) \to 2^{\text{her}(L)}$, $i \to i \cap \text{her}(L)$, which associates its positive part with a Herbrand bivalued interpretation. The corresponding order on IHB(L) is $i \leqslant j$ iff $\text{pos}(i) \subseteq \text{pos}(j)$.

Remark 1.2. We may consider a three-valued Herbrand interpretation as a partial interpretation, assuming that if ato and $\neg$ato do not belong to i, then ato does not get a truth value belonging to $\{T, F\}$ but $\text{tv}_i(\text{ato}) = I$.

We now introduce the truth value of the connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$, and the quantifiers $\forall, \exists$, taking the truth value Indefinite into account. We will then be able to extend the monotony property to the formulas making the body of the rules of the studied programs. So, in the following definition, we give the truth tables of the connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$, and more generally the truth value of a formula of a logical language (using the connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$ and the quantifiers $\forall, \exists$).

Definition 1.3. We define the truth values of $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$ (considered as applications from $\{T, F, I\}^2$ (or $\{T, F, I\}$) to $\{T, F, I\}$). The truth tables of $\wedge$ and $\vee$ are the following:

$\vee$	T	F	I
T	T	T	T
F	T	F	I
I	T	I	I

$\wedge$	T	F	I
T	T	F	I
F	F	F	F
I	I	F	I

The truth value of $\neg$ is defined by $\neg T = F$, $\neg F = T$, $\neg I = I$, and the truth value of $P \Rightarrow Q$ is defined as usual by the truth value of $\neg P \vee Q$, and the truth value of $P \Leftrightarrow Q$ by the truth value of $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$.

(i) We then define the truth value tv, with respect to a three-valued i Herbrand interpretation of a closed formula f using the connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$, and the quantifiers $\forall, \exists$, inductively by the following:

- If $f = \text{ato} \in \text{her}(L)$, $\text{tv}_i(f) = T$ iff $\text{ato} \in i$, F iff $\neg\text{ato} \in i$, Indefinite otherwise.
- If $f = (t = u)$, (where t and u are two ground terms of the language) $\text{tv}_i(f) = T$ iff t and u are syntactically identical, False otherwise.
- If $f = \neg g$, or $(g \gamma h)$ where $\gamma \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$, the truth value of f is defined as usual with the truth tables above.
- If $f = \forall x\, g$, $\text{tv}_i(f) = T$ iff for every t element of $\text{UNI}(L)$, $\text{tv}_i(g(t)) = T$, F iff there is a t element of $\text{UNI}(L)$, $\text{tv}_i(g(t)) = F$, Indefinite otherwise.
- The same if $f = \exists x\, g$.

(ii) If f is not closed, $\text{tv}_i(f) = \text{tv}_i(\forall f)$ where $\forall f$ is the universal closure of f.

With this third truth value, we are now able to extend the monotony property to the formulas making the body of the rules of a three-valued program (a program having rules like lit $\leftarrow$ for, with the formula for using the symbols $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow, \forall, \exists$). If $\text{For}(L)$ is the set of the formulas of the language using the symbols $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow, \forall, \exists$, then we have the following proposition.

Proposition 1.4. *The maping* ϕ: $\text{IHT}(L) \to \{$*Functions from* $\text{For}(L)$ *to* $\{T, F, I\}\}$, *such that* $\phi(i) = \text{tv}_i$ *is monotonic with respect to the set inclusion order on* $\text{IHT}(L)$ *and the order induced by* $I \leqslant T$, $I \leqslant F$ *on the set of the functions from* $\text{For}(L)$ *to* $\{T, F, I\}$.

Proof. We have to show that if the formula for uses the symbols $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$, $i \subseteq j$ implies that $\text{tv}_i(\text{for}) \leqslant \text{tv}_j(\text{for})$. By induction on for. □

Remark 1.5. This monotony property means that if the formula for uses the symbols $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$, having the truth value of some of its atoms unknown (Indefinite), its truth value cannot change (from True to False or the converse) but only gets a well-known truth value (True or False).

We have extended the monotony property to For(L) by introducing a new truth value Indefinite. Since, even with this new truth value I, the formula $f \Rightarrow g$ gets the same truth value as the formula $\neg f \vee g$ in any Herbrand interpretation, and since we do not yet have the intersection models property with rules like lit $\Leftarrow$ for, we are now going to *extend the logical language L* by adding a new implication connective to it. It is neither Lukasiewicz nor Kleene's one, but is denoted by $\rightarrow$, in order to give another expressivity to the implication connective as we said in the introduction.

Definition 1.6. If f and g are two ground formulas, for any Herbrand interpretation i,

$$\text{tv}_i(f \rightarrow g) = F \text{ iff } \text{tv}_i(f) = T \text{ and } \text{tv}_i(g) \neq T$$
$$\text{and } \text{tv}_i(f \rightarrow g) = T \text{ otherwise.}$$

$f \leftarrow g$ is logically equivalent to $g \rightarrow f$.

Remark 1.7. This connective is called non-monotonic:

(a) considered as an application from $\{T, F, I\}^2$ to $\{T, F, I\}$, with the order induced by $I \leqslant F$, $I \leqslant T$, on these sets, we do not have $\rightarrow(I, F) \leqslant \rightarrow(T, F)$ although $(I, F) \leqslant (T, F)$, since $\rightarrow(I, F) = T$ while $\rightarrow(T, F) = F$.

(b) if we include in For(L), the formulas using the connective $\rightarrow$, the mapping $\phi : \text{IHT}(L) \rightarrow \{\text{Functions from For}(L) \text{ to } \{T, F, I\}\}$, such that $\phi(i) = \text{tv}_i$ is no longer monotonic.

One of the reasons for the choice of $\rightarrow$ is the completeness of the logic obtained. We have proved [24] that there is a formal system such that:

- if A is a formula of propositional calculus using the symbols $\neg$, $\rightarrow$, we have an equivalence between "A is a theorem of this formal system" and "A is a tautology";
- we have another completeness theorem in propositional calculus for any formula using the symbols $\neg$, $\rightarrow$, $\wedge$, and in predicate calculus for any formula using the symbols $\neg$, $\rightarrow$, and the existential quantifier $\exists$ as well.

The last completeness theorem enables us to ensure that the set of tautologies of this logic in predicate calculus is recursively enumerable. We have also proved that it is not recursive [24].

At this step of the extension of the truth values and the language, we are now able to get the model intersection property for a Pr program having rules like lit $\leftarrow$ for, if lit is a literal and the formula for uses the symbols $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, $\forall$, $\exists$. We first give the definitions of weak and strong three-valued Herbrand model of a program with negation in the following definition.

Definition 1.8. A Herbrand interpretation i of IHT(L) is a strong (resp. weak) model of a set Ax of formulas of a logical language L if for each formula for of Ax, we have $\mathrm{tv}_i(\mathrm{for}) = \mathrm{True}$ (resp. $\neq \mathrm{False}$). If a set Ax has a strong (resp. weak) model, then Ax is strongly (resp. weakly) consistent.

Remark 1.9. The empty set is not necessarily a weak model of any formula: for example,

$$(p(x_1, \ldots, x_n) \to q(x_1, \ldots, x_n)) \to p(x_1, \ldots, x_n).$$

A formula does not necessarily have a weak model: for example,

$$((p(x_1, \ldots, x_n) \wedge \neg p(x_1, \ldots, x_n)) \to q(x_1, \ldots, x_n))$$

$$\to (q(x_1, \ldots, x_n) \wedge \neg q(x_1, \ldots, x_n)).$$

In the following, we will only use the notion of strong model, to be called model, since it is actually the same notion as the notion of weak model for the programs we study.

We then have the following intersection model property.

Proposition 1.10. *An intersection of three-valued Herbrand models of a program with negation is still a three-valued Herbrand model of this program. Consequently, for a consistent program, its least Herbrand model denoted by* ltm(Pr) *is the intersection of all its Herbrand models.*

Proof. Pr is a set of rules like lit $\leftarrow$ for, with lit being a literal and the formula for using the symbols $\exists, \forall, \wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$. Let $(i_e)_{e \in E}$ be a family of Herbrand three-valued models of Pr. If $i = \bigcap_{e \in E} (i_e)$, we must show that, for every ground instance, lit $\leftarrow$ for of any rule of Pr, $\mathrm{tv}_i(\mathrm{lit} \leftarrow \mathrm{for}) = \mathrm{True}$. The only case to check is when $\mathrm{tv}_i(\mathrm{for}) = \mathrm{True}$ according to the truth table of $\to$. $\mathrm{tv}_i(\mathrm{for}) = \mathrm{True}$ implies $\mathrm{tv}_{i_e}(\mathrm{for}) = \mathrm{True}$, for each i_e, by the monotony property, since the formula for uses the symbols $\exists, \forall, \wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$ and $i \subseteq i_e$. As i_e is a model of Pr, $\mathrm{tv}_{i_e}(\mathrm{lit}) = \mathrm{True}$, for each i_e, that means lit $\in i_e$ for each i_e, because lit is a literal and it follows that lit $\in i = \bigcap_{e \in E} (i_e)$, and so $\mathrm{tv}_i(\mathrm{lit}) = \mathrm{True}$, which implies that $\mathrm{tv}_i(\mathrm{lit} \leftarrow \mathrm{for}) = \mathrm{True}$.

So the least Herbrand three-valued model of a consistent program with negation is the intersection of all its Herbrand models. □

We now want to characterize this least Herbrand model as a least fixpoint of an operator associated with a program with negation. For this, we carry on with the extension of the bivalued case by extending the consequence operator of Apt and Van Emden to a consequence operator for a three-valued program. Since negation is allowed also in the head of the rules of a program, programs may not be consistent.

For instance, the following Pr program does not have any three-valued model:

$$p(x_1, \ldots, x_n) \leftarrow q(x_1, \ldots, x_n)$$

$$\neg p(x_1, \ldots, x_n) \leftarrow q(x_1, \ldots, x_n)$$

$$q(x_1, \ldots, x_n).$$

This is why even if we allow negation in the head of the rules, programs with only positive rules, will play a peculiar part as in the theory of Fitting. To palliate the case of inconsistency of a Pr program, we add an element to the set $\mathrm{IHT}(L)$, which we call Contra in the following way.

Definition 1.11. The set $\mathrm{IHT}^{\infty}(L)$ denotes the set $\mathrm{IHT}(L) \cup \{\mathrm{Contra}\}$ and the order on $\mathrm{IHT}^{\infty}(L)$ is the order on $\mathrm{IHT}(L)$ extended by $i \subseteq \mathrm{Contra}$ for each i element of $\mathrm{IHT}(L)$.

We are now able to define the "consequence" operator which is associated with a Pr program with negation.

Definition 1.12. $T_{\mathrm{Pr}}: \mathrm{IHT}^{\infty}(L) \to \mathrm{IHT}^{\infty}(L)$,

$$T_{\mathrm{Pr}}(i) = \{\mathrm{lit} \in \mathrm{Her}(L) \mid \text{there is a ground instance lit} \leftarrow \text{for of a rule of Pr such that } \mathrm{tv}_i(\mathrm{for}) = \mathrm{True}\}$$

if this set does not include an atom and its negation;

$$T_{\mathrm{Pr}}(i) = \mathrm{Contra} \text{ otherwise;}$$

$$T_{\mathrm{Pr}}(\mathrm{Contra}) = \mathrm{Contra}.$$

This operator provides a denotational semantics of a Pr program as the following theorem states.

Theorem 1.13. (1) T_{Pr} *is monotonic and has a least fixpoint denoted by* $\mathrm{lfp}(T_{\mathrm{Pr}})$.

(2) Pr *is consistent* $\Leftrightarrow \mathrm{lfp}(T_{\mathrm{Pr}}) \neq \mathrm{Contra}$. *In this case*:

(a) *an interpretation* $i \in \mathrm{IHT}(L)$ *is a model of* $\mathrm{Pr} \Leftrightarrow T_{\mathrm{Pr}}(i) \subseteq i$.

(b) $\mathrm{lfp}(T_{\mathrm{Pr}}) = \mathrm{ltm}(\mathrm{Pr}) = \cap$ (*models of* Pr).

Proof. (1) If $i \in \mathrm{IHT}^{\infty}(L)$, $i \subseteq \mathrm{Contra}$ and $T_{\mathrm{Pr}}(i) \subseteq \mathrm{Contra} = T_{\mathrm{Pr}}(\mathrm{Contra})$. Let i and j be two elements of $\mathrm{IHT}(L)$. $i \subseteq j$ implies that $\mathrm{tv}_i(\mathrm{for}) \leqslant \mathrm{tv}_j(\mathrm{for})$ if the formula for uses only the symbols $\exists$, $\forall$, $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, which is the case for any formula making the body of a rule of Pr. T_{Pr} is thus monotonic. T_{Pr} has a least fixpoint because

$\mathrm{IHT}^{\infty}(L)$ satisfies the Birkhoff-Tarski conditions, since the empty set $\emptyset$ is its bottom element and any totally ordered set $\{I_\alpha, \alpha \in A\}$ of $\mathrm{IHT}^{\infty}(L)$ has a lub (lowest upper bound) which is Contra if there is $\alpha \in A$ such that $I_\alpha = \mathrm{Contra}$ and $\bigcup_{\alpha \in A} I_\alpha$, otherwise.

(2) To show that Pr is consistent if $\mathrm{lfp}(T_{\mathrm{Pr}}) \neq \mathrm{Contra}$, we show that:

(a) if $i \in \mathrm{IHT}(L)$ is such that $T_{\mathrm{Pr}}(i) \subseteq i$, then i is a model of Pr.

Let $\mathrm{lit} \leftarrow \mathrm{for}$ be a ground instance of a rule of Pr. Let us show that $\mathrm{tv}_i(\mathrm{lit} \leftarrow \mathrm{for}) = \mathrm{True}$, if $i \in \mathrm{IHT}(L)$ is such that $T_{\mathrm{Pr}}(i) \subseteq i$. The only case to check is when $\mathrm{tv}_i(\mathrm{for}) = \mathrm{True}$. As $i \in \mathrm{IHT}(L)$ and $T_{\mathrm{Pr}}(i) \subseteq i$, $T_{\mathrm{Pr}}(i) \neq \mathrm{Contra}$ and $\mathrm{lit} \in T_{\mathrm{Pr}}(i)$ by the definition of T_{Pr}. So $\mathrm{lit} \in i$ and $\mathrm{tv}_i(\mathrm{lit}) = \mathrm{True}$.

(b) If $i \in \mathrm{IHT}(L)$ is a model of Pr, then $T_{\mathrm{Pr}}(i) \subseteq i$.

We have $\mathrm{tv}_i(\mathrm{lit} \leftarrow \mathrm{for}) = \mathrm{True}$, for every ground instance of such a rule of Pr. If $\mathrm{lit} \in T_{\mathrm{Pr}}(i)$, then there is a ground instance $\mathrm{lit} \leftarrow \mathrm{for}$ of a rule of Pr such that $\mathrm{tv}_i(\mathrm{for}) = \mathrm{True}$. Then $\mathrm{tv}_i(\mathrm{lit}) = \mathrm{True}$ and $\mathrm{lit} \in i$.

(c) If $i \in \mathrm{IHT}(L)$ is a model of Pr, then $T_{\mathrm{Pr}} \uparrow \alpha \subseteq i$, for any ordinal α.

By transfinite induction, $\alpha = 0$, $T_{\mathrm{Pr}} \uparrow 0 = \emptyset$. If α is a successor ordinal, $T_{\mathrm{Pr}}(T_{\mathrm{Pr}} \uparrow \alpha - 1) \subseteq T_{\mathrm{Pr}}(i)$, by monotony of T_{Pr} and induction, and $T_{\mathrm{Pr}}(i) \subseteq i$ according to (b). If α is a limit ordinal, $T_{\mathrm{Pr}} \uparrow \alpha = \mathrm{lub}\{T_{\mathrm{Pr}} \uparrow \beta, \beta < \alpha\} \subseteq i$, by induction. As $\mathrm{lfp}(T_{\mathrm{Pr}}) = T_{\mathrm{Pr}} \uparrow \alpha$, for an ordinal α, $\mathrm{lfp}(T_{\mathrm{Pr}}) \subseteq i$, for any i model of Pr.

If Pr is consistent, there is $i \in \mathrm{IHT}(L)$ such that i is a model of Pr and $\mathrm{lfp}(T_{\mathrm{Pr}}) \subseteq i$ so $\mathrm{lfp}(T_{\mathrm{Pr}}) \neq \mathrm{Contra}$. Conversely, if $\mathrm{lfp}(T_{\mathrm{Pr}}) \neq \mathrm{Contra}$, $\mathrm{lfp}(T_{\mathrm{Pr}}) \in \mathrm{IHT}(L)$ and $T_{\mathrm{Pr}}(\mathrm{lfp}(T_{\mathrm{Pr}})) \subseteq \mathrm{lfp}(T_{\mathrm{Pr}})$ so $\mathrm{lfp}(T_{\mathrm{Pr}})$ is a model of Pr which is consistent. We have shown that $\mathrm{lfp}(T_{\mathrm{Pr}})$ is a model of Pr when it is $\neq \mathrm{Contra}$. As we have $\mathrm{lfp}(T_{\mathrm{Pr}}) \subseteq i$, for any i model of Pr, when Pr is consistent, $\mathrm{lfp}(T_{\mathrm{Pr}}) = \mathrm{ltm}(\mathrm{Pr}) = \cap$ (models of Pr), according to Proposition 1.4. □

Remark 1.14. We have also studied the notion of optimal model (optimal post-fixpoint) which provides a denotational semantics other than the least fixpoint. Since the set of three-value Herbrand interpretations is not a complete lattice, we do not have a biggest model but only several maximal ones. The intersection of all these models is called the optimal model, which coincides with the biggest consistent model. For more details, see [24].

In the following proposition, we know exactly the α ordinal such that $T_{\mathrm{Pr}} \uparrow \alpha = \mathrm{lfp}(T_{\mathrm{Pr}})$ in the case when Pr is a definite program having rules like $\mathrm{lit} \leftarrow \mathrm{lit}_1, \ldots, \mathrm{lit}_n$, where lit is a literal without the equality predicate and the lit_i are literals.

Proposition 1.15. (a) *If* Pr *is a three-valued definite program, then* T_{Pr} *is continuous.* T_{Pr} *is not continuous for every* Pr.

(b) *If* Pr *is a three-valued definite program, then* $\mathrm{ltm}(\mathrm{Pr}) = \mathrm{lfp}(T_{\mathrm{Pr}}) = T_{\mathrm{Pr}} \uparrow \omega$.

Proof. (a) Let X be a directed subset ($\forall x, y \in X, \exists z \in X$ such that $x \leqslant z$ and $y \leqslant z$) of IHT(L), where L is the language intended by Pr.

$\text{lit} \in T_{\text{Pr}}(\text{lub}(X)) \Leftrightarrow$ $\text{lit} \leftarrow \text{lit}_1 \wedge \text{lit}_2 \wedge \cdots \wedge \text{lit}_n$ is a ground instance of a rule of Pr such that $\text{tv}_{\text{lub}(X)}(\text{lit}_1 \wedge \text{lit}_2 \wedge \cdots \wedge \text{lit}_n) = \text{True}$ ($\text{lub}(X)$ is the least upper bound of X)

$\Leftrightarrow$ $\text{lit} \leftarrow \text{lit}_1 \wedge \text{lit}_2 \wedge \cdots \wedge \text{lit}_n$ is a ground instance of a rule of Pr such that $\forall i = 1, 2, \ldots, n$, $\text{lit}_i \in \text{lub}(X)$

$\Leftrightarrow$ $\text{lit} \leftarrow \text{lit}_1 \wedge \text{lit}_2 \wedge \cdots \wedge \text{lit}_n$ is a ground instance of a rule of Pr such that $\exists I \in X$ $\forall i = 1, 2, \ldots, n$, $\text{lit}_i \in I$ because X is a directed subset of IHT(L)

$\Leftrightarrow$ $\exists I \in X$ such that $\text{lit} \leftarrow \text{lit}_1 \wedge \text{lit}_2 \wedge \cdots \wedge \text{lit}_n$ is a ground instance of a rule of Pr such that $\text{tv}_I(\text{lit}_1 \wedge \text{lit}_2 \wedge \cdots \wedge \text{lit}_n) = \text{True}$

$\Leftrightarrow$ $\exists I \in X$ such that $\text{lit} \in T_{\text{Pr}}(I)$

$\Leftrightarrow$ $\text{lit} \in \text{lub}(T_{\text{Pr}}(X))$.

$T_{\text{Pr}}(\text{lub}(X)) = \text{Contra} \Leftrightarrow$ $\text{ato} \leftarrow \text{lit}_1 \wedge \text{lit}_2 \wedge \cdots \wedge \text{lit}_n$ and $\neg\text{ato} \leftarrow \text{lit}'_1 \wedge \cdots \wedge \text{lit}'_p$ there are two ground instances of a rule of Pr such that: $\forall i = 1, 2, \ldots, n$, $\text{tv}_{\text{lub}(X)}(\text{lit}_i) = \text{True}$ and $\forall j = 1, 2, \ldots, p$, $\text{tv}_{\text{lub}(X)}(\text{lit}'_j) = \text{True}$

$\Leftrightarrow$ $\exists I \in X$ such that $\text{ato} \in T_{\text{Pr}}(I)$ and $\neg\text{ato} \in T_{\text{Pr}}(I)$ since X is directed

$\Leftrightarrow$ $\exists I \in X$ such that $T_{\text{Pr}}(I) = \text{Contra}$

$\Leftrightarrow$ $\text{lub}(T_{\text{Pr}}(X)) = \text{Contra}$.

(b) follows immediately from (a). □

Remark 1.16. When Pr has some quantifier and some function symbol, T_{Pr} is not generally continuous. For instance let Pr be

$$P(a) \leftarrow \forall x Q(x)$$

$$Q(a) \leftarrow$$

$$Q(f(x)) \leftarrow Q(x).$$

Let

$$I_0 = \emptyset;\ I_1 = \{Q(a)\};\ I_2 = \{Q(a), Q(f(a))\}; \ldots;\ I_n = I_{n-1} \cup \{Q(f^{n-1}(a))\}.$$

We thus have an increasing sequence of three-valued Herbrand interpretations.

$$T_{\text{Pr}}\left(\bigcup_{n \geqslant 0} I_n\right) = \{P(a)\} \cup \bigcup_{n \geqslant 0} I_n$$

$$T_{\text{Pr}}(I_n) = I_{n+1}$$

$$\bigcup_{n \geqslant 0} T_{\text{Pr}}(I_n) = \bigcup_{n \geqslant 0} I_n.$$

If Pr is a program without any quantifier, then it can be transformed into a definite program as we will see in the following, and since we will have $T_{Pr} = T_{Pr'}$, T_{Pr} will be continuous.

If Pr is a program without any function symbol, then every increasing sequence of interpretations is stationary and there is no problem of continuity.

In the next section, we study the relations between our "consequence" operator, the Apt and Van Emden's one for programs without negation, and Fitting's operator for programs with only positive rules. To make this last link, we will need to introduce two notions of $\neg$-completion and $\rightarrow$-completion.

2. The relations between our operator, Apt and Van Emden and Fitting's operators

We recall here the definition of the "consequence" operator of Apt and Van Emden in our formalism.

Definition 2.1. If Pr is a bivalued program without negation, having rules like ato $\leftarrow$ for, where ato is an atom and for a formula using the symbols $\forall$, $\exists$, $\wedge$, $\vee$, then we associate the following operator with Pr: $B_{Pr}: \mathrm{IHB}(L) \rightarrow \mathrm{IHB}(L)$, defined by

$$B_{Pr}(i) = \mathrm{pos}^{-1}\{\mathrm{ato} \in \mathrm{her}(L) \mid \text{there is a ground instance ato} \leftarrow \mathrm{for} \text{ of a rule of Pr such that } \mathrm{tv}_i(\mathrm{for}) = \mathrm{True}\}.$$

Remark 2.2. We know that B_{Pr} is monotonic, for the order $i \leqslant j \Leftrightarrow \mathrm{pos}(i) \subseteq \mathrm{pos}(j)$. It follows that B_{Pr} has a least fixpoint which is also the least bivalued Herbrand model of Pr, since the models of Pr are exactly the post-fixpoints of B_{Pr} and the least fixpoint of a monotonic operator is its least post-fixpoint (by transfinite induction, $B_{Pr}\uparrow\alpha \subseteq i$, for every i model of Pr and every α ordinal).

The following proposition makes the link between our operator and the operator of Apt and Van Emden when Pr is a program having rules like ato $\leftarrow$ for, where the formula for uses the symbols $\forall$, $\exists$, $\wedge$, $\vee$.

Proposition 2.3

(a) $B_{Pr} = \mathrm{pos}^{-1} \circ T_{Pr} \circ \mathrm{pos}$

(b) $\mathrm{lfp}(B_{Pr}) = \mathrm{pos}^{-1}(\mathrm{lfp}(T_{Pr})) = \mathrm{lbm}(\mathrm{Pr}) = \mathrm{pos}^{-1}(\mathrm{ltm}(\mathrm{Pr}))$.

Proof. (a) Use the fact that if the formula for uses the symbols $\forall$, $\exists$, $\wedge$, $\vee$, then $\mathrm{tv}_i(\mathrm{for}) = \mathrm{True} \Leftrightarrow \mathrm{tv}_{\mathrm{pos}(i)}(\mathrm{for}) = \mathrm{True}$, by induction on the formula for.

(b) Show that $B_{Pr}\uparrow\alpha = \mathrm{pos}^{-1}(T_{Pr}\uparrow\alpha)$, by transfinite induction, with (a). □

We are now going to study the relations between our operator and Fitting's one for programs having positive rules. In order to make these relations easier, we need the two notions of $\neg$-completion and of $\rightarrow$-completion of a program. To make these

two completions, we introduce the normal form of a program. Finally to make the logical link between a program and its normal form, we need to introduce the notion of three-valued Herbrand tautology and equivalence.

Before that we carry on with the extension of the logical language. We can see that the formula $f \to g$ and the formula $\neg g \to \neg f$ do not have the same truth value in any Herbrand interpretation; for instance if i is an interpretation such that $\mathrm{tv}_i(f) = I$ and $\mathrm{tv}_i(g) = F$. This is why, instead of having only one connective equivalence associated with $\to$ as in the bivalued case ($\Leftrightarrow$), we have two equivalence connectives according to the following definition.

Definition 2.4. We define the two equivalence connective $\leftrightarrow$ and $\leftarrow\rightarrow$ by, if f and g are two ground formulas:

(1) $\mathrm{tv}_i(f \leftrightarrow g) = \mathrm{True}$ iff $\mathrm{tv}_i(f) = \mathrm{tv}_i(g)$, $\mathrm{tv}_i(f \leftrightarrow g) = \mathrm{False}$ otherwise.

(2) $\mathrm{tv}_i(f \leftarrow\rightarrow g) = \mathrm{True}$ iff $[\mathrm{tv}_i(f) = \mathrm{True} \Leftrightarrow \mathrm{tv}_i(g) = \mathrm{True}]$,
$\mathrm{tv}_i(f \leftarrow\rightarrow g) = \mathrm{False}$ otherwise.

Remark 2.5. The first formula $f \leftrightarrow g$ has the same truth value as the formula:

$$(f \to g) \wedge (g \to f) \wedge (\neg f \to \neg g) \wedge (\neg g \to \neg f),$$

in any Herbrand interpretation. In fact, the connective $\leftrightarrow$ is exactly the connective $\equiv$ of Fitting.

The second formula $f \leftarrow\rightarrow g$ has the same truth value as the formula:

$$(f \to g) \wedge (g \to f),$$

in any Herbrand interpretation.

Definition 2.6. (a) A formula f of a logical language L is a Herbrand three-valued tautology if it is True in any Herbrand interpretation, which means that $\mathrm{tv}_i(\forall f) = \mathrm{True}$ for every Herbrand interpretation i. We denote the fact that f is a Herbrand tautology by $\models_{\mathrm{TH}} f$.

(b) Two formulas f and g of a logical language containing $\leftrightarrow$ are logically Herbrand equivalent iff $f \leftrightarrow g$ is a Herbrand three-valued tautology. We denote the fact that f and g are logically Herbrand equivalent by $f \equiv_{\mathrm{TH}} g$.

Remark 2.7. To say that $\models_{\mathrm{TH}} f \leftrightarrow g$ is stronger than saying that f and g have the same truth value in any Herbrand interpretation. $\models_{\mathrm{TH}} f \leftrightarrow g$ implies that f and g have the same truth value in any Herbrand interpretation but the converse is false except when f and g are closed formulas of the language.

We could have defined these notions more generally, for every three-valued interpretation, for the general case, see [24]. The following examples, which we need for making the logical link between a program and its normal form, are true in every three-valued interpretation.

2.1. Examples of logical three-valued equivalences

$A \Rightarrow B \equiv_{TH} \neg B \Rightarrow \neg A$,

$A \Rightarrow B \equiv_{TH} \neg A \vee B$,

$A \wedge (B \vee C) \equiv_{TH} (A \wedge B) \vee (A \wedge C)$,

$A \vee (B \vee C) \equiv_{TH} (A \vee B) \vee C$,

$\neg(A \vee B) \equiv_{TH} \neg A \wedge \neg B$,

$(A \vee B) \to C \equiv_{TH} (A \to C) \wedge (B \to C)$,

$A \to (B \wedge C) \equiv_{TH} (A \to B) \wedge (A \to C)$,

$\forall x\,(P \wedge Q) \equiv_{TH} (\forall x\,P) \wedge (\forall x\,Q)$,

$\neg(\forall x\,P) \equiv_{TH} (\exists x\,\neg P)$,

$A \Leftrightarrow B \equiv_{TH} (A \Rightarrow B) \wedge (B \Rightarrow A)$,

$A \vee (B \wedge C) \equiv_{TH} (A \vee B) \wedge (A \vee C)$,

$\neg\neg A \equiv_{TH} A$,

$A \wedge (B \wedge C) \equiv_{TH} (A \wedge B) \wedge C$,

$\neg(A \wedge B) \equiv_{TH} \neg A \vee \neg B$,

$(A \wedge B) \to C \equiv_{TH} A \to (B \to C)$,

$A \to (B \vee C) \equiv_{TH} (A \to B) \vee (A \to C)$,

$\exists x\,(P \vee Q) \equiv_{TH} (\exists x\,P) \vee (\exists x\,Q)$,

$\neg(\exists x\,P) \equiv_{TH} (\forall x\,\neg P)$,

$\forall x\,(P \to Q) \equiv_{TH} (\exists x\,P) \to Q$ if x is not a free variable of Q,

$\exists x\,(P \to Q) \equiv_{TH} (\forall x\,P) \to Q$ if x is not a free variable of Q.

Remark 2.8. In propositional calculus, we have studied more generally the expressivity of the connectives in three-valued logic. This can be helpful to conceive languages based on rules for expert systems which contain other connectives than $\to$, $\neg$, $\vee$. For this, see [24].

We need the notion of the universal closure of a set of formulas of the language for the following.

Definition 2.9. If Ax is a set of formulas of a logical language L, then the formula $\forall$Ax denotes the conjunction of all the universal closures of the formulas of Ax.

Before defining the normal form of a program, we can associate a definite program with a program without any quantifier, according to the following proposition.

Proposition 2.10. *If* Pr *is a three-valued program without any quantifier, having rules like* lit $\leftarrow$ for *where the formula* for *uses the symbols* $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, *there is a definite* Pr′ *program such that* $\forall \mathrm{Pr} \equiv_{TH} \forall \mathrm{Pr}'$.

Proof. This uses the two-valued classical transformations. If any formula for uses the symbols $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, then it is logically equivalent to a formula in disjunctive normal form (a disjunction of conjunctions of literals). Pr is a set of rules like lit $\leftarrow$ for, and we then use the fact that $(A \vee B) \to C \equiv_{TH} (A \to B) \wedge (A \to C)$ and $\forall x\,(A(x) \wedge B(x)) \equiv_{TH} \forall x\,A(x) \wedge \forall x\,B(x)$. □

Remark 2.11. This logical equivalence is not only true in a Herbrand interpretation, but in every three-valued interpretation. This proposition solves the problem of continuity for a program without any quantifier because it can easily be proved that, if Pr′ is the definite program associated with Pr, $T_{\mathrm{Pr}} = T_{\mathrm{Pr}'}$.

We are now going to define the normal form of a pack of rules having their head built on the same predicate symbol in order to complete a program with negation in two ways.

Definition 2.12. Let $r_1, r_2, \ldots, r_m$ be m positive rules with their head literals built on the same predicate symbol p; $r_i : p(t_{i,1}, t_{i,2}, \ldots, t_{i,n}) \leftarrow \text{for}_i$; for $i = 1, \ldots, m$. The normal form of the pack $(r_1, r_2, \ldots, r_m)$ is the rule

$$p(x_1, x_2, \ldots, x_n) \leftarrow \text{for}'_1 \vee \cdots \vee \text{for}'_m$$

where for'_i denotes the ground formula

$$\exists y_1 \cdots \exists y_p \quad (x_1 = t_{i,1} \wedge \cdots \wedge x_n = t_{i,n} \wedge \text{for}_i)$$

where $y_1, \ldots, y_p$ are the free variables of for_i, $t_{i,1}, t_{i,2}, \ldots, t_{i,n}$ and $x_1, \ldots, x_n$ are some variables different from the y_j.

We define the normal form of a pack of rules with a negative literal in their heads built on the same symbol of predicate p in the same way.

We can give the logical relation between a program and its normal form with the following proposition.

Proposition 2.13. *If r is the rule obtained from the normal form of the pack $(r_1, r_2, \ldots, r_m)$, we have*

$$\forall r \equiv_{\text{TH}} \forall r_1 \wedge \cdots \wedge \forall r_m.$$

Proof. Let r_i be $p(t_{i,1}, t_{i,2}, \ldots, t_{i,n}) \leftarrow \text{for}_i$ with $y_1, \ldots, y_p$ the free variables of $\text{for}_i, t_{i,1}, t_{i,2}, \ldots, t_{i,n}$. Since for'_i does not have any free variable, we have

$$\forall r \equiv_{\text{TH}} \forall x_1 \forall x_2 \cdots \forall x_n \, (p(x_1, x_2, \ldots, x_n) \leftarrow \text{for}'_1 \vee \cdots \vee \text{for}'_m).$$

Since $(A_1 \vee A_2 \vee \cdots \vee A_n) \rightarrow B \equiv_{\text{TH}} (A_1 \rightarrow B) \wedge \cdots \wedge (A_n \rightarrow B)$ and $\forall x \, (A(x) \wedge B(x)) \equiv_{\text{TH}} (\forall x \, A(x) \wedge \forall x \, B(x))$ we have

$$\forall x_1 \forall x_2 \cdots \forall x_n \, (p(x_1, x_2, \ldots, x_n) \leftarrow \text{for}'_1 \vee \cdots \vee \text{for}'_m) \equiv_{\text{TH}}$$

$$\forall x_1 \forall x_2 \cdots \forall x_n \, (p(x_1, x_2, \ldots, x_n) \leftarrow \text{for}'_1) \wedge \cdots$$

$$\wedge \forall x_1 \forall x_2 \cdots \forall x_n \, (p(x_1, x_2, \ldots, x_n) \leftarrow \text{for}'_m).$$

By using the fact that $\exists x \, A(x) \rightarrow B \equiv_{\text{TH}} \forall x \, (A(x) \rightarrow B)$ if x is not a free variable of B, we have

$$\forall x_1 \cdots \forall x_n \quad (p(x_1, \ldots, x_n) \leftarrow \text{for}'_i) \equiv_{\text{TH}}$$

$$\forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_p \, (p(x_1, \ldots, x_n) \leftarrow (x_1 = t_{i,1} \wedge \cdots \wedge x_n = t_{i,n} \wedge \text{for}_i)).$$

At this step of the demonstration, all the logical equivalences used are True in every three-valued interpretation and $\equiv$ could be used instead of $\equiv_{\text{TH}}$. From now on the logical equivalences used in the rest of the proof become true only in the interpreta-

tions where the equality predicate is interpreted by syntactical equality. In such an interpretation and particularly in any Herbrand interpretation, we have

$$\forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_p \, (p(x_1, \ldots, x_n) \leftarrow (x_1 = t_{i,1} \wedge \cdots \wedge x_n = t_{i,n} \wedge \text{for}_i)) \equiv_{\text{TH}}$$
$$\forall y_1 \cdots \forall y_p (p(t_{i,1}, \ldots, t_{i,n}) \leftarrow \text{for}_i). \tag{1}$$

As the formulas

$$f_1 = \forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_p \, (p(x_1, \ldots, x_n) \leftarrow (x_1 = t_{i,1} \wedge \cdots \wedge x_n = t_{i,n} \wedge \text{for}_i))$$

and

$$f_2 = \forall y_1 \cdots \forall y_p \, (p(t_{i,1}, \ldots, t_{i,n}) \leftarrow \text{for}_i)$$

are closed, to show (1) we only have to show that f_1 and f_2 have the same truth value in any Herbrand interpretation (which would be more generally true in an interpretation where the equality predicate is interpreted by syntactical equality). As the formulas f_1 and f_2 never take the truth value Indefinite (because the implication connective $\rightarrow$ never takes the truth value Indefinite), we only have to show that $\text{tv}_i(f_1) = \text{True}$ iff $\text{tv}_i(f_2) = \text{True}$.

$\text{tv}_i(f_1) = \text{True}$ iff

$\forall t_1 \cdots \forall t_n \forall u_1 \cdots \forall u_p \in \text{UNI}(L)^{n+p}$
$[\forall j \in \{1, \ldots, n\}, t_j = t_{i,j}(u_1, \ldots, u_p)$ and
$\text{tv}_i(\text{for}_i(u_1, \ldots, u_p)) = \text{True}$
implies $\text{tv}_i(p(t_1, \ldots, t_n)) = \text{True}]$.

$\text{tv}_i(f_2) = \text{True}$ iff

$\forall u_1 \cdots \forall u_p \in \text{UNI}(L)^p \, [\text{tv}_i(\text{for}_i(u_1, \ldots, u_p)) = \text{True}$ implies
$\text{tv}_i(p(t_{i,1}(u_1, \ldots, u_p), \ldots, t_{i,n}(u_1, \ldots, u_p)))$
$= \text{True}]$.

It is then clear that $\text{tv}_i(f_1) = \text{True}$ iff $\text{tv}_i(f_2) = \text{True}$. □

Proposition 2.14. *If* Pr *is a three-valued program and* Pr′ *is its normal form, we have* $\forall \text{Pr} \equiv_{\text{TH}} \forall \text{Pr}'$.

Proof. Easy with Proposition 2.13. □

We are now able to define the two kinds of completion of a program.

2.2. Completions of rules and programs

Definition 2.15. Let $(r_1, r_2, \ldots, r_m)$ be a pack of rules with, in their head, a positive literal built on the same predicate symbol p. The $\neg$-completion rule of the pack

$(r_1, r_2, \ldots, r_m)$ is the rule

$$\neg\text{lit} \leftarrow \neg\text{for}$$

where lit$\leftarrow$for is the normal form of the pack $(r_1, r_2, \ldots, r_m)$.

The $\rightarrow$-completion rule of the pack $(r_1, r_2, \ldots, r_m)$ is the rule:

$$\text{lit} \rightarrow \text{for}$$

where lit$\leftarrow$for is the normal form of the pack $(r_1, r_2, \ldots, r_m)$. This is not a three-valued rule in general.

We do the same for the packs of negative rules by using the fact that $\neg\neg\text{ato} \equiv_{TH} \text{ato}$.

Definition 2.16. The $\neg$-completion of a three-valued Pr program with only positive rules is the program obtained from Pr by first replacing all the packs of rules by their normal forms and adding the $\neg$-completion of these packs and secondly by adding to the Pr program $\neg p(x_1, \ldots, x_n)$ where p is a predicate symbol of Pr without being in any head of its rules. We denote this new program by

$$\text{Comp}(\neg, \text{Pr}).$$

The same for the $\rightarrow$-completion of Pr, denoted by

$$\text{Comp}(\rightarrow, \text{Pr}).$$

Example 2.17. We give an example of $\neg$-completion which allows us to define the predicate odd (by the negation of the predicate even); but as we allow the negation in the programs, we could have defined it without this mechanism. Let Pr be the program

$$r_1 : \text{even}(\text{ze})$$

$$r_2 : \text{even}(\text{fo}(\text{fo}(x))) \leftarrow \text{even}(x).$$

The normal form of r_1 and r_2 is r: $\text{even}(y) \leftarrow \exists x\ (y = \text{ze} \vee (y = \text{fo}(\text{fo}(x)) \wedge \text{even}(x)))$. The $\neg$-completion of Pr is the program obtain by adding to r the rule

$$\neg\text{even}(y) \leftarrow \forall x \quad (\neg(y = \text{ze}) \wedge (\neg(y = \text{fo}(\text{fo}(\text{x}))) \vee \neg\text{even}(x))).$$

We could have defined the $\neg$-even predicate without using the mechanism of completion by

$$\neg\text{even}(\text{fo}(y)) \leftarrow \text{even}(y).$$

Remark 2.18. If Pr is a bivalued program with the rules ato$\leftarrow$for, with the formula for using the symbols $\forall, \exists, \wedge, \vee$, then we have only one completion because $\forall\text{Comp}(\neg, \text{Pr}) \equiv_B \forall\text{Comp}(\Rightarrow, \text{Pr})$, since we have $\neg B \Rightarrow \neg A \equiv_B A \Rightarrow B$ ($f \equiv_B g$ means that the formula $f \Leftrightarrow g$ is a bivalued tautology, which means True in every bivalued interpretation).

The completion of a program Pr having positive rules in the way of Fitting is the $\rightarrow$-completed of the $\neg$-completed of our program. Since the formula $A \cong B$ is logically equivalent to $(A \rightarrow B) \wedge (B \rightarrow A) \wedge (\neg A \rightarrow \neg B) \wedge (\neg B \rightarrow \neg A)$, and Fitting completes his program by putting the pack of rules built on the same predicate p in their heads in normal form. The normal form of the pack $(r_1, r_2, \ldots, r_m)$ built on the predicate p is the rule

$$p(x_1, \ldots, x_n) \cong \text{for}'_1 \vee \cdots \vee \text{for}'_m$$

where for'_i denotes the ground formula

$$\exists y_1 \ldots \exists y_p \quad (x_1 = t_{i,1} \wedge \cdots \wedge x_n = t_{i,n} \wedge \text{for}_i)$$

and r_i is the rule $p(t_{i,1}, \ldots, t_{i,n}) \leftarrow \text{for}_i$; $i = 1, \ldots, m$ and $y_1, \ldots, y_p$ are the free variables of for_i, $t_{i,1}, \ldots, t_{i,n}$ and $x_1, \ldots, x_n$ are variables different from the y_j.

We are now able to make the relation between our "consequence" operator and Fitting's operator. To make this relation, we first give the definition of the operator of Fitting in our formalism:

Definition 2.19. The following operator introduced by Fitting [7] is defined only for programs with positive rules and denoted F_{Pr}. $F_{\text{Pr}}: \text{IHT}(L) \rightarrow \text{IHT}(L)$, is such that:

$$\begin{aligned} F_{\text{Pr}}(i) = \{&\text{ato} \mid \text{there is a ground instance ato} \leftarrow \text{for} \\ &\text{of a rule of Pr such that } \text{tv}_i(\text{for}) = \text{True}\} \\ \cup \{&\neg\text{ato} \mid \text{for every ground instance ato} \leftarrow \text{for of Pr, } \text{tv}_i(\text{for}) = \text{False}\}. \end{aligned}$$

The following proposition gives the relations between our operator and Fitting's one.

Proposition 2.20. (1) $T_{\text{Comp}(\neg,\text{Pr})} = F_{\text{Pr}}$.

(2) (a) *A three-valued Herbrand interpretation i is a model of* $\text{Comp}(\neg, \text{Pr})$ *iff* $F_{\text{Pr}}(\text{i}) \subseteq i$.

(b) *A three-valued Herbrand interpretation i is a model of* $\text{Comp}(\rightarrow, \text{Comp}(\neg, \text{Pr}))$ *iff* $F_{\text{Pr}}(i) = i$.

(3) $\text{lfp}(F_{\text{Pr}}) = \text{ltm}(\text{Comp}(\rightarrow, \text{Comp}(\neg, \text{Pr}))) = \text{ltm}(\text{Comp}(\neg, \text{Pr}))$.

Proof. (1) $T_{\text{Comp}(\neg,\text{Pr})} = F_{\text{Pr}}$. We first show that *if* Pr′ *is the normal form of* Pr, *then we have* $T_{\text{Pr}} = T_{\text{Pr}'}$. Let Pr be a three-valued program. If Pr′ is its normal form, then $\forall\text{Pr} \equiv_{\text{TH}} \forall\text{Pr}'$ (Proposition 2.14). Let $(\alpha_1, \ldots, \alpha_n) \in \text{UNI}(L)^n$ such that $\text{lit}(\alpha_1, \ldots, \alpha_n) \in T_{\text{Pr}}(i)$. Then there is $(t_1, \ldots, t_p) \in \text{UNI}(L)^p$ such that

$$\text{lit}(\alpha_1, \ldots, \alpha_n) \leftarrow \text{for}_i(t_1, \ldots, t_p)$$

is a ground instance of Pr with $\mathrm{tv}_i(\mathrm{for}_i(t_1, \ldots, t_p)) = \mathrm{True}$. This ground instance comes from a rule of Pr like

$$\mathrm{lit}(t_{i,1}(y_1, \ldots, y_p), \ldots, t_{i,n}(y_1, \ldots, y_p)) \leftarrow \mathrm{for}_i(y_1, \ldots, y_p)$$

with $\alpha_j = t_{i,j}(t_1, \ldots, t_p)$.

So

$$\mathrm{lit}(x_1, \ldots, x_n) \leftarrow \bigvee_i (\exists y_1 \cdots \exists y_p (x_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge x_n = t_{i,n}(y_1, \ldots, y_p) \wedge \mathrm{for}_i(y_1, \ldots, y_p)))$$

is a rule of Pr′, normal form of Pr and there is a ground instance of this rule

$$\mathrm{lit}(\alpha_1, \ldots, \alpha_n) \leftarrow \bigvee_i (\exists y_1 \cdots \exists y_p (\alpha_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge \alpha_n = t_{i,n}(y_1, \ldots, y_p) \wedge \mathrm{for}_i(y_1, \ldots, y_p)))$$

such that

$$\mathrm{tv}_i(\exists y_1 \cdots \exists y_p (\alpha_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge \alpha_n = t_{i,n}(y_1, \ldots, y_p) \wedge \mathrm{for}_i(y_1, \ldots, y_p))) = \mathrm{True}$$

since there is $(t_1, \ldots, t_p) \in \mathrm{UNI}(L)^p$ such that $\alpha_j = t_{i,j}(t_1, \ldots, t_p)$ and $\mathrm{tv}_i(\mathrm{for}_i(t_1, \ldots, t_p)) = \mathrm{True}$. So $\mathrm{lit}(\alpha_1, \ldots, \alpha_n) \in T_{\mathrm{Pr}'}(i)$.

In the same way, $\mathrm{lit}(\alpha_1, \ldots, \alpha_n) \in T_{\mathrm{Pr}'}(i)$ implies that $\mathrm{lit}(\alpha_1, \ldots, \alpha_n) \in T_{\mathrm{Pr}}(i)$.

$$\mathrm{lit}(\alpha_1, \ldots, \alpha_n) \in T_{\mathrm{Pr}}(i) \Leftrightarrow \mathrm{lit}(\alpha_1, \ldots, \alpha_n) \in T_{\mathrm{Pr}'}(i). \qquad (*)$$

So $T_{\mathrm{Pr}}(i) = T_{\mathrm{Pr}'}(i)$, since the case $T_{\mathrm{Pr}}(i) = \mathrm{Contra}$ comes from (*). So $T_{\mathrm{Pr}} = T_{\mathrm{Pr}'}$.

Secondly, we prove that $T_{\mathrm{Comp}(\neg, \mathrm{Pr})}(i) \neq \mathrm{Contra}$ *if* $i \in \mathrm{IHT}(L)$. (That comes from the fact that there is only one rule with a given predicate symbol in its head in the program $\mathrm{Comp}(\neg, \mathrm{Pr})$). $\mathrm{Comp}(\neg, \mathrm{Pr})$ is obtained by adding to the rules of the normal form of Pr like

$$\mathrm{ato}(x_1, \ldots, x_n) \leftarrow \bigvee_i (\exists y_1 \cdots \exists y_p (x_1 = t_{i,1} \wedge \cdots \wedge x_n = t_{i,n} \wedge \mathrm{for}_i))$$

the rules

$$\neg\mathrm{ato}(x_1, \ldots, x_n) \leftarrow \neg\left(\bigvee_i (\exists y_1 \cdots \exists y_p (x_1 = t_{i,1} \wedge \cdots \wedge x_n = t_{i,n} \wedge \mathrm{for}_i))\right).$$

If there is $(\alpha_1, \ldots, \alpha_n) \in \mathrm{UNI}(L)^n$ such that $\mathrm{ato}(\alpha_1, \ldots, \alpha_n) \in T_{\mathrm{Comp}(\neg, \mathrm{Pr})}(i)$ and $\neg\mathrm{ato}(\alpha_1, \ldots, \alpha_n) \in T_{\mathrm{Comp}(\neg, \mathrm{Pr})}(i)$, then, if

$$\mathrm{for}' = \bigvee_i (\exists y_1 \cdots \exists y_p (\alpha_1 = t_{i,1} \wedge \cdots \wedge \alpha_n = t_{i,n} \wedge \mathrm{for}_i)),$$

we both have $\mathrm{tv}_i(\mathrm{for}') = \mathrm{True}$ and False which is impossible.

For the literals $\neg p(x_1, \ldots, x_n)$ where p is a predicate symbol of Pr without being in any head of its rules, we cannot have both a ground instance of $p(x_1, \ldots, x_n)$ and its negation in $T_{\text{Comp}(\neg,\text{Pr})}(i)$ since p does not appear in the head of a rule of Pr.

Finally, we are now able to prove that $T_{\text{Comp}(\neg,\text{Pr})}(i) = F_{\text{Pr}}(i)$.

$T_{\text{Comp}(\neg,\text{Pr})}(i) = \{\text{lit} \in \text{Her}(L) \mid \text{there is a ground instance lit} \leftarrow \text{for of Comp}(\neg, \text{Pr}) \text{ such that } \text{tv}_i(\text{for}) = \text{True}\}$ if this set does not contain an atom and its negation, $T_{\text{Comp}(\neg,\text{Pr})}(i) = \text{Contra}$ otherwise.

$F_{\text{Pr}}(i) = \{\text{ato} \in \text{her}(L) \mid \text{there is a ground instance ato} \leftarrow \text{for of Pr such that } \text{tv}_i(\text{for}) = \text{True}\} \cup \{\neg\text{ato} \in \text{her}(L) \mid \text{for each ground instance ato} \leftarrow \text{for of Pr, } \text{tv}_i(\text{for}) = \text{False}\} = T_{\text{Pr}}(i) \cup \{\neg\text{ato} \in \neg\text{her}(L) \mid \text{for each ground instance ato} \leftarrow \text{for of Pr, } \text{tv}_i(\text{for}) = \text{False}\}$ because Pr has only positive rules. Let $\text{lit} \in T_{\text{Comp}(\neg,\text{Pr})}(i)$. Either $\text{lit} = \text{ato}$ and $\text{lit} \in T_{\text{Pr}'}(i) = T_{\text{Pr}}(i)$, so $\text{lit} \in F_{\text{Pr}}(i)$; or $\text{lit} = \neg\text{ato}(\alpha_1, \ldots, \alpha_n) \in T_{\text{Comp}(\neg,\text{Pr})}(i)$.

- Either for some $(\alpha_1, \ldots, \alpha_n) \in \text{UNI}(L)^n$ and some ground instance of Comp(¬, Pr):

  $$\neg\text{ato}(\alpha_1, \ldots, \alpha_n) \leftarrow \neg\text{for}'$$

  such that $\text{tv}_i(\text{for}') = \text{False}$, with $\text{for}' = \bigvee_i (\exists y_1 \cdots \exists y_p(\alpha_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge \alpha_n = t_{i,n}(y_1, \ldots, y_p) \wedge \text{for}_i(y_1, \ldots, y_p)))$. So $\forall i = 1, \ldots, m$ and $\forall (u_1, \ldots, u_p) \in \text{UNI}(L)^p$, if $\alpha_j = t_{i,j}(u_1, \ldots, u_p)$, for every $j = 1, \ldots, n$, then $\text{tv}_i(\text{for}_i(u_1, \ldots, u_p)) = \text{false}$. Let $\text{ato}(\alpha_1, \ldots, \alpha_n)$ be the head of a ground instance of a rule of Pr; it comes from

  $$\text{ato}(t_{i,1}(y_1, \ldots, y_p), \ldots, t_{i,n}(y_1, \ldots, y_p)) \leftarrow \text{for}_i(y_1, \ldots, y_p).$$

  Let $(u_1, \ldots, u_p) \in \text{UNI}(L)^p$ be such that $t_{i,j}(u_1, \ldots, u_p) = \alpha_j$, for every $j = 1, \ldots, n$. Then $\text{tv}_i(\text{for}_i(u_1, \ldots, u_p)) = \text{False}$, since $\text{tv}_i(\text{for}') = \text{False}$; so $\text{lit} = \neg\text{ato}(\alpha_1, \ldots, \alpha_n) \in F_{\text{Pr}}(i)$.
- Or $\neg\text{ato}(\alpha_1, \ldots, \alpha_n) \in T_{\text{Comp}(\neg,\text{Pr})}(i)$, because ato is a predicate symbol of Pr without being in any head of its rules. Then $\neg\text{ato}(\alpha_1, \ldots, \alpha_n) \in \{\neg\text{ato} \in \neg\text{her}(L) \mid \text{for each ground instance ato} \leftarrow \text{for of Pr, } \text{tv}_i(\text{for}) = \text{False}\}$ since there is no ground instance $\text{ato} \leftarrow \text{for}$ of Pr, and $\neg\text{ato}(\alpha_1, \ldots, \alpha_n) \in F_{\text{Pr}}(i)$.

Conversely, if $\text{lit} \in F_{\text{Pr}}(i)$, then either $\text{lit} \in T_{\text{Pr}}(i) = T_{\text{Pr}'}(i) \subseteq T_{\text{Comp}(\neg,\text{Pr})}(i)$, or $\text{lit} = \neg\text{ato}(\alpha_1, \ldots, \alpha_n)$ for some $(\alpha_1, \ldots, \alpha_n) \in \text{UNI}(L)^n$; then for every ground instance

$$\text{ato}(t_{i,1}(u_1, \ldots, u_p), \ldots, t_{i,n}(u_1, \ldots, u_p)) \leftarrow \text{for}_i(u_1, \ldots, u_p)$$

such that $t_{i,j}(u_1, \ldots, u_p) = \alpha_j$, for every $j = 1, \ldots, n$, we have $\text{tv}_i(\text{for}_i(u_1, \ldots, u_p)) = \text{False}$; so $\text{tv}_i(\bigvee_i (\exists y_1 \cdots \exists y_p(\alpha_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge \alpha_n = t_{i,n}(y_1, \ldots, y_p) \wedge \text{for}_i(y_1, \ldots, y_p)))) = \text{False}$ and $\neg\text{ato}(\alpha_1, \ldots, \alpha_n) \in T_{\text{Comp}(\neg,\text{Pr})}(i)$. If ato is not in any head of a rule of Pr, then $\neg\text{ato}(x_1, \ldots, x_n) \in \text{Comp}(\neg, \text{Pr})$ and every ground instance of $\neg\text{ato}(x_1, \ldots, x_n)$ belongs to $\{\neg\text{ato} \in \neg\text{her}(L) \mid \text{for each ground instance ato} \leftarrow \text{for of Pr, } \text{tv}_i(\text{for}) = \text{False}\} \subseteq F_{\text{Pr}}(i)$ and to $T_{\text{Comp}(\neg,\text{Pr})}(i)$.

We then get the result (2a) by Theorem 1.13(2a), because $i \in \text{IHT}(L)$ is a model of Pr iff $T_{\text{Pr}}(i) \subseteq i$. As $T_{\text{Comp}(\neg,\text{Pr})} = F_{\text{Pr}}$, we have that a three-valued Herbrand interpretation i is a model of $\text{Comp}(\neg, \text{Pr}) \Leftrightarrow T_{\text{Comp}(\neg,\text{Pr})}(i) \subseteq i \Leftrightarrow F_{\text{Pr}}(i) \subseteq i$.

Result (2b) may be proved in two ways:

- The first way using Fitting's argument saying that i is a model of Pr (which means a model of its completion) iff i is a fixpoint of F_{Pr} and using that the Fitting's completion of a Pr program is the $\rightarrow$-completion of the $\neg$-completion of a program in our way, we then obtain that a three-valued Herbrand interpretation i is a model of $\mathrm{Comp}(\rightarrow, \mathrm{Comp}(\neg, \mathrm{Pr})) \Leftrightarrow F_{\mathrm{Pr}}(i) = i$.
- The second way will be seen as a result of the following Proposition 2.21.

(3) $\mathrm{lfp}(F_{\mathrm{Pr}}) = \mathrm{ltm}(\mathrm{Comp}(\rightarrow, \mathrm{Comp}(\neg, \mathrm{Pr})))$ is a consequence of (2b). $\mathrm{lfp}(F_{\mathrm{Pr}}) = \mathrm{ltm}(\mathrm{Comp}(\neg, \mathrm{Pr}))$ is a consequence of (2a) and of the fact that $F_{\mathrm{Pr}} \uparrow \alpha \subseteq i$ for each model i of $\mathrm{Comp}(\neg, \mathrm{Pr})$ for each ordinal α, by transfinite induction. □

We end this section by a proposition which enables us to prove the result (2a) of the previous proposition.

Proposition 2.21. *If* Pr *is consistent, then* $\mathrm{Comp}(\rightarrow, \mathrm{Pr})$ *is consistent and we have*

(1) *An interpretation* $i \in \mathrm{IHT}(L)$ *is a three-valued model of* $\mathrm{Comp}(\rightarrow, \mathrm{Pr}) \Leftrightarrow T_{\mathrm{Pr}}(i) = i$.

(2) $\mathrm{ltm}(\mathrm{Pr}) = \mathrm{ltm}(\mathrm{Comp}(\rightarrow, \mathrm{Pr})) = \mathrm{lfp}(T_{\mathrm{Pr}})$.

Proof. We first remark that (2) is partly a consequence of (1): $\mathrm{ltm}(\mathrm{Comp}(\rightarrow, \mathrm{Pr}) = \mathrm{lfp}(T_{\mathrm{Pr}})$ and secondly a consequence of Theorem 1.13: $\mathrm{lfp}(T_{\mathrm{Pr}}) = \mathrm{ltm}(\mathrm{Pr})$. We also remark that (2b) of Proposition 2.20 may be proved with (1) of this proposition:

$$F_{\mathrm{Pr}}(i) = i \Leftrightarrow T_{\mathrm{comp}(\neg, \mathrm{Pr})}(i) = i \quad \text{(Proposition 2.20(1))}$$

$$\Leftrightarrow i \text{ is a model of } \mathrm{Comp}(\rightarrow, \mathrm{Comp}(\neg, \mathrm{Pr}))$$

(by (1) of the Proposition 2.21).

We just have to show (1):

$$i \neq \mathrm{Contra} \text{ and } T_{\mathrm{Pr}}(i) = i \Leftrightarrow i \text{ is a model of } \mathrm{Comp}(\rightarrow, \mathrm{Pr}).$$

That will also prove that if Pr is consistent, then $\mathrm{Comp}(\rightarrow, \mathrm{Pr})$ is also consistent since if Pr is consistent then $\mathrm{lfp}(T_{\mathrm{Pr}}) \neq \mathrm{Contra}$ according to Proposition 1.4 and $\mathrm{lfp}(T_{\mathrm{Pr}})$ is then also a model of $\mathrm{Comp}(\rightarrow, \mathrm{Pr})$ too.

(i) $T_{\mathrm{Pr}}(i) = i \Rightarrow i$ is a model of $\mathrm{Comp}(\rightarrow, \mathrm{Pr})$. Let r be a rule of $\mathrm{Comp}(\rightarrow, \mathrm{Pr})$.

- Either r is like

$$\mathrm{lit} \leftarrow \mathrm{for}$$

and r is a rule of Pr′, the normal form of Pr. Since $T_{\mathrm{Pr}}(i) = T_{\mathrm{Pr}'}(i)$, $T_{\mathrm{Pr}'}(i) \subseteq i$; hence i is a model of Pr′ and finally $\mathrm{tv}_i(\mathrm{lit} \leftarrow \mathrm{for}) = \mathrm{True}$.

- Or r is like

$$\mathrm{lit} \rightarrow \mathrm{for}.$$

We have to show that $\mathrm{tv}_i(\mathrm{lit} \rightarrow \mathrm{for}) = \mathrm{True}$ for every ground instance of r; r is like

$$\mathrm{lit}(x_1, \ldots, x_n) \rightarrow \bigvee_i (\exists y_1 \cdots \exists y_p (x_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge x_n = t_{i,n}(y_1, \ldots, y_p) \wedge \mathrm{for}_i(y_1, \ldots, y_p))).$$

If $(t_1, \ldots, t_n) \in \mathrm{UNI}(L)^n$ then

$$\mathrm{lit}(t_1, \ldots, t_n) \rightarrow \bigvee_i (\exists y_1 \cdots \exists y_p (t_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge t_n = t_{i,n}(y_1, \ldots, y_p) \wedge \mathrm{for}_i(y_1, \ldots, y_p)))$$

is a ground instance of r. The only case to check is when $\mathrm{tv}_i(\mathrm{lit}(t_1, \ldots, t_n)) = \mathrm{True}$. Since $i \subseteq T_{\mathrm{Pr}}(i) = T_{\mathrm{Pr}'}(i)$, $\mathrm{tv}_i(\mathrm{lit}(t_1, \ldots, t_n)) = \mathrm{True}$ implies that $\mathrm{lit}(t_1, \ldots, t_n) \in T_{\mathrm{Pr}'}(i)$. So there is a ground instance:

$$\mathrm{lit}(t_1, \ldots, t_n) \leftarrow \bigvee_i (\exists y_1 \cdots \exists y_p (t_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge t_n = t_{i,n}(y_1, \ldots, y_p) \wedge \mathrm{for}_i(y_1, \ldots, y_p)))$$

of a rule of Pr′ such that $\mathrm{tv}_i(\mathrm{for}') = \mathrm{True}$ with

$$\mathrm{for}' = \bigvee_i (\exists y_1 \cdots \exists y_p (t_1 = t_{i,1}(y_1, \ldots, y_p) \wedge \cdots \wedge t_n = t_{i,n}(y_1, \ldots, y_p) \wedge \mathrm{for}_i(y_1, \ldots, y_p))).$$

So $\mathrm{tv}_i(\mathrm{lit}(t_1, \ldots, t_n) \rightarrow \mathrm{for}') = \mathrm{True}$.

(ii) Conversely if i is a model of Comp(→, Pr), then $i \neq \mathrm{Contra}$ and $T_{\mathrm{Pr}}(i) = i$. If i is a model of Comp(→, Pr) then i is a model of Pr′ if Pr′ is the normal form of Pr and a model of Pr since $\forall \mathrm{Pr} \equiv_{\mathrm{TH}} \forall \mathrm{Pr}'$. So $T_{\mathrm{Pr}}(i) \subseteq i$ and $i \neq \mathrm{Contra}$. Besides that, $\mathrm{tv}_i(\mathrm{lit} \rightarrow \mathrm{for}') = \mathrm{True}$ for every ground instance $\mathrm{lit} \rightarrow \mathrm{for}'$ of a rule of Comp(→, Pr). So if $\mathrm{lit} \in i$, $\mathrm{tv}_i(\mathrm{for}') = \mathrm{True}$ and $\mathrm{lit} \in T_{\mathrm{Pr}'}(i) = T_{\mathrm{Pr}}(i)$ since there is a ground instance:

$$\mathrm{lit} \leftarrow \mathrm{for}'$$

of Pr′ such that $\mathrm{tv}_i(\mathrm{for}') = \mathrm{True}$ and $T_{\mathrm{Pr}'}(i) \neq \mathrm{Contra}$. So $i \subseteq T_{\mathrm{Pr}}(i)$.

If Pr is consistent, $\mathrm{lfp}(T_{\mathrm{Pr}}) \neq \mathrm{Contra}$ is a model of Comp(→, Pr) which is consistent too. □

In the next section, we study the *operational* semantics of a program: we show how to get a three-valued interpreter from a bivalued one.

3. Interpreters

The meaning of a program is given by its least model. We now search for algorithms that compute that smallest model or give informations on it.

We thus define three kinds of interpreters:

- the ground answer interpreters: which answer if a ground atom (resp. literal) belongs to lbm(Pr) (resp. ltm(Pr));
- the open answer interpreters: which answer, for a given atom (resp. literal) with variables, $\mathrm{ato}(x_1, \ldots, x_n)$, (resp. $\mathrm{lit}(x_1, \ldots, x_n)$), a sequence of substitutions

$\theta_1, \theta_2, \ldots$, such that all the ground instances of $\theta_i(\text{ato}(x_1, \ldots, x_n))$ (resp. $\theta_i(\text{lit}(x_1, \ldots, x_n))$) are exactly the ground instances of $\text{ato}(x_1, \ldots, x_n)$ (resp. $\text{lit}(x_1, \ldots, x_n)$) which are in lbm(Pr) (resp. ltm(Pr));

- the saturating interpreters: which give a finite representation of lbm(Pr) (resp. ltm(Pr)).

Remark 3.1. It is easy to transform a three-valued interpreter for a bivalued program into a bivalued interpreter for this program since $\text{lbm(Pr)} = \text{pos}^{-1}(\text{ltm(Pr)})$ (Proposition 2.3).

Remark 3.2. It is interesting to see how to get a three-valued interpreter from a bivalued one. We first transform a three-valued program into a bivalued one with the following transformations:

Let Pr be a three-valued program on a first order logical language L. $\sigma(L)$ is the first order logical language obtained from L by adding to it, not-P, for each predicate symbol P of L. Let $\sigma(\text{Pr})$ be the bivalued program on $\sigma(L)$ obtained from Pr with the following transformations:

(1) We remove $\Leftrightarrow$ and $\Rightarrow$ by using

$$A \Rightarrow B \equiv_{\text{TH}} (\neg A \vee B),$$

$$A \Leftrightarrow B \equiv_{\text{TH}} (A \wedge B) \vee (\neg A \wedge \neg B).$$

(2) We put the connective $\neg$ just before the atoms by using

$$\neg(A \vee B) \equiv_{\text{TH}} (\neg A \wedge \neg B),$$

$$\neg(A \wedge B) \equiv_{\text{TH}} (\neg A \vee \neg B),$$

$$\neg(\forall x\, A(x)) \equiv_{\text{TH}} \exists x\, (\neg A(x)),$$

$$\neg(\exists x\, A(x)) \equiv_{\text{TH}} \forall x\, (\neg A(x)),$$

$$\neg\neg A \equiv_{\text{TH}} A.$$

(3) We replace all the $\neg$ato by not-ato.

Definition 3.3. If $i \in \text{IHT}(L)$, then $\sigma(i) = \text{pos}(i) \cup \{\text{not-}p(t_1, \ldots, t_n) \mid \neg p(t_1, \ldots, t_n) \in i\}$. The interpretation $\sigma(i)$ belongs to $\text{IHT}(\sigma(L))$. σ is a bijection between $\text{IHT}(L)$ and $\text{pos}(\text{IHB}(\sigma(L))$.

We can get a three-valued interpreter from a bivalued one with the following theorem:

Theorem 3.4. *If* Pr *is a consistent three-valued program, then*

$$\text{ltm(Pr)} = \sigma^{-1}(\text{pos}(\text{lbm}(\sigma(\text{Pr})))).$$

Proof. We show by a transfinite induction that $\sigma(T_{\text{Pr}}\uparrow\alpha) = \text{pos}(B_{\sigma(\text{Pr})}\uparrow\alpha)$ for every α ordinal. As $\text{ltm(Pr)} = T_{\text{Pr}}\uparrow\alpha$ and $\text{lbm}(\sigma(\text{Pr})) = B_{\sigma(\text{Pr})}\uparrow\beta$, we have the result by

taking the greatest ordinal. We show that,

$$\forall i \in \mathrm{IHT}(L), \quad \sigma(T_{\mathrm{Pr}}(i)) = (\mathrm{pos} \circ B_{\sigma(\mathrm{Pr})} \circ \mathrm{pos}^{-1})(\sigma(i)). \qquad (*)$$

We use the fact that $B_{\sigma(\mathrm{Pr})} = \mathrm{pos}^{-1} \circ T_{\sigma(\mathrm{Pr})} \circ \mathrm{pos}$ since $\sigma(\mathrm{Pr})$ is a bivalued program, according to Proposition 2.3. So we have to show that $\sigma(T_{\mathrm{Pr}}(i)) = T_{\sigma(\mathrm{Pr})}(\sigma(i))$.

$\sigma(T_{\mathrm{Pr}}(i)) = \mathrm{pos}(T_{\mathrm{Pr}}(i)) \cup \{\text{not-}p(t_1, \ldots, t_n)$ such that $\neg p(t_1, \ldots, t_n) \in T_{\mathrm{Pr}}(i)\}$. $T_{\sigma(\mathrm{Pr})}(\sigma(i)) = \{\mathrm{ato} \in \sigma(L) \mid$ there is a ground instance ato $\leftarrow$ for of $\sigma(\mathrm{Pr})$ such that $\mathrm{tv}_{\sigma(i)}(\mathrm{for}) = \mathrm{True}\}$. If $\mathrm{lit} \in \sigma(T_{\mathrm{Pr}}(i))$, either lit = ato with $\mathrm{ato} \in T_{\mathrm{Pr}}(i)$ or lit = not-ato with $\neg\mathrm{ato} \in T_{\mathrm{Pr}}(i)$ and we have the result because $\mathrm{tv}_{\sigma(i)}(\sigma(\mathrm{for})) = \mathrm{True} \Leftrightarrow \mathrm{tv}_i(\mathrm{for}) = \mathrm{True}$. We then have $\sigma(T_{\mathrm{Pr}} \uparrow \alpha) = \mathrm{pos}(B_{\sigma(\mathrm{Pr})} \uparrow \alpha)$ for every ordinal α, by transfinite induction: $\alpha = 0$, $T_{\mathrm{Pr}} \uparrow 0 = \emptyset = \mathrm{pos}(\emptyset)$.

If α is a successor ordinal, then

$$\begin{aligned}
&\sigma(T_{\mathrm{Pr}}(T_{\mathrm{Pr}} \uparrow \alpha - 1)) \\
&\quad = (\mathrm{pos} \circ B_{\sigma(\mathrm{Pr})} \circ \mathrm{pos}^{-1})(\sigma(T_{\mathrm{Pr}} \uparrow \alpha - 1)) \quad \text{(according to (*))} \\
&\quad = (\mathrm{pos} \circ B_{\sigma(\mathrm{Pr})} \circ \mathrm{pos}^{-1})(\mathrm{pos}(B_{\sigma(\mathrm{Pr})} \uparrow \alpha - 1)) \quad \text{by induction} \\
&\quad = \mathrm{pos}(B_{\sigma(\mathrm{Pr})} \uparrow \alpha).
\end{aligned}$$

If α is a limit ordinal, then

$$\begin{aligned}
\sigma(T_{\mathrm{Pr}} \uparrow \alpha) &= \sigma(\mathrm{lub}\{T_{\mathrm{Pr}} \uparrow \beta, \beta < \alpha\}) \\
&= \sigma(\mathrm{lub}\{\mathrm{pos}(B_{\sigma(\mathrm{Pr})} \uparrow \beta), \beta < \alpha\}) \quad \text{by induction;}
\end{aligned}$$

we have $\mathrm{lub}\{\mathrm{pos}(i), i \in I\} = \mathrm{pos}(\mathrm{lub}\{i, i \in I\})$; so $\sigma(T_{\mathrm{Pr}} \uparrow \alpha) = \mathrm{pos}(B_{\sigma(\mathrm{Pr})} \uparrow \alpha)$.

So $\sigma(\mathrm{ltm}(\mathrm{Pr})) = \mathrm{pos}(\mathrm{lbm}(\sigma(\mathrm{Pr})))$ and $\mathrm{ltm}(\mathrm{Pr}) = \sigma^{-1}(\mathrm{pos}(\mathrm{lbm}(\sigma(\mathrm{Pr}))))$. □

Example 3.5. Let Pr be the following program where A, B, C, D are ground atoms:

A

$\neg B \leftarrow$

$C \leftarrow A$

$\neg D \leftarrow \neg B, C$

$E \leftarrow C, \neg D.$

Then $\sigma(\mathrm{Pr})$ is

A

not-$B \leftarrow$

$C \leftarrow A$

not-$D \leftarrow$ not-B, C

$E \leftarrow C$, not-D,

$\mathrm{lbm}(\sigma(\mathrm{Pr})) = \{A, \text{not-}B, C, \text{not-}D, E, \neg\text{not-}A, \neg\text{not-}C, \neg\text{not-}E, \neg B, \neg D\}$,

$\mathrm{pos}(\mathrm{lbm}(\sigma(\mathrm{Pr}))) = \{A, \text{not-}B, C, \text{not-}D, E\}$,

$\sigma^{-1}(\mathrm{pos}(\mathrm{lbm}(\sigma(\mathrm{Pr})))) = \{A, \neg B, C, \neg D, E\} = \mathrm{ltm}(\mathrm{Pr})$.

By purely syntactical operations on interpretations and programs, every bivalued interpreter may be transformed into a three-valued one. This has been done for Prolog interpreters [6] and is actually a classical method used for expert systems working in forward chaining.

References

[1] K. Apt and M. Van Emden, Contribution to the theory of logic programming, *J. ACM* **29**(3) (1982) 841-842.

[2] M. Ben Jacob and M. Fitting, Stratified and three-valued logic programming semantics, in: R.A. Kowalski and A. Bowen, eds., *Proc. Fifth International Conference and Symposium on Logic Programming* (MIT Press, Cambridge, MA, 1988) 1055-1069.

[3] N.D. Belnap, *Modern Uses of Multi-valued Logic* (Reidel, Dordrecht, 1977).

[4] K.L. Clark, Negation as failure, in: Gallaire and Minker, eds., *Logic and Databases* (Plenum Press, New York, 1978) 293-324.

[5] J.P. Delahaye, Programmation en logique trivaluée, Publication interne I.T. No. 115, Université des Sciences et Techniques de Lille, 1987.

[6] J.P. Delahaye and P. Mathieu, Logique partielle et Prolog, Séminaire de programmation logique de Tregastel, 1989.

[6a] J.P. Delahaye and V. Thibau, The optimal model of a logic program with negation, *Jelia 1990*, to appear in Lecture Notes in Computer Science (Springer, Berlin).

[7] M. Fitting, A Kripke-Kleene semantic for logic programs, *J. Logic Programming* **4** (1985) 295-312.

[8] M. Fitting, Notes on the mathematical aspects of Kripke's theory of truth, *Notre Dame J. Formal Logic* **27**(1) (1986) 75-88.

[9] M. Fitting, Partial models and logic programming, *Theoret. Comput. Sci.* **48** (1986) 229-255.

[10] D.M. Gabay and J. Sergot, Negation as inconsistency, *J. Logic Programming* **1** (1986) 1-35.

[11] K. Kunen, Negation in logic programming, *J. Logic Programming* **2** (1987) 289-308.

[12] K. Kunen, Some remarks on the completed databases, in: R. Kowalski and K. Bowen, eds., *Proc. Fifth International Conference on Logic Programming* (MIT Press, Cambridge, MA, 1988) 978-992.

[13] J.L. Lassez and M.J. Maher, Optimal fixpoints of logic programs, *Theoret. Comput. Sci.* **39** (1985) 15-25.

[14] J.W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 2nd ed., 1987).

[15] P. Mancarella, S. Martini and D. Pedreschi, Complete logic programs with domain-closure axiom, *J. Logic Programming* **5** (1988) 263-276.

[16] Z. Manna and A. Shamir, The theoretical aspects of the optimal fixpoints, *SIAM J. Comput.* **5**(3) (1976) 414-426.

[17] Z. Manna and A. Shamir, The optimal approach to recursive programs, *J. ACM* **20**(11) (1977) 824-831.

[18] A. Mycroft, Logic programs and many-valued logic, in: *STACS 84*, Lecture Notes in Computer Science **166** (Springer, Berlin, 1984) 274-286.

[19] T. Przymunsinski, Non monotonic formalisms and logic programming, in: *Proc. ICLP '89* (1989) 655-674.

[20] T. Przymunsinski, On the declarative semantics of deductive databases and logic programs, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufman, Los Altos, 1988).

[21] J.B. Rosser and A.R. Turquette, *Many-valued Logic* (North-Holland, Amsterdam, 1958).

[22] J.C. Shepherdson, Negation in logic programming, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufman, Los Altos, 1988) 19-87.

[23] J.C. Shepherdson, A sound and complete semantics for negation as failure, Research Report.

[24] V. Thibau, Une logique trivaluée appliquée à la programmation logique, Thèse de doctorat, Université des Sciences et Techniques de Lille, 1990.

[25] L. Thorne MacCarty, Fixedpoints semantics, *J. Logic Programming* **5** (1988) 3-31.

[26] R. Turner, *Logiques pour l'Intelligence Artificielle* (Masson, Paris, 1987).
[27] M.H. Van Emden, Quantitative deduction and its fixpoints theory, *J. Logic Programming* **1** (1986) 37-53.
[28] M.H. Van Emden and R.A. Kowalski, The semantics of predicate logic as a programming language, *J. ACM* **23** (1976) 733-742.
[29] A. Van Gelder, Negation as failure using tight derivations, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufman, Los Altos, 1988).
[30] A. Yasuhara, *Theory of Recursive Functions* (Academic Press, New York, 1971).