

Syntactical characterization of a subset of Domain Independent Formulas

Robert DEMOLOMBE
ONERA-CERT
2 Av. E.Belin BP 4025
31055 Toulouse FRANCE
demolomb@tls-cs.cert.fr

January 17, 2003

Abstract

A Domain Independent formula of First Order Predicate Calculus is a formula whose evaluation in a given interpretation does not change when we add a new constant to the interpretation domain. The formulas used to express queries, integrity constraints or deductive rules in the database field which have an intuitive meaning are Domain Independent. That is the reason why this class is of great interest in practice. Unfortunately this class is not decidable, and the problem is to characterize new sub-classes, as large as possible, which are decidable. We provide a syntactic characterization of a class of formulas, the Evaluable formulas, which are proved to be Domain Independent. This class is defined only for function free formulas. It is also proved that the class of Evaluable formulas contains the other classes of syntactically characterized Domain Independent formulas usually found in the literature, namely : Range Separable formulas and Range Restricted formulas. Finally we show that the expressive power of Evaluable formulas is the same as that of Domain Independent formulas. That is, each Domain Independent formula admits an equivalent Evaluable one. An important advantage of this characterization is that, to check if a formula is Evaluable, it is not necessary to transform it to a normal form, as is the case for Range Restricted formulas.

1 Introduction

In the Data Base field the First Order Predicate Calculus language ¹ is used by many researchers because it provides the framework of Mathematical Logic, and by users because it provides an easy way to express knowledge about the universe of an application.

In this paper we consider only a subset of the **language without functions**.

First of all this language has been used as a query language and was implemented, under different syntactical aspects, in Data Base Management Systems like QBE [21] and SYNTEX [2, 11].

Another application of the Predicate Calculus language is with rules that express integrity constraints on, or derive new information from, the information in the Data Base [13, 14].

A third application field is for expressing properties of the relations, defining different kinds of dependencies and normal forms [12].

However some problems may arise with the Predicate Calculus language, because the meaning of some formulas of the language is not clear. For example the query “who are the x who do not teach English ?” can be expressed by the formula : $\neg\text{Teach}(x, \text{English})$. The meaning of this query is not well-defined since it is not obvious that the set of possible values of the variable x is the set of persons, or the set of teachers, or the set of teachers who teach some language; likewise we cannot say if the sentence “All the x have a PhD”, expressed by the formula : $\forall x \text{Diploma}(x, \text{PhD})$, is true or false until the set to which the universal quantifier is applied has been specified. These formulas have a formal meaning in a given interpretation, but this formal meaning can change in other interpretations even when they express the same positive knowledge relative to a given universe.

From a practical point of view these queries cannot be translated into the Relational Algebra and cannot be evaluated by a Relational DBMS.

Another example of a meaningless formula is : $\forall x \forall y (\text{Language}(y) \rightarrow \text{Teach}(x, y))$. If such a formula is used, for example, with $\text{Language}(\text{English})$, we can derive $\text{Teach}(x, \text{English})$ for any value of x; not only values which denote any teacher and any person, but also any entity like : English, PhD, ..etc.. .

¹The Predicate Calculus language is also called Domain Calculus in [20].

These problems have been tackled by many researchers. Some researchers have semantically characterized the set of formulas which have a clear intuitive meaning : J.L.Kuhns, who introduced in 1967 the concept of Definite Formula, and J.D.Ullman, who introduced in 1980 the concept of Safe Formula. It has been shown in [18] that if we extend the class of Safe Formulas by equivalence, we obtain the same class as the class of Definite Formulas. So these concepts are very close, and this class is usually called the class of Domain Independent Formulas [12, 16].

In 1969 R.A. Di Paola showed in [19] that the class of Domain Independent Formulas is not solvable. That is the reason why we cannot provide a syntactical characterization of the class of all Domain Independent Formulas. Some researchers have tried to define the largest possible solvable subset of this class. In the field of query languages J.L.Kuhns defined in [15] the Proper Formulas, A.Artaud and J.M.Nicolas defined in [1] the Acceptable Formulas, E.F.Codd, for the tuple Predicate Calculus, the Range Separable Formulas, and E.Cooper defined in [7] the Permissible Formulas. In the field of integrity constraints J.M.Nicolas defined the Range Restricted Formulas. To express the general rules used to deduce information in the context of non-Monotonic Logic, G.Bossu and P.Siegel [4], and P.Besnard [3] use Predefined Variable Formulas. Finally, to characterize a large class of dependencies R.Fagin defined in [12] a subset of Horn clauses called implicational dependencies.

In section 3 of this paper we syntactically define the set of Evaluable Formulas. It will be proved in section 4 that Evaluable Formulas are Domain Independent. In section 5 we will show that this class of formulas is larger than the previous ones presented in the literature and in section 6 it will be shown that Evaluable Formulas have the same expressive power as Domain Independent Formulas.

2 Definition of Domain Independent Formulas

2.1 Definition of Predicate Calculus language

We briefly recall the definition of the Predicate Calculus language which will be used in the next sections. The vocabulary of the language is as follows :

Constant names : a,b,c,...

Variable names : x,y,z,...

Predicate names : P,Q,R,...
Logical symbols : $\neg, \vee, \wedge, \forall, \exists,), ($

We consider a Predicate Calculus language **without function symbols**. The well formed formulas (WFF's) of the language are defined by the following rules :

1. Each atomic formula is a WFF. Atomic formulas are Predicate names followed by a list of arguments in parentheses. The arguments may be constants or variables.
2. If F_1 and F_2 are WFF's then $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$ are WFF's.
3. If F_1 is a WFF then $(\neg F_1)$, $(\forall x F_1)$ and $(\exists x F_1)$ are WFF's.
4. All the WFF's are defined by the rules 1,2 and 3, modulo the following remarks.

Remarks :

- When there is no risk of ambiguity some parentheses may be omitted.
- It is assumed that quantified variables are **free in the scope of their quantifiers**. That is the formula $\exists x P(y)$ is not allowed.
- It is assumed that quantified variables cannot appear out of the scope of their quantifiers. That is, **a variable cannot be both free and quantified, and it is quantified at most once**. For instance the formula : $P(x, y) \wedge \exists x(Q(x, y) \wedge R(x)) \wedge \exists x Q(x, x)$, has to be written in the form : $P(x, y) \wedge \exists z(Q(z, y) \wedge R(z)) \wedge \exists t Q(t, t)$.
- We **do not permit use of the equality predicate**; we will see later the reason for this restriction.

An **open formula** (resp. **closed formula**) is a formula having at least one free variable (resp. having no free variable).

2.2 Interpretation of the language

Let D be a non empty set : $D = \{ a_1, a_2, a_3, \dots \}$. An interpretation I of the Predicate Calculus language is a mapping which assigns to each constant c_i an element

a_j of D , and which assigns to each predicate P_i of rank n_i a subset of D^{n_i} , called the relation R_i . We will use the shorthand notation $I = \langle D, R_1, R_2, \dots, R_n \rangle$ for an interpretation I assuming that the constants and their corresponding elements in the domain have the same name.

Valuation of a formula : let $F(x_1, x_2, \dots, x_s)$ be a formula, where x_1, x_2, \dots, x_s are the free variables of F , and let I be an interpretation, the valuation $T(F, I)$ of the formula F in the interpretation I is :

- if $s > 0$: the set of the elements of D^s which satisfy F ,
- if $s = 0$: t or f , according to whether F is satisfied or not satisfied in I ,

where t and f respectively denote **true** and **false**.

2.3 Stable Formulas

The anomalies of some formulas mentioned in the introduction can be formally defined using the concept of stable formulas.

Let F be a formula in which only the predicate symbols P_1, P_2, \dots, P_t occur, and I and I' be two interpretations where the predicates P_1, P_2, \dots, P_t are interpreted by the same relations R_1, R_2, \dots, R_t :

$$I = \langle D, R_1, R_2, \dots, R_t, R_{t+1}, \dots, R_n \rangle$$

$$I' = \langle D', R_1, R_2, \dots, R_t, R'_{t+1}, \dots, R'_n \rangle$$

We say that I and I' are equivalent with regard to F , and express this in our notations as $E(F, I, I')$.

Definition 1 (D1). Stable Formulas :

A formula F is stable iff :

$$\forall I \forall I' (E(F, I, I') \Rightarrow T(F, I) = T(F, I'))$$

This property will be illustrated by means of a short example. Let us consider

a Data Base with three relations : Married, Speak and Teach, whose interpretations are :

Married	
John	Lucy
Jack	Marilyn

Speak	
John	English
Lucy	English
Jack	English
Marilyn	English

Teach	
Lucy	English

This Data Base can be formalized as an interpretation I which has the domain $D = \{John, Lucy, Jack, Marilyn, English\}$ and the relations Married, Speak and Teach, so that :

$$I = \langle D, \text{Married}, \text{Speak}, \text{Teach} \rangle$$

Suppose now we add the new information to the Data Base : “Robert teaches French”. The new relations are :

Married	
John	Lucy
Jack	Marilyn

Speak	
John	English
Lucy	English
Jack	English
Marilyn	English

Teach'	
Lucy	English
Robert	French

The new interpretation associated to this Data Base will be :

$$I' = \langle D', \text{Married}, \text{Speak}, \text{Teach}' \rangle$$

with :

$$D' = \{John, Lucy, Jack, Marilyn, English, Robert, French\}$$

Let us now consider the formulas :

$$F_1 = \neg \text{Married}(x, \text{Marilyn}) \quad F_2 = \text{Speak}(x, \text{English}) \wedge \neg \text{Married}(x, \text{Marilyn})$$

We have :

$T(F_1, I) = \{\text{John, Lucy, Marilyn, English}\}$
 $T(F_1, I') = \{\text{John, Lucy, Marilyn, English, Robert, French}\}$

so we have $T(F_1, I) \neq T(F_1, I')$ but $E(F, I, I')$. Therefore F_1 is not stable.

On the other hand F_2 is stable. Note that $E(F, I, I')$ and :

$$T(F_2, I) = \{\text{John, Lucy, Jack}\} \quad T(F_2, I') = \{\text{John, Lucy, Jack}\}$$

F_2 is stable because for each $\langle I, I' \rangle$, $E(F, I, I')$ will hold.

We can make two remarks about F_1 :

- If we are dealing with Many Sorted Logic we find the same anomaly $T(F_1, I) \neq T(F_1, I')$. Suppose we have two sorts : Person and Language, and x is of the sort Person in F_1 , then we have :

$$T_S(F_1, I) = \{\text{John, Lucy, Marilyn}\}$$

$$T_S(F_1, I') = \{\text{John, Lucy, Marilyn, Robert}\}$$

that is $T_S(F_1, I) \neq T_S(F_1, I')$, where $T_S(F, I)$ is the valuation of F in the Many Sorted interpretation I .

- If the domain of I is infinite then $T(F_1, I)$ is an infinite set.

2.4 Safe Formulas

Safe Formulas were introduced by J.D.Ullman [20] for reasons similar of those of as J.L.Kuhns. We will first define the concept of “Domain of a formula”.

Let F be a formula. We define $D_F(x)$, the domain of F , by :

If F is an atomic formula $P(t_1, t_2, \dots, t_n)$, where the t_i are variable symbols or constant symbols, then :

$$D_F(x) = \exists x_2 \exists x_3 \dots \exists x_n P(x, x_2, x_3, \dots, x_n) \vee$$

$$\exists x_1 \exists x_3 \dots \exists x_n P(x_1, x, x_3, \dots, x_n) \vee$$

$$\dots$$

$$\exists x_1 \exists x_2 \dots \exists x_{n-1} P(x_1, x_2, \dots, x_{n-1}, x) \vee$$

$$x = t_{i_1} \vee x = t_{i_2} \vee \dots \vee x = t_{i_p}$$

where the t_i are all the constant symbols in F .

If F is not an atomic formula, and A_1, A_2, \dots, A_m are all the atomic formulas occurring in F , then :

$$D_F(x) = D_{A_1}(x) \vee D_{A_2}(x) \vee \dots \vee D_{A_m}(x)$$

For example, if $F = P(a, x) \wedge Q(x, y)$ we have :

$$D_F(x) = \exists x_2 P(x, x_2) \vee \exists x_1 P(x_1, x) \vee x = a \vee \exists x_2 Q(x, x_2) \vee \exists x_1 Q(x_1, x)$$

Definition 2 (D2). Safe Formulas :

A formula $F(x_1, x_2, \dots, x_r)$ whose free variables are x_1, x_2, \dots, x_r is Safe iff :

1. $F(a_1, a_2, \dots, a_r) \Rightarrow D_F(a_1) \wedge D_F(a_2) \wedge \dots \wedge D_F(a_r)$
2. For each sub-formula of F of the form $\exists x_j F'(x_1, x_2, \dots, x_j, \dots, x_s)$ and for each $a_1, a_2, \dots, a_j, \dots, a_s$, we have : $F'(a_1, a_2, \dots, a_j, \dots, a_s) \Rightarrow D_F(a_j)$.
3. For each sub-formula of F of the form $\forall x_k F''(x_1, x_2, \dots, x_k, \dots, x_t)$ and for each $a_1, a_2, \dots, a_k, \dots, a_t$, we have : $\neg D_F(a_k) \Rightarrow F''(a_1, a_2, \dots, a_k, \dots, a_t)$.

where the a_i s are constant names.

Remark : the definition given by Ullman does not consider the case of closed formulas; the equivalence of the expressive power of Safe Formulas and of Relational Algebra shown by Ullman is valid only for open formulas since truth values cannot be represented by formulas in the Relational Algebra.

We note that for any Safe Formula $F(x)$, $\exists x F(x)$ (resp. $\forall x F(x)$) and $\exists x (D_F(x) \wedge F(x))$ (resp. $\forall x (D_F(x) \rightarrow F(x))$) have always the same valuation.

We note also that the property of a formula of being Safe is not entirely semantic . Indeed there are formulas, for example $(\exists x P(x)) \vee (\exists y Q(y))$, which are Safe and which are such that there are equivalent formulas, for example $(\exists x (\exists y (P(x) \vee Q(y))))$, which are not Safe.

2.5 Definite Formulas

The Definite Formulas were defined by J.L.Kuhns in [15]. Their definition refers to the concept of the *-extension of an interpretation.

Let I be the interpretation $\langle D, R_1, R_2, \dots, R_n \rangle$. The *-extension of I , called I^* , is an interpretation which has the same domain plus a new constant, $*$, which is different from all the elements of D , and where the predicates are interpreted by the same relations. That is :

$$\begin{aligned} I &= \langle D, R_1, R_2, \dots, R_n \rangle \\ I^* &= \langle D', R_1, R_2, \dots, R_n \rangle \\ D' &= D \cup \{*\} \end{aligned}$$

Definition 3 (D3). Definite Formulas (or Domain Independent Formulas)

A formula is definite iff :

$$\forall I (T(F, I) = T(F, I^*))$$

It is intuitively clear, and is formally proved in [17], that the class of stable formulas is identical to the class of definite formulas. Moreover this class is identical to the class of formulas which are equivalent to a Safe formula. In the sequel this class will be called the class of **Domain Independent** formulas, and for convenience we will use in the proofs the definition (D3) for the definition of Domain Independent formulas.

Remark : the reason why we consider only *formulas without the equality predicate* in this paper is that the interpretation of this predicate is not free, in particular any interpretation must satisfy $\forall x(x = x)$. The consequence is that the equality interpretation cannot be the same in I and I^* , because in I^* we have $* = *$. The definitions and the results presented in the next section could be adapted to a language with equality, but the definitions would be more complex at the level of atomic formulas, because we might have to distinguish atomic formulas formed with equality.

3 Definition of Evaluable Formulas

We will give a syntactic characterization ² of a subset of Domain Independent formulas which is based on the concepts of Restricted, Unrestricted, Positive and Negative variables.

We will use the following notations :

$FRes(x,F)$: means that the free variable x is Restricted in formula F .

$FUnres(x,F)$: means that the free variable x is Unrestricted in formula F .

$FPos(x,F)$: means that the free variable x is Positive in formula F .

$FNeg(x,F)$: means that the free variable x is Negative in formula F .

$Free(x,F)$: means that the variable x is free in formula F .

Before formally defining these concepts, we present the intuitive ideas behind them.

The first idea is to define the property of Restrictedness in such a way that if a free variable x is restricted in a formula F , then for any tuple satisfying F the instantiation of x must belong to the interpretations of the predicates which appear in F . If a formula contains only free variables, and if this property holds for all its free variables, we can see that the truth value of F , for a given instantiation of its variables depends only on the predicate interpretations, and therefore F is Domain Independent.

The second idea is that it is not necessary to impose on quantified variables conditions as strong as for free variables. For example on existentially quantified variables we impose the condition that they be Positive, which is weaker than requiring them to be Restricted. Positiveness guarantees that for the quantified subformula $F_1(x)$ there is *at least one* instantiation of the quantified variable x satisfying $F_1(x)$ in I iff there is *at least one* instantiation of x satisfying $F_1(x)$ in I^* . For example, for $F = \exists x(P(x) \vee Q(a))$, the number of x instantiations satisfying $F_1(x)$ may be different in I and I^* , although the truth value of F is always the same in I and I^* .

The third idea is to define recursively the properties imposed on the variables in terms of the logical operators, and in terms of the sub-formulas. The conditions imposed for each logical operator are not independent. In particular if we call Property one of the properties : $FRes$, $FUnres$, $FPos$ or $FNeg$, and if we respectively call $\overline{\text{Property}}$ one of the properties : $FUnres$, $FRes$, $FNeg$ or

²Another presentation of Evaluable Formulas can be found in [8] or in [10]; the definition in [8] contains a slight error.

FPos, we will define $\text{Property}(x, \neg F)$ in terms of $\overline{\text{Property}}(x, F)$. That is, we impose the conditions :

$$\begin{aligned}\text{Property}(x, \neg F) &\Leftrightarrow \overline{\text{Property}}(x, F) \\ \overline{\text{Property}}(x, \neg F) &\Leftrightarrow \text{Property}(x, F)\end{aligned}$$

as a consequence we have :

$$\text{Property}(x, \neg\neg F) \Leftrightarrow \text{Property}(x, F)$$

Another consequence of our definitions is that whenever we impose a Property on an existentially quantified variable, we impose $\overline{\text{Property}}$ on a universally quantified variable in the same situation, because we have $\forall x(F)$ equivalent to $\neg\exists x(\neg F)$, and $\text{Property}(x, \neg F)$ is equivalent to $\overline{\text{Property}}(x, F)$.

We note also that $\overline{\text{Property}}(x, F_1 \wedge F_2)$ and $\overline{\text{Property}}(x, F_1 \vee F_2)$ are respectively defined in terms of $\text{Property}(x, F_1 \vee F_2)$ and $\text{Property}(x, F_1 \wedge F_2)$. For example we have :

$$\overline{\text{Property}}(x, F_1 \wedge F_2) \Leftrightarrow \text{Property}(x, (\neg F_1) \vee (\neg F_2))$$

because $F_1 \wedge F_2$ is equivalent to $\neg((\neg F_1) \vee (\neg F_2))$. Thus we can get the definition of $\overline{\text{Property}}(x, F_1 \wedge F_2)$ from the definition of $\text{Property}(x, (\neg F_1) \wedge (\neg F_2))$ by replacing $\text{Property}(x, \neg F_1)$ and $\text{Property}(x, \neg F_2)$, by $\overline{\text{Property}}(x, F_1)$ and $\overline{\text{Property}}(x, F_2)$, respectively.

In conclusion the definitions of FUNres and FNeg can be derived from the definitions of FRes and FPos. Then to understand the intuitive meaning of FRes, FPos, FUNres and FNeg we can concentrate on FRes and FPos alone.

The definition of FPos is not very intuitive and needs some comments in the case of a subformula of the form $F_1 \wedge F_2$ where the variable x is free in both F_1 and F_2 . In that case it is not enough to impose the condition on the variable x that it be Positive either in F_1 or in F_2 (as is the case for the FRes definition). Indeed we may have formulas like : $(P(x) \vee Q(y)) \wedge \neg R(x)$ where x is Positive in $P(x) \vee Q(y)$ and x is not Positive in $\neg R(x)$, and $\exists y \exists x((P(x) \vee Q(y)) \wedge \neg R(x))$ is not Domain Independent. However a formula like $\exists y \exists x((P(x) \vee Q(y)) \wedge (R(x) \vee S(y)))$, where x is Positive in both formulas $P(x) \vee Q(y)$ and $R(x) \vee S(y)$, is Domain

Independent. Another way to have a Domain Independent formula when x is free in both operands but not Positive in both operands, is to require x to be restricted in one operand. Indeed in that case x is Restricted in $F_1 \wedge F_2$ and thus Positive in $F_1 \wedge F_2$. For example the formula $\exists x(P(x) \wedge \neg R(x))$ is Domain Independent, though x is free but not Positive in $\neg R(x)$, because x is Restricted in $P(x)$.

Though it is intuitively easier to understand Definition 4 and Definition 5 when we consider only FRes and FPos and the logical operators $\wedge, \vee, \neg, \forall$ and \exists , from a technical point of view the proofs of the following theorems T1, T2 and T3 are simpler if we consider only the logical operators \vee, \neg and \exists . For this reason the definitions (D4) and (D5) are presented with the logical operators \vee, \neg and \exists , for the formal proofs, and they are extended to the operators \wedge and \forall , for intuitive understanding and in view of practical implementation. Of course the proofs given for formulas containing \vee, \neg or \exists are also valid for formulas containing \wedge or \forall since these operators can be rewritten in terms of the former ones.

Definition 4 (D4). Restricted and Unrestricted variables :

$$\begin{aligned} \text{FRes}(x, F) &\Leftrightarrow \text{Free}(x, F) \wedge \text{FResCase}(x, F) \\ \text{FResCase}(x, F_1 \vee F_2) &\Leftrightarrow \text{FRes}(x, F_1) \wedge \text{FRes}(x, F_2) \\ \text{FResCase}(x, \exists y F_1) &\Leftrightarrow \text{FRes}(x, F_1) \\ \text{FResCase}(x, \neg F_1) &\Leftrightarrow \text{FUNres}(x, F_1) \\ (\text{AtomicFormula}(F) \Rightarrow \\ &\quad \text{FResCase}(x, F) \Leftrightarrow \text{True}) \end{aligned}$$

$$\begin{aligned} \text{FUNres}(x, F) &\Leftrightarrow \text{Free}(x, F) \wedge \text{FUNresCase}(x, F) \\ \text{FUNresCase}(x, F_1 \vee F_2) &\Leftrightarrow \text{FUNres}(x, F_1) \vee \text{FUNres}(x, F_2) \\ \text{FUNresCase}(x, \exists y F_1) &\Leftrightarrow \text{FUNres}(x, F_1) \\ \text{FUNresCase}(x, \neg F_1) &\Leftrightarrow \text{FRes}(x, F_1) \\ (\text{AtomicFormula}(F) \Rightarrow \\ &\quad \text{FUNresCase}(x, F) \Leftrightarrow \text{False}) \end{aligned}$$

If we accept in the language the \wedge and \forall operators we have :

$$\begin{aligned} \text{FResCase}(x, F_1 \wedge F_2) &\Leftrightarrow \text{FRes}(x, F_1) \vee \text{FRes}(x, F_2) \\ \text{FResCase}(x, \forall y F_1) &\Leftrightarrow \text{FRes}(x, F_1) \end{aligned}$$

$$\text{FUNresCase}(x, F_1 \wedge F_2) \Leftrightarrow \text{FUNres}(x, F_1) \wedge \text{FUNres}(x, F_2)$$

$$\text{FUNresCase}(x, \forall y F_1) \Leftrightarrow \text{FUNres}(x, F_1)$$

Definition 5 (D5). Positive and Negative variables :

$$\begin{aligned} \text{FPos}(x, F) &\Leftrightarrow \text{Free}(x, F) \wedge \text{FPosCase}(x, F) \\ \text{FPosCase}(x, F_1 \vee F_2) &\Leftrightarrow (\text{Free}(x, F_1) \Rightarrow \text{FPos}(x, F_1)) \wedge (\text{Free}(x, F_2) \Rightarrow \text{FPos}(x, F_2)) \\ \text{FPosCase}(x, \exists y F_1) &\Leftrightarrow \text{FPos}(x, F_1) \\ \text{FPosCase}(x, \neg F_1) &\Leftrightarrow \text{FNeg}(x, F_1) \\ (\text{AtomicFormula}(F) \Rightarrow \\ &\quad \text{FPosCase}(x, F) \Leftrightarrow \text{True}) \end{aligned}$$

$$\begin{aligned} \text{FNeg}(x, F) &\Leftrightarrow \text{Free}(x, F) \wedge \text{FNegCase}(x, F) \\ \text{FNegCase}(x, F_1 \vee F_2) &\Leftrightarrow \\ &(\text{Free}(x, F_1) \wedge \text{Free}(x, F_2) \Rightarrow \\ &\quad (\text{FNeg}(x, F_1) \wedge \text{FNeg}(x, F_2)) \vee \text{FUNres}(x, F_1) \vee \text{FUNres}(x, F_2)) \wedge \\ &(\text{Free}(x, F_1) \wedge \neg \text{Free}(x, F_2) \Rightarrow \\ &\quad \text{FNeg}(x, F_1)) \wedge \\ &(\text{Free}(x, F_2) \wedge \neg \text{Free}(x, F_1) \Rightarrow \\ &\quad \text{FNeg}(x, F_2)) \\ \text{FNegCase}(x, \exists y F_1) &\Leftrightarrow \text{FNeg}(x, F_1) \\ \text{FNegCase}(x, \neg F_1) &\Leftrightarrow \text{FPos}(x, F_1) \\ (\text{AtomicFormula}(F) \Rightarrow \\ &\quad \text{FNegCase}(x, F) \Leftrightarrow \text{False}) \end{aligned}$$

If we accept in the language the logical operators \wedge and \forall we have :

$$\begin{aligned} \text{FPosCase}(x, F_1 \wedge F_2) &\Leftrightarrow \\ &(\text{Free}(x, F_1) \wedge \text{Free}(x, F_2) \Rightarrow \\ &\quad (\text{FPos}(x, F_1) \wedge \text{FPos}(x, F_2)) \vee \text{FRes}(x, F_1) \vee \text{FRes}(x, F_2)) \wedge \\ &(\text{Free}(x, F_1) \wedge \neg \text{Free}(x, F_2) \Rightarrow \\ &\quad \text{FPos}(x, F_1)) \wedge \\ &(\text{Free}(x, F_2) \wedge \neg \text{Free}(x, F_1) \Rightarrow \\ &\quad \text{FPos}(x, F_2)) \\ \text{FPosCase}(x, \forall y F_1) &\Leftrightarrow \text{FPos}(x, F_1) \\ \text{FNegCase}(x, F_1 \wedge F_2) &\Leftrightarrow (\text{Free}(x, F_1) \Rightarrow \text{FNeg}(x, F_1)) \wedge (\text{Free}(x, F_2) \Rightarrow \text{FNeg}(x, F_2)) \\ \text{FNegCase}(x, \forall y F_1) &\Leftrightarrow \text{FNeg}(x, F_1) \end{aligned}$$

Corollaries

We can easily prove the following propositions from definitions D4 and D5.

1. $FRes(x, F) \Rightarrow FPos(x, F)$
2. $FUnres(x, F) \Rightarrow FNeg(x, F)$
3. $FUnres(x, F) \Leftrightarrow FRes(x, \neg F)$
4. $FNeg(x, F) \Leftrightarrow FPos(x, \neg F)$
5. A variable may be neither Restricted nor Unrestricted in a formula. For example the variable x in the formula : $P(x) \vee Q(y)$.
6. A variable may be neither Positive nor Negative in a formula. For example the variable x in the formula : $P(x) \vee \neg Q(x)$.

Definition 6 (D6). Evaluable Formulas

A formula F is evaluable iff :

- each free variable of F is Restricted in F ,
- each existentially quantified variable is Positive in the subformula which is in the range of its quantifier,
- each universally quantified variable is Negative in the subformula which is in the range of its quantifier.

It is easy to check that the following formuals are Evaluable : $\neg\neg Q(x)$, $P(x) \wedge \neg Q(x)$, $(P(x) \wedge \neg Q(x, y)) \wedge (R(y) \wedge \neg S(x, y))$, $P(x) \vee Q(x)$ $\exists x(P(x) \vee Q(a))$, $\exists x \exists y(P(x) \vee Q(y))$, $\forall x(\neg(P(x) \vee Q(a)) \vee \neg R(x))$, $\forall x(\neg R(x) \vee P(x) \vee Q(a))$, $\exists y \forall x(\neg P(x) \vee Q(x, y))$.

The following formulas are not Evaluable : $P(x) \vee Q(y)$, $\forall x(\neg P(x) \vee Q(x, y))$, $\forall x(\neg(P(x) \vee Q(a)) \vee R(x))$.

4 Evaluable Formulas are Domain Independent

Before proving that Evaluable formulas are Domain Independent we need to prove two preliminary theorems.

Notations :

$$X = \langle x_1, x_2, \dots, x_n \rangle$$

$$A = \langle a_1, a_2, \dots, a_n \rangle$$

$F(X)$ denotes that the free variables of F are $\langle x_1, x_2, \dots, x_n \rangle$

$F(A)$ denotes F after replacing each variable x_i by a_i

$s(F)$ = number of logical symbols \neg, \vee, \exists occurring in F

Theorem 1 (T1)

Let $F(x_1, x_2, \dots, x_n)$ be a formula whose free variables are x_1, x_2, \dots, x_n . For each interpretation I , and for the $*$ -extension I^* of I , if a tuple $A = \langle a_1, a_2, \dots, a_n \rangle$ satisfies F in I^* (i.e. $T(F(A), I^*) = t$), and if the variable x_i is Restricted in F , then the value a_i of x_i is different from $*$.

Proof

The proof is by induction on $s(F)$.

Induction hypothesis (H1). H1 can be written :

$$(H1) \text{ FRes}(x_i, F) \text{ and } T(F(A), I^*) = t \text{ and } s(F) < p + 1 \Rightarrow a_i \neq *$$

We have to prove that :

$$(1) \text{ FRes}(x_i, F) \text{ and } (2) T(F(A), I^*) = t \text{ and } (3) s(F) = p + 1 \\ \text{implies } a_i \neq *$$

We consider all the possible forms of F .

Case $F = F_1 \vee F_2$:

$$(D4), (1) \Rightarrow (4) \text{ FRes}(x_i, F_1) \text{ and } \text{FRes}(x_i, F_2)$$

$$(2) \quad \Rightarrow T(F_1(A_1, I^*)) = t \text{ or } T(F_2(A_2, I^*)) = t$$

We suppose, for example, that : (5) $T(F_1(A_1, I^*)) = t$

$$(4) \quad \Rightarrow (6) \text{ FRes}(x_i, F_1)$$

$$(H1), (5), (6), s(F) < p + 1 \Rightarrow a_i \neq *$$

Case $F = \exists x_{n+1} F_1(X, x_{n+1}) :$

$$(D4), (1) \Rightarrow (4) \text{ FRes}(x_i, F_1)$$

$$(2) \Rightarrow \text{exists } a_{n+1} \text{ such that } (5) T(F_1(A, a_{n+1}, I^*)) = t$$

$$(H1), (4), (5), s(F_1) = p \Rightarrow a_i \neq *$$

Case $F = \neg F' :$

Sub-case $F' = \neg F_1 :$

$$(D4), (1) \Rightarrow (4) \text{ FUnres}(x_i, F')$$

$$(D4), (4) \Rightarrow (5) \text{ FRes}(x_i, F_1)$$

$$(2), F \equiv F_1 \Rightarrow (6) T(F_1(A, I^*)) = t$$

$$(H1), (5), (6), s(F_1) = p - 1 \Rightarrow a_i \neq *$$

Sub-case $F' = F_1 \vee F_2 :$

$$(D4), (1) \Rightarrow (4) \text{ FUnres}(x_i, F')$$

$$(D4), (4) \Rightarrow (5) \text{ FUnres}(x_i, F_1) \text{ or } \text{FUnres}(x_i, F_2)$$

$$(D4), (5) \Rightarrow (6) \text{ FRes}(x_i, \neg F_1) \text{ or } \text{FRes}(x_i, \neg F_2)$$

$$(2) \Rightarrow (7) T(F_1(A_1, I^*)) = f \text{ and } T(F_2(A_2, I^*)) = f$$

$$(7) \Rightarrow (8) T(\neg F_1(A_1, I^*)) = t \text{ and } T(\neg F_2(A_2, I^*)) = t$$

$$(H1), (8), (7), s(\neg F_1) < p + 1 \text{ and } s(\neg F_2) < p + 1 \Rightarrow a_i \neq *$$

Sub-case $F' = \exists x_{n+1} F_1(X, x_{n+1}) :$

$$(D4), (1) \Rightarrow (4) \text{ FUnres}(x_i, F_1)$$

$$(D4), (4) \Rightarrow (5) \text{ FRes}(x_i, \neg F_1)$$

$$(2) \Rightarrow (6) \forall a_{n+1} \in D' T(F_1(A, a_{n+1}, I^*)) = f$$

$$(6) \Rightarrow (7) \forall a_{n+1} \in D' T(\neg F_1(A, a_{n+1}, I^*)) = t$$

$$(H1), (5), (7), s(\neg F_1) = p \Rightarrow a_i \neq *$$

Sub-case $F' =$ atomic formula :

We do not have to consider this case because $F' =$ atomic formula and $FRes(x_i, \neg F')$ are contradictory.

Therefore in all cases where $s(F) = p + 1$, (1) and (2) imply $a_i \neq *$.

Basis of induction :

For $p = 0$ F is an atomic formula. By definition of I^* there is no tuple in the relations of I^* containing $*$, therefore $T(F(A), I^*) = t \Rightarrow a_i \neq *$.

Lemma 1 (L1)

If F is an open Evaluable Formula then : $T(F, I^*) \cap D^n = T(F, I^*)$.

Proof : Let X be the set of free variables of F . From (D6), for each x_i in X we have $Res(x_i, F)$.

Let A be an element of D^n . From (T1), if we have $T(F(A), I^*) = t$, all the elements of A are different of $*$, then $A \in D^n$. Hence the valuation of F in I^* is included in D^n .

Definition 7 (D7). Quasi-Evaluable Formulas

A formula F is quasi-evaluable iff :

- each free variable of F is Positive in F ,
- each existentially quantified variable is Positive in the subformula which is in the range of its quantifier,
- each universally quantified variable is Negative in the subformula which is in the range of its quantifier.

Notations : Let A be a tuple of D^n . $A(D)$ stands for the set of tuples obtained from A by substituting for any occurrence of $*$ all possible elements of D . For example if $D' = \{a, b, *\}$ and $A = \langle a, *, b, * \rangle$ then $A(D) = \{ \langle a, a, b, a \rangle, \langle a, b, b, a \rangle, \langle a, a, b, b \rangle, \langle a, b, b, b \rangle \}$.

Let X be a tuple of variables, let Y be a sub-tuple of X and let A be a tuple which is a value of X ; $A[Y]$ stands for the tuple obtained from A by substituting

for the variables of Y their corresponding values in A . For example if $X = \langle x_1, x_2, x_3, x_4 \rangle$, $Y = \langle x_1, x_3 \rangle$ and $A = \langle a_1, a_2, a_3, a_4 \rangle$ then $A[Y] = \langle a_1, a_3 \rangle$.

Theorem 2 (T2)

For each interpretation I , if $F(X)$ is a quasi-evaluable formula and $T(F(A), I^*) = t$ then for each A' in $A(D)$ $T(F(A'), I^*) = t$, where I^* is the $*$ -extension of I .

Proof : the proof is by induction on $s(F)$.

Induction hypothesis (H2). H2 can be written :

(H2) $F(X)$ quasi-evaluable and $T(F(A), I^*) = t$ and $s(F) < p+1 \Rightarrow \forall A' \in A(D) T(F(A'), I^*) = t$

We have to prove that :

(1) $F(X)$ quasi-evaluable and (2) $T(F(A), I^*) = t$ and (3) $s(F) = p+1$ implies $\forall A' \in A(D) T(F(A'), I^*) = t$

We consider all the possible forms of F :

Case $F = F_1 \vee F_2$:

Notations :

- X_1 : tuple of free variables of F_1
- X_2 : tuple of free variables of F_2
- A_1 : projection of A on X_1
- A_2 : projection of A on X_2

Let x_i be a variable of X_1 :

- (D5),(1) \Rightarrow (4) $FPos(x_i, F_1)$
- (4) \Rightarrow (5) $F_1(X_1)$ quasi-evaluable

We prove in the same way that : (6) $F_2(X_2)$ quasi-evaluable

- (2) $\Rightarrow T(F_1(A_1), I^*) = t$ or $T(F_2(A_2), I^*) = t$

We suppose that : (7) $T(F_1(A_1), I^*) = t$

- (H2),(5),(7), $s(F_1) < p + 1 \Rightarrow$ (8) $\forall A'_1 \in A_1(D) T(F_1(A'_1), I^*) = t$
(8) \Rightarrow (9) $\forall A' \in A(D) T(F_1(A'_1), I^*) = t$
(9) \Rightarrow (10) $\forall A' \in A(D) T(F_1(A'_1) \vee F_2(A'_2), I^*) = t$
(10) \Rightarrow (11) $\forall A' \in A(D) T(F(A'), I^*) = t$

Case $F = \exists x_{n+1} F_1(X, x_{n+1}) :$

- (2) $\Rightarrow \exists a_{n+1} \in D'$ (3) $T(F_1(A, a_{n+1}), I^*) = t$
(D5),(1) \Rightarrow (4) $F_1(X, a_{n+1})$ quasi-evaluable
(H2),(3),(4), $s(F_1) = p \Rightarrow$ (5) $\forall A' \in A(D) T(F_1(A', a_{n+1}), I^*) = t$
(5) \Rightarrow (6) $\forall A' \in A(D) T(\exists x_{n+1} F_1(A', x_{n+1}), I^*) = t$
(6) \Rightarrow (7) $\forall A' \in A(D) T(F(A'), I^*) = t$

Case $F = \neg F' :$

Sub-case $F' = \neg F_1 :$

Let x_i be a variable of $X :$

- (D5),(1) \Rightarrow (3) $FNeg(x_i, F')$
(D5),(3) \Rightarrow (4) $FPos(x_i, F_1)$
(2) \Rightarrow (6) $T(F_1(A), I^*) = t$
(H2),(4),(6), $s(F_1) = p - 1 \Rightarrow$ (7) $\forall A' \in A(D) T(F_1(A'), I^*) = t$
(7), $F \equiv F_1 \Rightarrow$ (8) $\forall A' \in A(D) T(F(A'), I^*) = t$

Sub-case $F' = F_1 \vee F_2 :$

We will use the notations :

- X_1 : tuple of free variables of F_1
 Y_1 : tuple of negative variables of F_1
 Z_1 : tuple of non-negative variables of F_1
 $A_1 = A[X_1] \quad A'_1 = A'[X_1]$
 $B_1 = A[Y_1] \quad B'_1 = A'[Y_1]$
 $C_1 = A[Z_1] \quad C'_1 = A'[Z_1]$

Similar notations will be used for F_2 .

Let x_i be a variable of Z_1 .

From the notations we have : (4) not $\text{FNeg}(x_i, F_1)$

(D5),(1) \Rightarrow (5) $\text{FNeg}(x_i, F_1 \vee F_2)$

(D5),(4),(5),(Corollary 2)

\Rightarrow (6) $\text{FUnres}(x_i, F_2)$

(6) \Rightarrow (7) $\text{FRes}(x_i, \neg F_2)$

(2) \Rightarrow (8) $T(\neg F_1(A_1), I^*) = t$ and (9) $T(\neg F_2(A_2), I^*) = t$

(T1),(7),(9) \Rightarrow (10) $a_i \neq *$

Since (10) holds for any variable in Z_1 we have :

(11) C_1 does not contain $*$

In the same way we have :

(12) C_2 does not contain $*$

From the notations we have :

(13) $\neg F_1(Y_1, C_1)$ quasi-evaluable

(H2),(8),(13), $s(\neg F_1) < p + 1 \Rightarrow$ (14) $\forall B'_1 \in B_1(D) T(\neg F_1(B'_1, C_1), I^*) = t$

(14),(11) \Rightarrow (15) $\forall A'_1 \in A_1(D) T(\neg F_1(A'_1), I^*) = t$

In the same way we have :

(16) $\forall A'_2 \in A_2(D) T(\neg F_2(A'_2), I^*) = t$

(15),(16) \Rightarrow (17) $\forall A' \in A(D) T((\neg F_1(A'_1)) \wedge (\neg F_2(A'_2)), I^*) = t$

(17), $\neg F' \equiv (\neg F_1) \wedge (\neg F_2) \Rightarrow \forall A' \in A(D) T(\neg F'(A'), I^*) = t$

Sub-case $F' = \exists x_{n+1} F_1(X, x_{n+1})$:

(2) \Rightarrow (4) $\forall a_{n+1} \in D' T(F_1(A, a_{n+1}), I^*) = f$

(4) \Rightarrow (5) $\forall a_{n+1} \in D' T(\neg F_1(A, a_{n+1}), I^*) = t$

(D5),(1) \Rightarrow (6) $\neg F_1(X, a_{n+1})$ quasi-evaluable

(H2),(5),(6), $s(\neg F_1) = p \Rightarrow$ (7) $\forall a_{n+1} \in D' \forall A' \in A(D) T(\neg F_1(A', a_{n+1}), I^*) = t$

(7) \Rightarrow (8) $\forall A' \in A(D) \forall a_{n+1} \in D' T(F_1(A', a_{n+1}), I^*) = f$

(8) $\Rightarrow \forall A' \in A(D) T(\neg \exists x_{n+1} F_1(A', x_{n+1}), I^*) = t$

Sub-case F' =atomic formula : we do not have to consider this case because F' =atomic formula and $\text{FPos}(x_i, \neg F')$ are contradictory.

Basis of induction :

For $p=0$, F is an atomic formula, so in all cases the variables of F are positive. If $T(F(A), I^*) = t$, by construction of I^* there can be no occurrence of $*$ in A , therefore $A(D) = \{A\}$ and $T(F(A), I^*) = t \Rightarrow \forall A' \in A(D) T(F(A'), I^*) = t$.

Theorem T3 (T3).

If F is an Evaluable formula then F is Domain Independent.

Proof : the proof is by induction on $s(F)$.

Induction hypothesis (H3).

(H3) F Evaluable and $s(F) < p + 1 \Rightarrow F$ Domain Independent

We have to prove that :

(1) F Evaluable and (2) $s(F) = p + 1$
implies $\forall I T(F, I) = T(F, I^*)$

We consider all the possible forms of F :

Let X be the tuple of variables which are free in F . Let x_i be a variable of X .

Case $F = F_1 \vee F_2$:

(1), Free(x_i, F) \Rightarrow (3) FRes(x_i, F)
(D4),(3) \Rightarrow (4) FRes(x_i, F_1) and FRes(x_i, F_2)
(D4),(4) \Rightarrow (5) Free(x_i, F_1) and Free(x_i, F_2)
Each variable which is free in F is also free in F_1 and F_2 hence,
if we call X_1 and X_2 the free variables of F_1 and F_2 , then :
(5) \Rightarrow (6) $X = X_1 = X_2$
(6) \Rightarrow (7) $T(F, I) = T(F_1, I) \cup T(F_2, I)$
(1),(4) \Rightarrow (8) F_1 Evaluable
(1),(4) \Rightarrow (9) F_2 Evaluable
(H3),(8), $s(F_1) < p + 1 \Rightarrow$ (10) $T(F_1, I) = T(F_1, I^*)$
(H3),(9), $s(F_2) < p + 1 \Rightarrow$ (11) $T(F_2, I) = T(F_2, I^*)$
(7),(10),(11) \Rightarrow (12) $T(F, I) = T(F_1, I^*) \cup T(F_2, I^*)$
(12) \Rightarrow (13) $T(F, I) = T(F, I^*)$

If F is a closed formula, we have a similar proof replacing (7) by $T(F, I) =$

$T(F_1, I) \vee T(F_2, I)$.

Case $F = \exists x_{n+1} F_1(X, x_{n+1})$:

(1) \Rightarrow (3) $FRes(x_i, F)$

(D4),(3) \Rightarrow (4) $FRes(x_i, F_1)$

Since we have $FRes(x_i, F_1)$ for each free variable in $F_1(X, a)$ we have :

(D6),(1),(4) \Rightarrow (5) $F_1(X, a)$ Evaluable

(H3),(5), $s(F_1) = p \Rightarrow$ (6) $T(F_1(X, a), I) = T(F_1(X, a), I^*)$

Until sentence (17) we assume $A \in D^n$.

From the definition of T we have :

(7) $T(F(A), I) = t \Leftrightarrow \exists a \in D T(F_1(A, a), I) = t$

(6),(7) \Rightarrow (8) $T(F(A), I) = t \Leftrightarrow \exists a \in D T(F_1(A, a), I^*) = t$

(D4),(1) \Rightarrow (9) $FPos(x_{n+1}, F_1)$

(9) \Rightarrow (10) $F_1(A, x_{n+1})$ quasi-evaluable

From the definition of T we have :

(11) $(\exists a \in D' T(F_1(A, a), I^*) = t) \Leftrightarrow ((\exists a \in D T(F_1(A, a), I^*) = t) \text{ or } T(F_1(A, *), I^*) = t)$

(T2),(10) \Rightarrow (12) $T(F_1(A, *), I^*) = t \Rightarrow \forall a \in D T(F_1(A, a), I^*) = t$

(12) \Rightarrow (13) $T(F_1(A, *), I^*) = t \Rightarrow \exists a \in D T(F_1(A, a), I^*) = t$

(11),(13) \Rightarrow (14) $(\exists a \in D' T(F_1(A, a), I^*) = t) \Leftrightarrow (\exists a \in D T(F_1(A, a), I^*) = t)$

(8),(14) \Rightarrow (15) $(T(F(A), I) = t) \Leftrightarrow (\exists a \in D' T(F_1(A, a), I^*) = t)$

(15) \Rightarrow (16) $(T(F(A), I) = t) \Leftrightarrow (T(F(A), I^*) = t)$

(16) \Rightarrow (17) $T(F, I) = T(F, I^*) \cap D^n$

(1),(L1),(17) \Rightarrow (18) $T(F, I) = T(F, I^*)$

If F is a closed formula the proof is very similar replacing (10) by $F_1(x_{n+1})$ evaluable, and $F_1(A, a)$ by $F_1(a)$.

Case $F = \neg F'$:

Sub-case $F' = \neg F_1$:

(D6),(1) \Rightarrow (3) $FRes(x_i, F)$

(D4),(3) \Rightarrow (4) $FUnres(x_i, F')$

(D4),(4) \Rightarrow (5) $FRes(x_i, F_1)$

(D6),(1),(5) \Rightarrow (6) $F_1(X)$ evaluable

(H3),(6), $s(F_1) = p - 1 \Rightarrow$ (7) $T(F_1, I) = T(F_1, I^*)$

(7), $F \equiv F_1 \Rightarrow$ (8) $T(F, I) = T(F, I^*)$

Sub-case $F' = F_1 \vee F_2$:

We will use notations similar to those in the proof of Theorem 2.

X : tuple of free variables of F
 X_1 : tuple of free variables of F_1
 Y_1 : tuple of unrestricted variables of F_1
 Z_1 : tuple of non unrestricted variables of F_1
 A : instance of X
 $A_1 = A[X_1]$ $B_1 = A[Y_1]$ $C_1 = A[Z_1]$

We have similar notations for $X_2, Y_2, Z_2, A_2, B_2,$ and $C_2,$ with respect to $F_2.$

Let x_i be a variable of Z_2 :

(1) \Rightarrow (3) $FRes(x_i, \neg(F_1 \vee F_2))$
(D4),(3) \Rightarrow (4) $FUnres(x_i, F_1 \vee F_2)$
(D4),(4) \Rightarrow (5) $FUnres(x_i, F_1)$ or $FUnres(x_i, F_2)$
 Z_2 definition \Rightarrow (6) $FUnres(x_i, F_2)$
(6) \Rightarrow (7) $FRes(x_i, \neg F_2)$

Until sentence (17) we assume $A \in D^n.$

From T definition we have :

(8) $T(F(A), I^*) = t \Leftrightarrow$ (9) $T(\neg F_1(B_1, C_1), I^*) = t$ and (10) $T(\neg F_2(B_2, C_2), I^*) = t$

(T1),(7),(10) \Rightarrow (11) C_1 does not contain *

(D4),(1) \Rightarrow (12) $\neg F_1(Y_1, C_1)$ evaluable

(H3),(12),(11), $s(\neg F_1) < p + 1$

\Rightarrow (13) $(T(\neg F_1(B_1, C_1), I^*) = t \Leftrightarrow T(\neg F_1(B_1, C_1), I) = t)$

In the same way we have :

(14) $(T(\neg F_2(B_2, C_2), I^*) = t) \Leftrightarrow T(\neg F_2(B_2, C_2), I) = t)$

(8),(13),(14) \Rightarrow

(15) $T(F(A), I^*) = t \Leftrightarrow T(\neg F_1(A_1), I) = t$ and $T(\neg F_2(A_2), I) = t$

(15) \Rightarrow (16) $(T(F(A), I^*) = t \Leftrightarrow T(F(A), I) = t)$

(16) \Rightarrow (17) $T(F, I^*) \cap D^n = T(F, I)$

(1),(L1),(17) \Rightarrow (18) $T(F, I^*) = T(F, I)$

Sub-case $F' = \exists x_{n+1} F_1(X, x_{n+1})$:

- (1) \Rightarrow (3) $\text{Pos}(x_{n+1}, F_1(X, x_{n+1}))$
(3) \Rightarrow (4) $F_1(A, x_{n+1})$ quasi-evaluable
(1) \Rightarrow (5) $\neg F_1(X, a)$ evaluable

Until sentence (19) we assume $A \in D^n$.

From T definition we have :

- (6) $T(F(A), I^*) = t \Leftrightarrow (7) \forall a \in D' T(\neg F_1(A, a), I^*) = t$
(7) $\Leftrightarrow (8) \forall a \in D T(\neg F_1(A, a), I^*) = t$ and (9) $T(\neg F_1(A, *), I^*) = t$
(H3),(5),s($\neg F_1$) = p \Rightarrow (10) $T(\neg F_1(A, a), I^*) = T(\neg F_1(A, a), I)$

Hence:

$$(8) \Leftrightarrow (11) \forall a \in D T(\neg F_1(A, a), I) = t$$

Therefore, from T definition :

$$(8) \Leftrightarrow (12) T(F(A), I) = t$$

Let's assume that (9) is false, i.e. : (13) $T(\neg F_1(A, *), I^*) = f$

- (13) \Rightarrow (14) $T(F_1(A, *), I^*) = t$
(T2),(4),(14) \Rightarrow (15) $\forall a \in D T(F_1(A, a), I^*) = t$
(15) \Rightarrow (16) $\forall a \in D T(\neg F_1(A, a), I^*) = f$
(16) \Rightarrow (17) $\exists a \in D T(\neg F_1(A, a), I^*) = f$

Since (17) contradicts (8), the assumption that (9) is false implies that (8) is false, then we have :

$$(8) \Rightarrow (9)$$

Therefore we have : (6) \Leftrightarrow (8) and (9) \Leftrightarrow (8) \Leftrightarrow (12)

Hence we have : (6) \Leftrightarrow (12), that is :

- (18) $T(F(A), I^*) = t \Leftrightarrow T(F(A), I) = t$
(18) \Rightarrow (19) $T(F, I^*) \cap D^n = T(F, I)$
(1),(L1),(19) $\Rightarrow T(F, I^*) = T(F, I)$

In the above sub-cases we have very similar proofs if F is closed formula.

Sub-case $F' =$ atomic formula. If F' is an open formula then $\neg F'$ cannot be evaluable and we do not have to consider this case. If F is a closed formula, from the definition of I^* , $T(F_1, I) = T(F_1, I^*)$, therefore $T(\neg F_1, I) = T(\neg F_1, I^*)$.

Basis of induction :

For $s(F)=0$ F is an atomic formula and from the definition of I^* , $T(F, I) = T(F, I^*)$.

5 Comparison with other classes of formulas

5.1 Comparison with Range Restricted formulas

Range Restricted formulas were introduced by J-M.Nicolas in [17] in order to characterize a subset of closed formulas in conjunctive prenex normal form, which are Domain Independent. We will extend this definition to open formulas and then to formulas in disjunctive prenex normal form.

Notations : Formulas in prenex normal form will be noted $F=QM$ where Q is the list of quantifiers and M the matrix. For formulas in conjunction normal form we will use the notation :

$$M = C_1 \wedge C_2 \wedge \dots \wedge C_n$$

$$C_i = L_{i1} \vee L_{i2} \vee \dots \vee L_{ip}$$

where L_{ij} is a positive atomic formula (form P_{ij}) or a negative atomic formula (form $\neg P_{ij}$).

Definition 8 (D8). Range Restricted Formulas in conjunctive prenex normal form.

A formula in conjunctive prenex normal form is a Range Restricted formula iff :

- for each free variable x there is a conjunct C_i such that all its literals L_{ij} contain x and are positive,
- for each existentially quantified variable x , if x occurs in a negative literal, then there is a conjunct C_i such that all its literals L_{ij} contain x and are positive,
- for each universally quantified variable x , if x occurs in a conjunct C_i , then one of its literals L_{ij} contains x and is negative.

Theorem 4 (T4)

If F is a formula in conjunctive prenex normal form then F is Range Restricted iff F is Evaluable.

Proof : Let F be a formula in conjunctive prenex normal form.

Case of free variables. Let x be a free variable of F .

$$\begin{aligned}
(D6) &\Rightarrow (F \text{ Evaluable} \Leftrightarrow FRes(x, F)) \\
(D4) &\Rightarrow (FRes(x, F) \Leftrightarrow FRes(x, M)) \\
(D4) &\Rightarrow (FRes(x, M) \Leftrightarrow \exists i FRes(x, C_i)) \\
(D4) &\Rightarrow (\exists i FRes(x, C_i) \Leftrightarrow \exists i \forall j FRes(x, L_{ij})) \\
(D4) &\Rightarrow (\exists i \forall j FRes(x, L_{ij}) \Leftrightarrow \exists i (\forall j L_{ij} \text{ is a positive literal} \\
&\quad \text{in which } x \text{ occurs}))
\end{aligned}$$

Therefore with respect to free variables : F Evaluable \Leftrightarrow F Range Restricted.

Case of existentially quantified variables. Let x be an existentially quantified variable of F . We assume F to be of the form $Q_1 \exists x Q_2 M$.

We will call C_1, C_2, \dots, C_m the C_i s in which x occurs, and C_{m+1}, \dots, C_n the C_i s in which x does not occur.

$$\begin{aligned}
(D6) &\Rightarrow (F \text{ Evaluable} \Leftrightarrow FPos(x, Q_2 M)) \\
(D5) &\Rightarrow (FPos(x, Q_2 M) \Leftrightarrow (1) FPos(x, M)) \\
(D5) &\Rightarrow ((1) \Leftrightarrow (2) FPos(x, C_1 \wedge C_2 \wedge \dots \wedge C_m)) \\
(D5) &\Rightarrow ((2) \Leftrightarrow ((3) \forall i \in [1, m] FPos(x, C_i) \text{ or} \\
&\quad (4) \exists i \in [1, m] FRes(x, C_i))) \\
(D5) &\Rightarrow (FPos(x, C_i) \Leftrightarrow \forall j (FPos(x, L_{ij}) \text{ or not Free}(x, L_{ij})))
\end{aligned}$$

Therefore :

$$(3) \Leftrightarrow (5) \forall i \in [1, m] x \text{ occurs only in positive literals of } C_i$$

By definition of m :

$$\begin{aligned}
(5) &\Leftrightarrow x \text{ occurs only in positive literals of } M \\
(D4) &\Rightarrow (4) \Leftrightarrow (6) \exists i \in [1, m] \forall j FRes(x, L_{ij}) \\
(D4) &\Rightarrow (6) \Leftrightarrow (7) \exists i \in [1, m] \forall j L_{ij} \text{ is a positive literal} \\
&\quad \text{in which } x \text{ occurs}
\end{aligned}$$

Therefore : (3) or (4) \Leftrightarrow (5) or (7) \Leftrightarrow

If x occurs in a negative literal then $\exists i \forall j L_{ij}$ is a positive literal in which x occur.

Therefore, with respect to existentially quantified variables :
 F Evaluable \Leftrightarrow F Range Restricted

Case of universally quantified variables. Let x be a universally quantified variable. We assume F to be of the form $F = Q'_1 \forall x Q'_2 M$.

The proof is very similar to the case of existentially quantified variables.

The definition (D7) can easily be transformed for formulas in disjunctive normal form. We will use the notations :

$$M = D_1 \vee D_2 \vee \dots \vee D_n$$

$$D_i = L_{i1} \wedge L_{i2} \wedge \dots \wedge L_{ip}$$

Definition 9 (D9). Range Restricted Formulas in disjunctive prenex normal form.

A formula in disjunctive prenex normal form is a Range Restricted formula iff :

- for each free variable x , for each disjunct D_i there is one of its literals L_{ij} which contains x and is positive,
- for each existentially quantified variable x , for each disjunct D_i , if x occurs in D_i , then there is at least one of its literal L_{ij} which contains x and is positive,
- for each universally quantified variable x , there is a disjunct D_i such that all its literals L_{ij} contain x and are negative.

Theorem 5 (T5).

If F is a formula in disjunctive prenex normal form then F is Range Restricted iff F is Evaluable.

Proof : the proof is quite similar to the proof of Theorem T4.

From theorems T4 and T5 we can conclude that in the set of formulas in conjunctive or disjunctive prenex normal form, the subset of Evaluable formulas is the same as the subset of Range Restricted formulas. However there are formulas of this form which are Domain Independent and which are neither Evaluable nor Range Restricted. For example the formula $\exists x(P(x) \vee \neg P(x))$.

If we call Prenex Normal Form the formulas in conjunctive or in disjunctive prenex normal form, the structure of these classe of formulas is represented by the relations :

$$\text{Range Restricted} \subset \text{Evaluable} \subset \text{Domain Independent}$$

$$\text{Range Restricted} = \text{Evaluable} \cap \text{Prenex Normal Form}$$

We may note that the definition of Range Restricted formulas cannot be used to recognize Domian Independent formulas which are not in normal form. Suppose we extend the definition of Range Restricted formulas as follows : “F is Range Restricted iff when F is transformed in prenex normal form, F satisfies D8 or D9”. Let’s call C the class of formulas having this property. This class is not solvable because a given formula F may have an infinite number of equivalent formulas which are in normal form; some of them may satisfy D8 or D9 and others formulas may not. Therefore definitions D8 and D9 can be used only for formulas which are in normal form. For example the formula :

$F = \exists x \exists y (P(y) \vee ((Q(x) \vee R(x)) \wedge \neg S(x)))$ yields the in disjunctive prenex normal form :

$F_{N_1} = \exists x \exists y (P(y) \vee (Q(x) \wedge \neg S(x)) \vee (R(x) \wedge \neg S(x)))$ and yields the in conjunctive normal form :

$F_{N_2} = \exists x \exists y ((P(y) \vee Q(x) \vee R(x)) \wedge (P(y) \vee \neg S(x)))$.

The formula F_{N_1} satisfies D9 but the formula F_{N_2} does not satisfy D8. Note that the formula F is evaluable.

5.2 Comparison with Permissible formulas

The definition of Permissible formulas introduced by E.C.Cooper in [7] is very close to the definition D8. However there are some differences which unfortunately allow it to encompass formulas which are not Domain Independent. The condition imposed on existentially quantified variables is the same as in definition D8, but the same condition is imposed on free variables. That is why a formula like $P(x) \vee Q(y)$ satisfies the definition although it is not Domain Independent.

The condition imposed on universally quantified variables is slightly different

from the definition D8. It can be expressed as follows :

“for each universally quantified variable x there exists i such that x occurs in D_i and for each j , if x occurs in L_{ij} then L_{ij} is a negative literal.”

This condition is satisfied by the formula $\forall x((\neg P(x) \wedge Q(a)) \vee R(x))$ which is not Domain Independent. Indeed in the interpretation where $Q(a)$ is false the formula is equivalent to $\forall x R(x)$ which is not Domain Independent.

5.3 Comparison with Range Separable formulas

E.F.Codd introduced in [6] the definition of Range Separable formulas for tuple Relational Calculus, but the translation from the tuple Relational Calculus to Predicate Calculus is straightforward and can be found in [20]. In the following we assume that Codd's definition is translated into Predicate Calculus.

In the first step “Proper Range formulas” are defined. They are formulas without quantifiers, with only one free variable, and either the \neg operator does not occur at all or it immediately follows an \wedge operator. Moreover comparison predicates $=, <, >, \dots$ are forbidden in these formulas. The consequence is that all free variables of a Proper Range formula are Restricted.

“Range Coupled Quantifiers” are then defined; these are of the form : $\exists x(\Gamma(x) \wedge \Delta(x))$ or $\forall x(\neg\Gamma(x) \vee \Delta(x))$, where Γ is a proper range formula. The consequence is that x is Restricted, and therefore Positive in $\Gamma(x) \wedge \Delta(x)$, and x is Unrestricted and therefore Negative in $\neg\Gamma(x) \vee \Delta(x)$. So, if all the quantifiers are range coupled all the quantified variables satisfy the conditions imposed in the definition of Evaluable formulas. Finally Range Separable formulas are of the form :

$$U_1 \wedge U_2 \wedge \dots \wedge U_n \wedge V$$

where all the U_i are proper range formulas and each variable of V occurs in an U_i ; moreover all the quantifiers in V are range coupled and all the predicates in V are comparison predicates, save in the case of proper range formulas which are used to restrict the range of quantified variables. The consequence is that each free variable of a Range Separable formula is Restricted because it is Restricted in one of the U_i . Therefore each Range Separable formula is Evaluable. The reverse is not true. There are Evaluable formulas, like $(P(x,y) \wedge Q(x)) \vee (R(x,y) \wedge S(y))$, which are not Range Separable.

5.4 Comparison with Predefined Variable Formulas

This class of formula was introduced by G.Bossu and P.Siegel in [4] to define a decidable procedure to deduce information from Data Bases where the information is represented with generalized Horn clauses.

Definition 10 (D10). Predefined Variable Formulas.

The Predefined Variable Formulas are defined by the following rules :

1. If A_1, A_2, \dots, A_n are atomic formulas with free variables x_1, x_2, \dots, x_p , then $\forall x_1 \forall x_2 \dots \forall x_p (\neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_n)$ is a Predefined Variable formula.
2. If F_1, F_2, \dots, F_n are Predefined Variable formulas then $\neg(F_1 \wedge F_2 \wedge \dots \wedge F_n)$ is a Predefined Variable formula.
3. If A_1, A_2, \dots, A_n are atomic formulas with free variables x_1, x_2, \dots, x_p and F_1, F_2, \dots, F_m are Predefined Variable formulas, then $\forall x_1 \forall x_2 \dots \forall x_p (\neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_n \vee \neg(F_1 \wedge F_2 \wedge \dots \wedge F_m))$ is a Predefined Variable formula.
4. All the Predefined Variable formulas are defined by the rules 1, 2 and 3.

It is easy to prove that each Predefined Variable formula is an Evaluable formula. Indeed formulas defined by rule 1 are Evaluable because all the variables are universally quantified and they are negative in the sub-formula which is in the range of the quantifier. The formulas defined by rule 2 are Evaluable if F_1, F_2, \dots, F_n are Evaluable. The formulas defined by the rule 3 are Evaluable if F_1, F_2, \dots, F_m , because each variable x_i is negative in $\neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_n$ and is negative too in $\neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_n \vee \neg(F_1 \wedge F_2 \wedge \dots \wedge F_m)$ are Evaluable because, F_1, F_2, \dots, F_n are closed formulas. So all the Predefined Variable formulas are Evaluable. However, the reverse is not true. For example the formulas $\forall x \neg(P(x) \wedge Q(x))$ and $\exists x P(x)$ are Evaluable but they are not Predefined Variable formulas.

6 Expressive power of Evaluable Formulas

We will show that Evaluable formulas have the same expressive power as Domain Independent formulas; that is, for each Domain Independent formula there is an equivalent Evaluable formula.

We have seen that Domain Independent formulas and Safe formulas have the same expressive power. J.D.Ullman has shown in [20] that Safe formulas and Relational Algebra have the same expressive power. So it is sufficient to prove that for each Relational Algebra expression there exists an equivalent Evaluable formula.

That can be done using the same proof as that of theorem 5-1 given by J.D.Ullman in [20], with Evaluable formulas playing the same role as Safe formulas. Indeed the only properties of Safe formulas which are used in this proof are the following :

1. Atomic formulas are Safe.
2. If $F_1(X)$ and $F_2(X)$ are Safe then $F_1(X) \vee F_2(X)$ is Safe.
3. If $F_1(X)$ and $F_2(X)$ are Safe then $F_1(X) \wedge \neg F_2(X)$ is Safe.
4. If $F_1(X)$ and $F_2(X)$ are Safe then $F_1(X) \wedge F_2(X)$ is Safe.
5. If $F_1(X)$ is Safe then $\exists x_i F_1(X)$ is Safe.
6. If $F_1(X)$ is Safe then $F_1(X) \wedge \sigma(x_{i_1}, x_{i_2}, \dots, x_{i_p})$ is Safe.

where $x_{i_1}, x_{i_2}, \dots, x_{i_p}$ are variables of X , and σ is a boolean formula in which the only predicates are comparison predicates.

We can easily check that these properties are also satisfied by Evaluable formulas. For example if $F_1(X)$ and $F_2(X)$ are Evaluable formulas then $F_1(X) \vee F_2(X)$ is also an Evaluable formula.

7 Conclusion

We have defined a class of formulas which have been proved to be Domain Independent and we have shown that this class is larger than the previous ones encountered in the literature. Moreover, we have proved that this class of formulas has the same expressive power as the Domain Independent formulas.

It is interesting to note that the definition of Evaluable formulas does not require the formulas to be put in a normal form. The implementation of a procedure which recognizes Evaluable formulas has been implemented in PROLOG.

The program is very simple and fast, and is close to the definitions given in this paper.

We think that it is possible to find a larger syntactic characterization of Domain Independent formulas, but such a characterization would be more complex. In particular, we could distinguish different cases of \vee operator usage in the formulas. In many cases the operands of an \vee operator are not linked to one another; that is the reason why a formula like $P(x) \vee Q(y)$ is not Domain Independent. If there an x value such that $P(x)$ is true then y can take any value of the domain. However, in some cases the \vee has the same meaning as an \wedge . Consider for example the formula $\exists t R(x, t) \wedge \forall z (\neg R(x, z) \vee S(z, y))$. If we denote by R_x the set $\{z \mid R(x, z)\}$, and by S_y the set $\{z \mid S(z, y)\}$, then the meaning of this formula can be expressed informally by $\exists t R(x, t) \wedge (R_x \subseteq S_y)$, which shows that the variables x and y which appear in the operands of the \vee are linked to one another as in the case of the operands of an \wedge . This kind of formula correspond to what we have called “generalised division” in the Relational Algebra [9]. On the other hand we could try to consider tautologies and contradictions in the definition. Indeed formulas like $\exists x (P(x) \vee \neg P(x))$ or $\forall x (P(x) \wedge \neg P(x))$ are Domain Independent but are not Evaluable. However it is well known that the problem of recognizing if a formula is a tautology or a contradiction is not decidable, so we cannot hope to extend the definition significantly in this direction.

Our feeling is that it is not easy to find a simple and significant extension to Evaluable formulas because there are relatively few formulas which are equivalent to an Evaluable formula and which are not Evaluable. Indeed we have shown in [10] that, if we consider equivalent formulas obtained from a given formula using the transformation rules corresponding to factorization or ditribution of the operators \vee , \wedge , \neg , \forall or \exists , and associativity and commutativity of the operators \wedge and \vee , then a variable’s property of being Restricted or Unrestricted is preserved. Furthermore, a variable’s property of being Positive or Negative is always preserved except in one very particular case. This case, for a Positive variable, is that of transforming $F_1 \vee (F_2 \wedge F_3)$ into $(F_1 \vee F_2) \wedge (F_1 \vee F_3)$ when the following condition holds :

$$((FPos(x, F_1) \wedge \neg FRes(x, F_1)) \vee \neg Free(x, F_1)) \wedge Free(x, F_2) \wedge Free(x, F_3) \wedge ((FRes(x, F_2) \wedge \neg FPos(x, F_3)) \vee (\neg FPos(x, F_2) \wedge FRes(x, F_3))).$$

For example x is Positive in $(P(x) \vee Q(a)) \vee (R(x) \wedge \neg S(x))$ but it is not Positive in $(P(x) \vee Q(a) \vee R(x)) \wedge (P(x) \vee Q(a) \vee \neg S(x))$. There is a similar case for Negative variables.

A possible extension to this work could be to consider a language with equality, or to consider specific problems arising with comparison predicates.

Finally one could suggest looking for a shorter proof for Theorem 3 using the Relativization theorem [5](5-2-20); however to apply the Relativization theorem we should need to prove that Evaluable formulas are equivalent to their relativized transform, and this proof would be as long as the proof in section 4.

Acknowledgements : we would like to thank the referee whose numerous and sound comments and suggestions were of great importance to the quality of this paper. Thanks also to Andrew Jones for improving the english of the paper.

References

- [1] A. Artaud and J-M. Nicolas. Systeme d'interrogation experimental SYNTAX. Technical report, ONERA-CERT, 1972.
- [2] A. Artaud and J-M. Nicolas. An experimental query system :SYNTAX. In *Proc. of International Computing Symposium*. North-Holland, 1974.
- [3] P. Besnard. *Une procedure de décision en logique non-Monotone*. PhD thesis, Université de Rennes, France, 1983.
- [4] G. Bossu and P. Siegel. *La saturation au secours de la non-Monotonie*. PhD thesis, Université d'Aix-Marseille, France, 1981.
- [5] C. C. Chang and H. J. Keisler. *Model Theory*. North-Holland, 1973.
- [6] E. F. Codd. Relational Completeness of Data Base Sublanguages. In R. Rustin, editor, *Data Base Systems, Courant Computer Science Symposium 6*, pages 65–98. Prentice-Hall, 1972.
- [7] E. Cooper. On the expressive power of query languages for Relational Databases. Technical Report TR-14-80, Aiken Computation Laboratory, 1980.
- [8] R. Demolombe. Assigning meaning to ill-defined queries expressed in Predicate Calculus language. In H. Gallaire, J. Minker, and J-M. Nicolas, editors, *Advances in Data Base Theory, Voll.* Plenum-Press, 1981.
- [9] R. Demolombe. Generalized division for Relational Algebraic language. *Information Processing Letters*, 14(4), 1982.
- [10] R. Demolombe. Utilisation du Calcul des Prédicats comme langage d'interrogation des Bases de Données. Thèse d'Etat, Université Paul Sabatier, Toulouse, 1982.

- [11] R. Demolombe, M. Lemaitre, and J-M. Nicolas. The language of SYNTEX-2, an experimental relational like DBMS. In Moneta, editor, *Proc. of Jerusalem Conference on Information Technology*. North-Holland, 1978.
- [12] R. Fagin. Horn clauses and Database dependencies. In *Proc. of ACM Symposium on Theory of Computing*, 1980.
- [13] H. Gallaire and J. Minker. *Logic and Data Bases*. Plenum Press, 1978.
- [14] H. Gallaire, J. Minker, and J-M. Nicolas. *Advances in Data Base Theory, Vol1*. Plenum Press, 1981.
- [15] J. L. Kuhns. Interrogating a relational data file. Technical Report R-511-PR, Rand Corporation, 1970.
- [16] J. A. Makowski. Characterizing Data Base Dependencies. In *Lecture Notes in Computer Science, Vol115*, 1981.
- [17] J-M. Nicolas. Logic for improving integrity checking in relational data bases. *Acta Informatica*, Vol 18(Num 3), 1982.
- [18] J-M. Nicolas and R. Demolombe. On the stability of Relational queries. In *Proc. of Workshop : Logical Bases for Data Bases*, 1982.
- [19] R. A. Di Paola. The recursive unsolvability of the decision problem for the class of Definite Formulas. *Journal of ACM*, 16(2), 1969.
- [20] J. D. Ullman. *Principles of Database Systems*. Computer Science Press, 1980.
- [21] M. Zloof. Query-by-example. In *Proc. of AFIPS Vol4*, 1975.