SPECIFICATION OF INFORMATION SYSTEMS BY STATE MACHINE DEDUCED FROM DEDUCTIVE CONCEPTUAL MODELS

Villena, Salvador; Clares, Buenaventura

Dpto. Lenguajes y Sistemas Informáticos
Facultad de Ciencias Universidad de Granada
Campus Universitario Fuentenueva s/n
(18071) Granada (Spain)
Tf: 58-243177; Fax: 58-243179
Mail: svillena@ugr.es

Abstract. We propose a transformation from declarative approach to conceptual information system modelling onto an operational implementation. We will introduce an alternative way to obtain the minimum model of the Herbrand associated with a logic program. This process provides a new expression of a logic program as a finite state machine. This process is then extended to apply over the specification of the information systems by deductive conceptual models.

1.Introduction.

The formal specification of Information Systems (IS) by Deductive Conceptual Models (DCM) has been widely studied [OLL82] [OLL83] [OLI85] [SIS87] [SAN90]. wThe specification by DCM is constituted of only logic components and the behavior of systems is specified by points in time. All possible information in the systems is associated with a time point. This time point is the instant at which it happene, it is a component of its own information.

The implementation of DCM on which J.Sistac and M.R. Sancho have worked [SIS87] [SAN90], has been its biggest drawback.

Our work is directed towards the transformation to an operational approach which Olivé proposed as an efficient form of implementation [OLI85].

We assume the first order logic language as a specification language of DCM.

Firstly, we have developed a method to translate from a logic program, which expressed in a relational language, to this program expressed as a finite state machine. After, we will extend this method to apply a specification of the IS by DCM.

Our method is based on a generalization of the Lloyd method [LLO87A] for cheking integrity constraint in deductive databases.

This generalization is provided by two concepts: the generic fact, and the more general closed substitution. The generic fact propose an abstraction of the concept of fact. The concept of the more general closed substitution is an application over the arguments of the logic formulas to equal some parts of

these formulas. This process is a syntactic manipulation without changing their semantic.

The method proposes a alternative way to obtain the minimum model of the Herbrand associated with a logic program. This model is obtained as a sequence of the transactional operations.

To finish, we propose the form to deduce from a DCM of the IS, to a logic program. This program is expressed in a relational language such that the time is included as argument in all predicates.

The concept of the temporary normalized is included to apply this method over the specification by DCM.

This paper is structured as follows. Firstly, we propose the translations from logic program to state machines, and, then, the extension of this process over the specification of IS by DCM.

2. Potential Effects Method over Logic Program.

The potential effects method (PEM) propose a way to translate from a logic program to a set of transactional operations, which it applied over the facts in this logic program itself get the model of this logic program. That is to say, the PEM provide other way to perfom the model and the interpretation of a logic program.

In this secction, the PEM is introduced. Firstly, we include the frame work and previous concepts, next, the new concepts used by PEM it are developed.

2.1 Frame Work and Previous Concepts.

We will consider a relational language $\mathbf{L}=(\mathbb{A},\mathbb{F})$, where $\mathbb{A}=\mathbf{A} \cup \mathbf{P} \cup \mathbf{V} \cup \mathbf{S}$ and $\mathbb{F}$ is the set of the well formed formulas (wff) in **L**, where **A** is the set of constants, **P** the set of predicates, **V** the set of variables, and **S** the set of the symbols in **L**. A relational interpretation $\mathbf{I}=(D,I_c,I_v)$ is defined over this language, where D is a semantic domain, I_c is a set of the applications suches that if p is predicate in **P**, with arity n, then $I_c(p):D^n \longrightarrow \{T,F\}$, if a is a member in **A**, then $I_c(a) \in D$, and I_v is an application $I_v:\mathbf{V} \longrightarrow D$.

Further, a deductive system K formed by a set of the axioms and inference rules [THY88] will be assumed.

The concepts of atom, basic atom, and literal are assumed as in [THY88].

Definition 1.- let **L** be a relational language, and $\mathcal{T}$ a deductive theory in **L**, then a triplet (l,c,l') is a *potential element* if:

a) Let l and l' be literals in **L**, c is a conjunction of the literals in **L**,then two literals formed by the same atomic formula but with sign changed cannot appear in the triplet (l,c,l'), and

b) $\vdash(l \wedge c) \Rightarrow \vdash l'$ is valid in $\mathcal{T}$.

A potential element (l,c,l') is a *potential attandence* if l' is a positive literal, and a potential element is a *potential absence* if l' is a negative

literal. ■

Let I be a relational interpretation in **L**, then l and c are necessary pre-condition in order that l' be true, with independence of the sign of l. The (b) condition provide this characteristic.

Definition 2.- Let **L** be a relational language, and $\mathcal{T}$ a deductive theory in **L**. A *potential chain over* l_1 *literal* is a set:

$$G(l_1) = \{(l_1,c_1,\{l_2\}),\ (l_1,c_1\wedge c_2,\{l_2,l_3\}),\ \ldots$$
$$\ldots,\ (l_1,c_1\wedge c_2\wedge \ldots \wedge c_n,\{l_2,l_3,\ \ldots\ ,l_{n+1}\})$$

if $(l_1,c_1\wedge c_2\wedge \ldots \wedge c_i,\{l_2,l_3,\ \ldots,l_{i+1}\})\in G(l_1)$, for $i=1,\ldots,n$, then $\vdash(l_1\wedge c_1\wedge c_2\wedge \ldots \wedge c_i) \Rightarrow \vdash(l_2\wedge l_3\wedge \ldots \wedge l_{i+1})$ is valid in $\mathcal{T}$, where $l_1,l_2,\ \ldots\ ,l_{n+1}$ are positive or negative literals, and $c_1,c_2,\ \ldots\ ,c_n$ are conjunction of literals. ■

Evidently, a literal l can be associated with one or more potential chains. The set formed by all the potential chains associated with the literal l, will be denoted as G(l).

Let c be a conjunction of literals in **L**, {c} expresses the set of literals which form c. The members in {c} are its sign.

Definition 3.- Let c_1 and c_2 be two conjunctions of literals in **L**, let $\{c_1\}$ and $\{c_2\}$ be these sets of literals associated with these conjunctions. We can say that c_1 is included in c_2, which we note as $c_1 c: c_2$, if $\{c_1\}c\{c_2\}$. ■

It becomes evident to show that the relation c: establishes a partial order over the set of all conjunctions of literals in **L**. It becomes evident from the definition (2) to prove that the relation c: provides a total order over the set of any potential chain G(l). It is enough to apply the relation c: over these sets of literals which are associated with a second component of each member in G(l).

Definition 4.- let **L** be a language relational, $\Gamma(x_1,x_2,\ \ldots,\ x_n,u_1,u_2,\ \ldots,u_k)$ and $\Phi(y_1,y_2,\ \ldots,y_n,z_1,z_2,\ \ldots,z_m)$ are wffs, such that at least one literal is in both formulas which is formed by the same predicate $p\in\mathbf{P}$ and with the same number of arguments. We will denote $x=(x_1,\ \ldots\ ,x_n)$, $u=(u_1,\ \ldots\ ,u_k)$, $y=(y_1,\ \ldots\ ,y_n)$, $z=(z_1,\ \ldots\ ,z_m)$, and we will assume that the predicate p appears over the variable x, in the i-th literal of the Γ, and the predicate p appears over the variable y in the j-th literal of the Φ. We will call *ij-more general closed substitution (mgcs) over* p as an application:

$$\theta_p^{ij}[\Gamma(x,u),\Phi(y,z)]:\{x_1,\ \ldots\ ,x_n,u_1,\ \ldots\ ,u_k,y_1,\ \ldots\ ,y_n,z_1,\ \ldots\ ,z_m\} \longrightarrow \{x_1,\ \ldots\ ,x_n,u_1,\ \ldots\ ,u_k,y_1,\ \ldots\ ,y_n,z_1,\ \ldots\ ,z_m\},$$

it must accomplish:

a) $x_1 = \theta_p^{ij}[\Gamma(x,u),\Phi(y,z)](y_1), \ldots, x_n = \theta_p^{ij}[\Gamma(x,u),\Phi(y,z)](y_n)$.

b) $\vdash (\Phi(y_1, \ldots, y_n, z_1, \ldots, z_m)) \Rightarrow \Phi(\theta_p^{ij}[\Gamma(x,u),\Phi(y,z)](y_1), \ldots, \theta_p^{ij}[\Gamma(x,u),\Phi(y,z)](y_n), \theta_p^{ij}[\Gamma(x,u),\Phi(y,z)](z_1), \ldots, \theta_p^{ij}[\Gamma(x,u),\Phi(y,z)](z_m))$. ■

From now on, the explicit reference of i, j position will be omitted when the mgcs is applied over the first occurrences in both formulas. Also the predicate will be omitted when there is no confusion in the context.

Henceforth, let $\Gamma(x_1,x_2, \ldots, x_n)$ and $\Delta(y_1,y_2, \ldots, y_m)$ be two wffs, and θ a mgcs over these formulas; we will note the formula $\Delta(\theta[\Gamma(x),\Delta(y)](y_1), \ldots, \theta[\Gamma(x),\Delta(y)](y_m))$ by the expression $\Delta(y)\theta[\Gamma(x),\Delta(y)]$. And if there is no confusion, then we will note $\Delta(y)\theta$, and we will understand θ as $\theta[\Gamma(x),\Delta(y)]$.

EXAMPLE.- Let $p(x,y,z)$ and $p(z,t,u)\wedge\Phi(y,z)$ be two wffs, where $x,y,z,t,u\in V$. Then two cmgs will have over these wffs:

a) θ defined as: $x \longrightarrow t$, $y \longrightarrow u$, $z \longrightarrow x$, $t \longrightarrow y$, $u \longrightarrow z$

b) θ' defined as: $x \longrightarrow u$, $y \longrightarrow t$, $z \longrightarrow x$, $t \longrightarrow y$, $u \longrightarrow z$

Where θ and θ' satisfy the (a) and (b) conditions of the definition 4. .

Theorem 1.- let **L** be a relational language, $\Gamma(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_k)$ and $\Phi(y_1, y_2, \ldots, y_n, z_1, z_2, \ldots, z_m)$ are two wffs, such that at least one literal exists in both formulas and these literals are formed by a predicate $p\in P$ with arity n. If the predicate p appears over the variables $(x_1, x_2, \ldots, x_n)$ in the i-esimo literal of the Γ, and the predicate p appears over the variables $(y_1, y_2, \ldots, y_n)$ in the j-th literal of Φ, then a *ij-more general closed substitute (mgcs) over* p between both wffs can always be defined.

Proof. We construct the $\theta_p^{ij}[\Gamma,\Phi]$ with the following form:

a) Le the variables x_i be

$$x_1 = \theta_p^{ij}[\Gamma,\Phi](y_1), \ldots, x_n = \theta_p^{ij}[\Gamma,\Phi](y_n)$$

b) For the rest of the variables the application is defined with the following form. Let x' be a set which is defined as $x'=x\cap(y\cup z)$,

b.1) If $x'=\emptyset$, then $z_1 = \theta_p^{ij}[\Gamma,\Phi](z_1), \ldots, z_m = \theta_p^{ij}[\Gamma,\Phi](z_m)$.

b.2) If $x'= \{x_f, \ldots, x_g\}$, then $y_f=\theta_p^{ij}[\Gamma,\Phi](x_f), \ldots, y_g= \theta_p^{ij}[\Gamma,\Phi](x_g)$, and the rest of the variables are defined as in (b.1).

It is evident that $\theta_p^{ij}[\Gamma,\Phi]$ accomplishes the (a) and (b) conditions in the

definition. ■

Let p(x) and p(y)∧Φ(z) be two wffs, where $x,y \in \mathbf{V}^n$, $z \in \mathbf{V}^m$, and $\theta_1[p(x),p(y)\wedge\Phi(z)]$ is a mgcs such that $p(x)=p(y)\theta_1$, that is to say, $p(x)=p(y)\theta_1$. In general, it is obvious that $p(x)\theta_1 \neq p(y)\theta_1$, and we can define an other mgcs as $\theta_2[p(x)\theta_1,(p(y)\wedge\Phi(z))\theta_1]$. This process can be repeated a number of times, and the set $\Theta=\{\theta_1,\theta_1,\ldots\}$ formed by mgcss will be obtained.

Theorem 2.- Let p(x) and p(y)∧Φ(z) be two wffs in L, where $p \in \mathbf{P}$, $x,y \in \mathbf{V}^n$, $z \in \mathbf{V}^m$, and $\Theta=\{\theta_1\theta_2, \ldots\}$ is a set of the mgcss, which has been obtained by the above process. Then the Θ is a finite set.

Proof.- It is evident since the mgcs is defined over a finite set of the variables.

Notice that, the definition (3) can be extended tanking account the theorem (1). A more general relation of inclusion is proposed. Let c_1 and c_2 be two conjunctions of the literal, We can say that c_2 is included with respect to a mgcs Θ in c_1, if $\Theta[c_1,c_2]$ exists, and $c_2\Theta \subset c_1$. This relation is denoted as $c_2 \subset_\Theta c_1$.

It is evident that Θ can be an identity application.

Definition 5.- let L be a relational language, $\Gamma(x_1,x_2, \ldots ,x_n)$, $\Delta(y_1,y_2, \ldots ,y_m)$ are two wffs in **L**, and $\mathcal{P}$ the set of predicates in both formulas, such that $\mathcal{P}\neq\emptyset$. Let γ_p and δ_p be the numbers of occurrences of p in Γ y Δ, for each p in $\mathcal{P}$, and $k=\min\{\gamma_p,\delta_p\}$.Then the application $\Xi[\Gamma(x),\Delta(y)]:\{x_1, \ldots ,x_n,y_1, \ldots ,y_m\} \longrightarrow \{x_1, \ldots ,x_n,y_1, \ldots ,y_m\}$ is a *more general global closed substitute (mggcs)* if it is accomplished:

a) $\forall p \in \mathcal{P}$, Ξ makes the variables in the first k occurrences of p in Δ equal to the first k occurrences of p in Γ, and

b) $\vdash (\Delta(y_1, \ldots ,y_m) \Longrightarrow \Delta(y_1, \ldots ,y_m)\Xi[\Gamma(x),\Delta(y)])$. ■

Let Γ(x) and Δ(y) be two wffs in L, then a gmgcs Ξ over both formulas cannot always be found.

Definition 6.- A *logic specification* or *logic program* is defined as a set of clausulas of Horn with or without negation in the body. We will denote this as S.

Definition 7.- Let S be a program, if a predicate appears in the head of a clausula such that this clausula does not have an empty body, then this predicate is called *derivate predicate*. The set of derivate predicates in S will be denoted as $\mathbf{P}_D$.

If a predicate appears in a base atom or in some body of a clausula, or in the head of a clausula with an empty body, and further, it is not a derivate predicate, then this predicate is a *basic predicate*. The set of the basic predicates in S will be denoted as $\mathbf{P}_B$.

The relational predicates (=,<,>,..) can appear in the body of the sentences in the program S. These predicates are neither derivates nor basic. Further these predicates are not important in our work. From now on they will not be taken into consideration. When we refer to the predicates of a program, the relational predicates are not included. ■

Let $\mathbf{P}_S$ be the set of predicates in a program S, then $\mathbf{P}_D \cap \mathbf{P}_B = \varnothing$, in virtue of definition (7), and $\mathbf{P}_D \cup \mathbf{P}_B \subseteq \mathbf{P}_S$. ■

Definition 8.- Let S be a logic program. If a basic atom appears in the head of a clausula with empty body, then this base atom is a *fact*. ■

Definition 9.- Let S be a logic program, a *generic fact* is a class of the atomic formulas formed by a basic predicate and its arguments are any of the variables. ■

The generic facts of the a program S can be obtainned from the following form:

a) From the basic predicates which appear in the body of the clausulas of S. When an atom formed by a basic predicate is found in the body of a clausula, then we change this arguments by variables. In this way, a set of atoms is formed. All the members of this set have a common predicate.

b) From the facts which appears in the program S. When the arguments of a fact aredefined in $\mathbf{A}^n$ we change them for other arguments which are defined in $\mathbf{V}^n$. In this way, we obtain a set of atoms such that all the members in this set have a common predicate.

c) Let A_S be the set of the atomic formulas, which results in the processes (a) and (b), and let R be an equivalence relation over the members in A_S, such that two elements are equivalent if these members are formed with the same predicate. Then each equivalence class of the A_S/R is represented by a generic fact. ■

Definition 10.- A logic program is *pure*, if all facts in this program are formed with basic predicates. ■

An equivalent pure program can be found for all programs, and it is easy to prove.

From now on, we will suppose that the arguments of the atoms are formed by variables only, for all atoms in the body of any clausula of a program.

In a program all sentences $p(x) \Leftarrow q(x,a) \wedge \phi(z)$, can be rewritten as $p(x) \Leftarrow q(x,y) \wedge \phi(z) \wedge (y=a)$.

In all logic programs we distinguish two parts. One is formed by a set of the clausulas with head and body not empty, which is nominated as *the kernel* of the logic program. The other is formed by a set of facts which is nominates as the *set of facts* of the logic program.

From now on, we use the following concepts: the hierarchy and stratified logic program, allowed formula, and allowed program, as proposed by Apt [APT88]. Also we use the concept of the normalized program proposed by Lloyd and Topor [LLO84] [LLO87].

Definition 11.- Let S be a logic program. We can say that a clausula in S is *well constructed*, if the body is not empty and if, in the body of the clausula, all the literal have one or more common argument in the head of this clausula. ■

Definition 12.- A logic program S is *well constructed* if all the clausulas in the kernel of S are well constructed. ■

From now on, when we make reference to a *program*, then we are considering a pure, well constructed, hierarchy or stratified, allowed, and normalized logic program.

Let S be a program, h(x) a generic fact, and G(h(x)) the set formed by all potential chains with respect to h(X). We define a *level* as the specific position of the members in the potential chain of G(h(x)). For example the zero level is formed by the first elements of all the potential chains in G(h(x)).

2.2 The Set of Potential attandence and absence of a Logic Program.

We now consider a subset of G(h(x)) which it is formed by these potential chains in the zero level, such that the first component of the members in this potential chain can be equalized to h(x) by means of some mgcs.

Definition 13.- Let S be a program, and h(x) is a generic fact in S. We define the *set of the potential attandences for* h(x) *associated with* S *at zero level* as:

$$E_+^0(h(x)) = \{(h(x),\Phi(x_1)\theta,p(y)\theta) \mid \exists(p(y)\Leftarrow h(x_2)\wedge\Phi(x_1))\in S,\ \theta[h(x),h(x_2)\wedge\Phi(x_1)] \text{ is mgcs}\},$$

where p is a predicate and $\Phi(x_1)$ is a conjunction of the literals. ■

Notice that in the deductive theory associated with S, a member in E_+^0 is an other way to express the sentences of S, and if h(x) and $\Phi(x_1)\theta$ are valid on this theory, then $p(y)\theta$ is valid too.

Definition.14.- Let S be a program, and h(x) a generic fact in S. We define the *set of the potential absences for* h(x) *associated with* S *at zero level* as:

$$E_-^0(h(x)) = \{(h(x),C\Theta,\neg p'(y')\Theta) \mid \exists(p'(y')\Leftarrow \neg h(x_2')\wedge\Phi'(x_1'))\in S,$$

$$C=\wedge_{i=1}^{m}\neg\Phi_i'(x_i'')\theta_i, \text{ if } \exists(p'(y_i')\Leftarrow \Phi_i'(x_i''))\in S,\ i=1,\ldots,m, \text{ and}$$

$$\theta_i[p'(y')\Theta,(p'(y_i')\Leftarrow.\Phi_i'(x_i''))], \text{ else } C=T \text{ (True), and}$$

$$\Theta[h(x),\neg h(x_2')\wedge\Phi'(x_1')] \text{ is mgcs}\},$$

where p' is a predicate, and Φ'and Φ'_i are a conjunction of the literals. ■

Under the supposition of the Clark's complement, a member in E^0_- expresses when the p'((y')Θ is not valid, or else it cannot be deduced in the theory associated with *S*.

Definition 15.- Let *S* be a program, and h(x) a generic fact in *S*. We define the *set of potential attandences for* h(x) *associated with S at* (n+1) *level* as:

$$E^{n+1}_+(h(x)) = \{(q(z),\Delta(z_1)\theta',p'(t)\theta') \mid \exists(p'(t)\Leftarrow q(z_2)\wedge\Delta(z_1))\in S,$$
$$\theta'[q(z),q(z_2)\wedge\Delta(z_1)] \text{ is mgcs, and}$$
$$\exists(l(s),\Gamma(u),q(z))\in E^n_+(h(x))\}$$
$$\cup \{(\neg q'(v),\Upsilon(v_1)\theta'',p''(w)\theta'' \mid \exists(p''(w)\Leftarrow \neg q'(v_2)\wedge\Upsilon(v_1))\in S,$$
$$\theta''[\neg q'(v),\neg q'(v_2)\wedge\Upsilon(v_1)] \text{ is mgcs, and}$$
$$\exists(l'(s_1),\Psi(u_1),\neg q'(v))\in E^n_-(h(x))\},$$

where l and l' represent the positive or negative literals, q, q',p', and p'' represent the predicates, and Δ, Γ, Υ, and Ψ represent conjunctions of the literals. ■

Definition 16.- Let *S* be a program and h(x) a generic fact in *S*. We can say what this fact has a *maximum level of the potential attandences* m if, in the sets of potential attandences for h(x) at levels higher than m, elements only appear which are already included in the sets of lower levels. ■

Definition 17.- Let *S* be a program, and h(x) a generic fact in S. We can define the *set of potential attandences with respect to the generic fact* h(x), as:

$$E_+(h(x)) = \bigcup_{i=0}^{m} E^i_+(h(x)),$$

where m is the maximum level of potential attandences of h(x). ■

Definition 18.- Let *S* be a program, and h(x) a generic fact in *S*. We define the *set of potential absences for* h(x) *associated with S at* (n+1) *level* as:

$$E^{n+1}_-(h(x)) = \{(q(z),\Psi(v)\theta',\neg p'(t)\theta') \mid \exists(p'(t)\Leftarrow \neg q(z_2)\wedge\Delta(z_1))\in S,$$
$$\Psi(v)=\bigwedge_{i=1}^{m}\neg\Phi_i(v_i)\theta_i \quad y \quad \theta_i[p'(t)\theta',(p'(t_i)\Leftarrow\Phi_i(v_i))] \quad \text{if}$$
$$\exists(p'(t_i)\Leftarrow\Phi_i(v_i))\in S,\ i=1,\ldots,m,$$
$$\text{or} \quad \Psi=T \text{ (true)} \quad \text{if} \quad \neg(\exists(p'(t_i)\Leftarrow\Phi_i(v_i))\in S),\ i=1,\ldots,m,$$
$$\theta'[q(z),\neg(q(z_2)\wedge\Delta(z_1))] \text{ is mgcs, and}$$
$$\exists(l(s),\Gamma(u),q(z))\in E^n_+(h(x))\}$$
$$\cup \{(\neg q'(z'),\Upsilon(v')\theta'',\neg p''(t')\theta'') \mid \exists(p''(t')\Leftarrow q'(z'_2)\wedge\Delta'(z'_1))\in S,$$
$$\Upsilon(v')=\bigwedge_{i=1}^{m}\neg\Phi_i(v'_i)\theta'_i \quad y \quad \theta'_i[p''(t')\theta'',(p''(t'_i)\Leftarrow\Phi_i(v'_i))]$$
$$\text{if} \quad \exists(p''(t'_i)\Leftarrow\Phi_i(v'_i))\in S,\ i=1,\ldots,m,$$
$$\text{or} \quad \Upsilon=T \text{ (true)} \quad \text{if} \quad \neg(\exists(p''(t'_i)\Leftarrow\Phi_i(v'_i))\in S),\ i=1,\ldots,m,$$

$\theta''[\neg q'(z'),(q'(z'_2)\wedge\Delta'(z'_1))]$ is mgcs, and

$\exists(l'(s'),\Gamma'(u'),\neg q'(z'))\in E_-^n(h(x))\}$,

Where l and l' are positive or negative literals, q, q', p', and p'' are predicates, and Δ, Δ',Γ, Γ',Ψ, and Υ represent the conjunctions of the literals. ■

Similarly, the *maximum level of the potential absences* is defined as the definition (16).

Definition 19.- Let *S* be a program, and h(x) a generic fact in *S*. We define the *set of potential absences with respect to the generic fact* h(x), as:

$$E_-(h(x)) = \bigcup_{i=0}^{n} E_-^i(h(x)),$$

where n is the maximum level of the potential absences for h(x). ■

Definition 20.- We can say that a generic fact h(x) has got a *maximum level* n in a program *S*, if $n = \max\{n_1,n_2\}$, where n_1 is the maximum level of potential attandences for of h(x) in *S*, and n_2 is the maximum level of the potential absences of h(x). ■

Definition 21.- Let *S* be a program, and h(x) a generic fact in *S*. We define the *set of the potential attandences and absences associated with respect to* h(x) associated with *S*, as a set

$$E(h(x)) = E_+(h(x)) \cup E_-(h(x)).$$ ■

Definition 22.- Let *S* be a program, we can define the *set of the potential attandences and absences associated with S* as

$$\mathbf{E} = \bigcup_{i=1}^{n'} E(h_i(x)),$$

where n' is the number of the generic facts which are in *S*. ■

Theorem 3.- Let *S* be a pure, well constructed, hierarchy or stratified, and allowed program. For any generic facts in *S*, then the set of potential attandences or absences associated with each generic fact have a finite maximum level.

Proof.- First, we prove this theorem when *S* is a hierarchy program, that is to say, the program *S* is not recursive.

If *S* is not recursive, then a finite set of the clausulas exists, such that its head is formed by predicates which are not in the body of any clausulas in *S*. It is evident, which the members (l,c,p(x)) or (l,c,¬p(x)) are attained at a finite level such that of p is a predicate which is not in the body of the clausulas in *S*.

If *S* is stratified, that is to say, the positive literals can be defined recursively in the same clausula, the following circumstances can occur in the calculation of the potential attandence or absence:

a) If the clausula $p(x)\Leftarrow(\Phi(z)\wedge p(u))\wedge q(r)$ is in S, when we are calculating the potential attandence to a generic fact at level n, and the (l,c,q(t)) appears at level (n-1), then this potential attandence will induce $(q(t),(\Phi(z)\wedge p(u))\theta,p(x)\theta)$ at level n. We denote this potential attandence as $(q(t),(\Phi(z')\wedge p(u')),p(x'))$. This potential attandence will induce $(p(x'),(\Phi(z')\wedge q(t))\theta',p(x')\theta')$ at the level (n+1), This process can be repeated in the levels (n+2), (n+3),..., in virtue of the theorem (2), the number of levels are finite.

A Similar reasoning can be applied to the calculation of the potential absence in level (n-1).

b) If the clausula $p(x)\Leftarrow(\Phi(z)\wedge\neg p(u))\wedge q(r)$ is in S, when we are calculating the potential attandence to a generic fact in level n, and the (i,c,q(r)) appears at the level (n-1), this will induce $(q(t), (\Phi(z)\wedge\neg p(u))\theta, p(x)\theta)$, at level n. We denote this potential attandence as $(q(t), (\Phi(z')\wedge\neg p(u'), p(x'))$. This potential attandence will induce the potential absence $(p(x'),C(s)\theta',\neg p(x')\theta')$ at level (n+1), and this potential absence will induce an other potential attandence at level (n+2). This process can be repeated at consecutive levels, in vitue of the theorem (2) this process is finite.

When one or more clausulas are defined recursively, the same reasoning is applied.

Any other circumstances can be decomposed in the event of (a) and (b). ■

2.3 Operations associate to a Generic Facts.

In this secction, we will defined the concept of the operation.

Definition 23.- Let S be a program, $\mathcal{T}$ is the theory associated to S, $\mathbf{AP}_D$ is the set of atoms which are formed by derivate predicates in S, and $\mathcal{P}(\mathbf{AP}_D)$ is the set of parts in $\mathbf{AP}_D$. We can define an *operation* as $(h(x),c,\mathcal{A},\mathcal{B})$, where $h(x)$ is a generic fact in S, c is a conjunction of the literals, and $\mathcal{A},\mathcal{B}\in\mathcal{P}(\mathbf{AP}_D)$ are such that $\mathcal{A}$ is a set of atoms of $\mathbf{AP}_D$ which can be deduced in $\mathcal{T}$ if $\vdash(h(x)\wedge c)$, and $\mathcal{B}$ is a set of $\mathbf{AP}_D$ which cannot be deduced in $\mathcal{T}$. ■

Theorem 4.- Let S be a pure program, and $h(x)$ a generic fact in S, such that $E(h(x))\neq\emptyset$. Then for any element$(l_{n-1},c_{n-1},l_n)\in E(h(x))$ a potential chain exits as:

$$G(h(x))=\{(h(x),c_0,\{l_1\}),(h(x),c_0\wedge c_1,\{l_1,l_2\}),\ \ldots$$
$$\ldots,\ (h(x),c_0\wedge c_1\wedge\ \ldots\ \wedge c_{n-1},\{l_1,l_2,\ \ldots,l_{n-1},l_n\})$$

Proof.- If $(l_{n-1},c_{n-1},l_n)\in E(h(x))$, then the element (l_{n-1},c_{n-1},l_n) can be in $E_+(h(x))$ or in $E_-(h(x))$, that is to say, at level n the element (l_{n-1},c_{n-1},l_n) must be in $E^n_+(h(x))$ or in $E^n_-(h(x))$. If (l_{n-1},c_{n-1},l_n) exists at level n, then thet other element $(l_{n-2},c_{n-2},l_{n-1})$ must exist in E(h(x)) too,

such that this element is in E_+^{n-1} or in E_-^{n-1} and so on until an element $(h(x),c_0,l_1)$ is obtained, such that this element is in E_+^0 or in E_-^0. We will obtain the succession of the potential attandence or absence at the levels 0,1, ...,n, with the following form:

$$(h(x),c_0,l_1),\ (l_1,c_1,l_2),\ \ldots\ ,(l_{n-1},c_{n-1},l_n)$$

It Is evident that any i-esimo element of this succession satisfies the conditions of definition (2), in consequence, the potential chain G(h(x)) must exist. ■

From now on, we will call a *potential chain associated with the generic fact* h(x) *in S* as any potential chain obtained as above.

Let G(h(x)) be a potential chain associated with the generic fact h(x), and $(h(x),c_0\wedge c_1\wedge\ \ldots\ \wedge c_{i-1},\{l_2,l_3,\ \ldots\ ,l_i\})$ is the element i-esimo of the G(h(x)).

This element can be rewritten as

$$(h(x),c_0\wedge c_1\wedge\ \ldots\ \wedge c_{i-1},\mathcal{A}_i\cup\mathcal{B}_i)$$

where $\mathcal{A}_i$ is a set formed by all positive literals, and $\mathcal{B}_i$ is a set formed by all negative literals $\{l_2,l_3,\ \ldots\ ,l_i\}$.

Corollary.- Let *S* be a pure program, and h(x) a generic fact, such that $E(h(x))\neq\emptyset$. Then an operation is associated for each element E(h(x)).

Proof.- It is evident that all elements (l_{i-1},c_{i-1},l_i) can be associated with the operation as $(h(x),c_0\wedge c_1\wedge\ \ldots\ \wedge c_{i-1},\mathcal{A}_i,\mathcal{B}_i)$. ■

notice that the elements in $\mathcal{A}_i$or $\mathcal{B}_i$ are formed by derivate predicates only.

Definition 24.- Let G(h(x)) be a potential chain associated with the generic fact h(x) in a program *S*, such that $G(h(x))=\{(h(x),c_i,\mathcal{A}_i\cup\mathcal{B}_i),\ i=1,2,\ \ldots\ ,n\}$. We will call the a *potential chain of operations associated with the generic fact* h(x) *in S*, the set of operations sorted

$$Gop(h(x))=\{(h(x),c_i,\mathcal{A}_i,\mathcal{B}_i),\ i=1,2,\ \ldots\ ,n\}\ \blacksquare$$

From now on, if it is not confused, then it will be called *chain of operations* only.

Notice, let $(h(x),c,\mathcal{A},\mathcal{B})$ be an operation in a program *S*, and $\mathcal{T}$ the deductive theory associated with *S*, for any $p(y)\in\mathcal{A}$, $\vdash (p(y)\leftarrow h(x)\wedge c)$, further, this formula is allowed. And for all $p(x)\in\mathcal{B}$, $\vdash(\neg p(x)\leftarrow h(x)\wedge c)$, but we cannot be sure that this formula is allowed.

Definition 25.- Let *S* be a pure program. For each generic fact h(x) in *S*, we will define the *set of operations associated with* h(x) *in S* as:

$$Op(h(x)) = \bigcup_i Gop_i(h(x)) \quad \blacksquare$$

Notice that the set Op(h(x)) has all the operations it can define from any element in any potential chain G(h(x)) and which it can construct from any element in E(h(x)). In virtue of this, if for a generic fact E(h(x)) is empty, then Op(h(x))=ø.

Definition 26.- Let *S* be a pure program. We will define the set Op of the operations associated with *S* as:

$$Op = \bigcup_{h(x)} Op(h(x)) \quad \blacksquare$$

Notice that Op depends on the kernel and on the generic fact in *S*. But Op is not dependent on the *set of facts* in *S*. In *S*, if the set of facts is changed, then the Op is an invariant of *S*.

Definition 27.- Let *S* be a pure program, h(x) a generic fact in *S*, and $(h(x),c_1,\mathcal{A}_1,\mathcal{B}_1)$ and $(h(x),c_2,\mathcal{A}_2,\mathcal{B}_2)$ two operations in Op(h(x)). We will say that the operation $(h(x),c_1,\mathcal{A}_1,\mathcal{B}_1)$ is included in the operation $(h(x),c_2,\mathcal{A}_2,\mathcal{B}_2)$, and we will denote it as $(h(x),c_1,\mathcal{A}_1,\mathcal{B}_1) \subset: (h(x),c_2,\mathcal{A}_2,\mathcal{B}_2)$, if $\{c_1\}c:\{c_2\}$, $\mathcal{A}_1 \subseteq \mathcal{A}_2$, y $\mathcal{B}_1 \subseteq \mathcal{B}_2$ are accomplished. ■

Theorem 5.- let *S* be a pure program, h(x) a generic fact in *S*, and G(h(x)) a potential chain associated with h(x) in *S*. Then the set of the operations associated with the elements in G(h(x)) has astructure of the lettice with respect to the relation of the inclusion of these operations.

Proof.- It is evident. ■

2.4 Reduction of the Logic Program.

Let **a** be an atomic formula formed by a derivate predicate in *S*. We can always find a operation $(h(x),c,\mathcal{A},\mathcal{B})$, such that $\mathbf{a}\in\mathcal{A}$ for some h(x) and c. This operation will correspond with some element of some of the potential chains associated with h(x). In virtue of theorem (5) and definition (25), a high least operation $(h(x),c',\mathcal{A}',\mathcal{B}')$. can be found such that $\mathbf{a}\in\mathcal{A}'$ and $\mathcal{A}'$ is the minimum set associated with this chain. Then $\vdash(h(x)\wedge c')$ is the minimum condition to satisfy $\vdash\mathbf{a}$.

Definition 28.- Let *S* be a pure program, and p(y) an atomic formula such that p is a derivate predicate of *S*. We will call the *minimum condition the atomic formula* p(y) *is valid*, at a conjunction of the formula as h(x)∧c such that a chain of operations Gop(h(x)) exist in *S* and the following conditions are satisfied:

a) $\exists(h(x),c,\mathcal{A},\mathcal{B})\in Gop(h(x))$ such that $p(y)\in\mathcal{A}$, and

b) $\neg\exists(h(x),c',\mathcal{A}',\mathcal{B}')\in Gop(h(x))$ such that $(c'c:c)\wedge(p(y)\in\mathcal{A}')\wedge(\mathcal{A}'\subseteq\mathcal{A})\}$ ■

It is evident, that one or more chain of operations Gop(h(x)) can be associated with the same formula atomic p(y). Also for each atomic formula formed with a derivate predicate, a set of the minimum conditions, can be

associated, which will be denoted as $\mathcal{A}_c(p(y))$, Further, this set cannot empty.

Definition 29.- Let S be a pure program, and p(y) an atomic formula such that p is a derivate predicate of S. We will call the *minimum condition for which the atomic formula* p(y) *is not valid*, at a conjunction of the formula h(x)∧c such that a chain of the operations Gop(h(x)) exist in S, and the following conditions are satisfied:

a) $\exists(h(x),c,\mathcal{A},\mathcal{B})\in Gop(h(x))$ such that $p(y)\in\mathcal{B}$, and
b) $\neg\exists(h(x),c',\mathcal{A}',\mathcal{B}')\in Gop(h(x))$ such that $(c'\subset:c)\wedge(p(y)\in\mathcal{B}')\wedge(\mathcal{B}'\subseteq\mathcal{B})$ ■

Evidently, one or more minimum conditions can satisfy the definition (29). The set of these minimum conditions, which will be denoted as $\mathcal{B}_c(p(y))$, can be empty.

Definition 30.- Let S be a pure program, and p(y) an atomic formula such that p is a derivate predicate of S. A generic fact h(x) is *primordial to validate* p(y), if this generic fact is in some element of $\mathcal{A}_c(p(y))$. ■

The set of primordial generic fact to validate the atomic formula p(y) is denoted as $\mathcal{A}_c^0(p(y))$, that is to say,

$$\mathcal{A}_c^0(p(y))=\{h(x) \mid \exists(h(x)\wedge c)\in\mathcal{A}_c(p(y))\}$$

Definition 31.- Let S be a pure program, and p(y) an atomic formula, such that p is a derivate predicate in S, A generic fact h(x) is *primordial to non-validate* p(y), if this generic fact is in some element of $\mathcal{B}_c(p(y))$. ■

The set of primordial generic facts to non-validate the atomic formula p(y) is denoted as $\mathcal{B}_c^0(p(y))$, that is to say,

$$\mathcal{B}_c^0(p(y))=\{h(x) \mid \exists(h(x)\wedge c)\in\mathcal{B}_c^0(p(y))\}$$

Notice that, some p(y) will have primordial generic facts as much in that $\mathcal{A}_c^0(p(y))$ as in $\mathcal{B}_c^0(p(y))$.

The construction of the sets $\mathcal{A}_c^0(p(y))$ and $\mathcal{B}_c^0(p(y))$ allow us to isolate the generic facts which esentielly depend on the derivation or non-derivation of p(y). With a view to classic logic resolution, the sets of the primordial generic fact allow us to establish an order of evaluation which is deduced from the very own program.

Lemma 1.- Let S be a pure program. If h(x) is a primordial fact, then $E(h(x))\neq\emptyset$.

Proof.- It is immediate. ■

Definition 32.- Let S be a pure program, and p(y) an atomic formula, such that p is a derivate predicate. A conjunction of the literals c is a *residual condition to validate the atomic formula* p(y), if this condition c is included in some members of $\mathcal{A}_c(p(y))$. ■

We will denote the set of residual conditions to validate p(y) as

$$\mathcal{A}^{\Gamma}_{c}(p(y))=\{c \mid \exists(h(x)\wedge c)\in\mathcal{A}_{c}(p(y))\}$$

Definition 33.- Let S be a pure program, and p(y) an atomic formula, such that p is a derivate predicate. A conjunction of the literals c is a *residual condition to non-validate the atomic formula* p(y), if this condition c is included in some members of $\mathcal{B}_{c}(p(y))$. ■

Notice that the same conjunction of the literals c can be in $\mathcal{A}^{\Gamma}_{c}(p(y))$ and $\mathcal{B}^{\Gamma}_{c}(p(y)$ at the same time.

We denote

$$a(p(y)) = \bigvee_{(h(x)\wedge c)_i\in\mathcal{A}_{c}(p(y))} (h(x)\wedge c)_i$$

$$b(p(y)) = \bigvee_{(h(x)\wedge c)_i\in\mathcal{B}_{c}(p(y))} (h(x)\wedge c)_i$$

Theorem 6.- Let S be a pure program which is associated with a deductive theory $\mathcal{T}$ under the assumption of the complementary. Then S is correct if and only if for any p(y), such that p is a derivate predicate and $\mathcal{B}_{c}(p(y))\neq\emptyset$, and under $\mathcal{T}$ verify itself

$$\vdash (\neg(a(p(y))\wedge b(p(y)))$$

Proof.- Firstly, the necessary condition will be proved. If S is a correct program, then an atomic formula p(y) cannot exist such that it is valid and non-valid at the same time in the hypothesis of the theorem. In virtue of this, the generic facts h(x) and h'(x'), and the conjunctions c and c', such that $(h(x)\wedge c)\in\mathcal{A}_{c}(p(y))$, $(h'(x')\wedge c')\in\mathcal{B}_{c}(p(y))$ and $\vdash(h(x)\wedge c\wedge h'(x')\wedge c')$, cannot exist. If S is correct, then $\vdash (\neg(a(p(y))\wedge b(p(y)))$. ■

With similar reasoning the sufficient condition will be proved. If $\mathcal{B}_{c}(p(y))\neq\emptyset$, and the expression of the theorem is valid, then the generic facts and condition in the before conditions, such that it can deduce itself true and false at time, it cannot exist. Thus S is correct. ■

This theorem provides a method to prove if a program is correct or not. For each atom p(y) in S the expression $a(p(y))$ and $b(p(y))$ must be constructed, and if the theorem is validated for all atoms in S, this program is correct.

Theorem 7.- Let S be a pure correct program which is associted with adeductive theory $\mathcal{T}$ with the assumption of the complementation. For any atom p(y) in S, such that p is derivate predicate, then the following expression must verify:

a) $\vdash (p(y) \Leftrightarrow (a(p(y))\wedge\neg b(p(y))))$

b) $\vdash (\neg p(y) \Leftrightarrow (\neg a(p(y))\vee b(p(y)))$.

Proof.- If we consider the construction form of $\mathcal{A}_c$, $\mathcal{B}_c$ and the expression $a(p(y))$, $b(p(y))$, then it is evident. ■

Corollary 1.- Let S be a correct pure program which is associated with a deductive theory $\mathcal{T}$ under the assumption of the complementation. For any atom p(y) in S such that p is a derivate predicate, then in $\mathcal{T}$ the following expression must be verified:

$$\vdash (p(y)) \Leftrightarrow a(p(y)))$$

Proof.- If $\mathcal{B}_c(p(y))$ is empty, then the expression $b(p(y))$ cannot be constructed and in virtue of theorem (7), the proof of the corollary is evident. If $\mathcal{B}_c(p(y))$ is not empty, in virtue of theorem (6), $\vdash a(p(y)) \Rightarrow \vdash \neg b(p(y))$, and because of theorem (7), $\vdash (p(y) \Leftrightarrow a(p(y)))$ ■

Corollary 2.- Let S be a correct pure program which is associated with a deductive theory with assuption of the omplementation. For any atom p(y) in S, such that p is derivate predicate, if $\vdash(p(y)\wedge\phi(z))$ is satisfied, then $\vdash(a(p(y))\wedge\phi(z))$ is satisfied also.

Proof.- It is evident. ■

Corollary 3.- Let S be a correct pure program which is associated with a deductive theory with assumption of the complementation. For any atom p(y) in S, such that $\mathcal{B}_c(p(y))=\varnothing$, then in $\mathcal{T}$ the following expression must be satidfied:

$$\vdash (\neg p(y) \Leftrightarrow \neg a(p(y)))$$

Proof.- In virtue of theorem (7) it is evident. ■

The corollary (1) of the theorem (7) provide the possibility of rewritting S, such that in the new program the kernel is formed by clausulas with the body as $h(x)\wedge c$, where h(x) is a primordial generic fact, and c is a conjunction of the literals. In order to obtain the new program, we will select an atom p(y) for each derivate predicate in S, and we will construct the expression $a(p(y))$.We will then write the new clausulas as conjunctions in $a(p(y))$. The new clausulas are the form: $p(y)\Leftarrow h(x)\wedge c$, where h(x) is a primordial generic fact, such that $h(x)\wedge c$ is a conjunction in $a(p(y))$, and the literals with derivate or basic predicate can appear in c.

Next, In virtue of corollary (2) of the theorem (7) and the mgcs in definition (7), a process of the complete reduction can been stablished. This process allows to rewrite a hierarchy program S, such that the kernel of the new program is formed by clauses with bodies which are formed by a conjunction of the primordial generic facts only. If the program S is stratified, then literals formed by derivate and basic predicates can appear in the bodies of the new clausulas. These literals formed by derivate predicates are atoms which have been defined recursivaly.

Notice that the new kernel of the S is equivalent to S by the corollary (2). Further, this new kernel is formed by allowed formulas too.

Definition 34.- Let S be a correct pure program and $p(u)\wedge\phi(z)$ a wff in **L** such that p is a derivate predicate of S. We define a *reduction* over $p(u)\wedge\phi(z)$ as

the process which transforms this formula into an other formula as $(\alpha(p(y))\wedge\phi(z))\theta$, where θ is a mgcs as $\theta[\alpha(p(y)), p(u)\wedge\phi(z)]$. ■

Notice that if $\vdash(p(u)\wedge\phi(z))$, then $\vdash(\alpha(p(y))\wedge\phi(z))\theta$, in virtue of corollary (2).

If the reduction process is applied over the formula $p(u)\wedge\phi(z)$, then we will get an other formula $\alpha(p(u))\wedge\phi(z)\theta$ without the atom p(u). If the program is a hierarchy, then all atoms of the bodies formed by derivate predicates can be replaced in a finit numbers of the reductions.

Definition 35.- Let *S* be a correct pure program, and $p(u)\wedge\phi(z)$ a wff in **L**, such that p is a derivate predicate of *S*. We can define a *complete reduction* over $p(u)\wedge\phi(z)$ as the process which transforms this formula to an other formula as $(\alpha(p(y))\wedge\phi(z))\theta$ in a finite number of reductions, such that the atoms formed by derivate predicates are not in the new formula, and θ is a mgcs as $\theta[\alpha(p(y)),p(u)\wedge\phi(z)]$. Then we can say that this formula is *completely reduced.*

It is obvious if a correct pure program *S* is a hierarchy (non-recursive), then the body of all the clausulas of the kernel of *S* can be reduced completely, in virtue of theorem (1) and (2).

In the event of *S* being stratified (recursive), when we try to apply a completely reduction over a formula, we will come into an infinite reduction process, thus this formulas cannot be reduced completely. Hovewer, if an atom with a derivate predicate p appears in a step of the reduction process, and this predicate has appeared in the previous step, then the reduction process is not applied over this atom. In consequence of this change in the reduction process, the atoms can be classified as recursive and non-recursive atoms. If only the reduction process is applied over the non-recursive atoms, then the completed reduction process is a finite numbers of steps. In this case we can say that the formula is reduced completely too.

Definition 36.- A pure program *S* is in *completely reduced form*, if the body in all clausulas of the kernel are reduced completely. The completely reduced form of the program *S* is denoted as *S'*. ■

It is evident that a completely reduced program is allowed too.

Theorem 8.- If a program is recursive o not recursive, pure and correct, then this program can be rewritten in completely reduced form.

Proof.- It is evident. ■

2.5 Concepts of the Characteristic and Generic State.

Let *S'* be a completely reduced program. For all primordial generic fatcs in *S* the potential chains G'(h(x)) must exist in *S'*, in virtue of lemma (1) and theorem (4), such that any element in G'(h(x)) as $(h(x),c'_0\wedge c'_1\wedge \ldots \wedge c'_{n-1},\{l'_1,l'_2, \ldots ,l'_n\})$, the c'_i is completely reduced for i=1,...,n. If we apply the process of reduction over G(h(x)), then we obtain G'(h(x)) also.

Definition 37.- Let *S'* be a pure program, such that it is in completely reduced form, and let h(x) be a primordial generic fact. A *potential reduced*

chain associated with a primordial generic fact h(x) *in S* is a potential chain associated with h(x) in S', if all conjunctions in the elements of this potential chain are completely reduced. ■

The potential chain associated with a primordial generic fact has been called *potential reduced chain.*

Definition 38.- Let S' be pure program in completely reduced form, h(x) is a primordial generic fact in S', G'(h(x)) is a potential reduced chain and $(h(x),c'_0 \wedge c'_1 \wedge \ \dots \ \wedge c'_{n-1},\{l'_1,l'_2, \ \dots \ ,l'_n\}) \in G(h(x))$ is an element of the chain. Then the following trio:

$$(h(x),c'_0 \wedge c'_1 \wedge \ \dots \ \wedge c'_{n-1},l'_1 \wedge l'_2 \wedge \ \dots \ \wedge l'_n).$$

is called *potential characteristic state* (pcs). ■

the set of potential characteristic state associated with the program S' will be denote as ξ.

A pcs is associated with an element of the potential chain and a primordial generic fact. Therefore the concept of state is implicit with the ocurrence of a primordial generic fact. With a view to defining a finite state machine, we will include an empty potential chain which we will denote as G(ø). This is not associated with some primordial generic facts, and we agree to associate a *start potential characteristic state* which we denote as s_0.

Definition 39.- Let $(h(x),c'_0 \wedge c'_1 \wedge \ \dots \ \wedge c'_{n-1},l'_1 \wedge l'_2 \wedge \ \dots \ \wedge l'_n)$ be a pcs. A *characteristic state* is the last componet of the pcs, that is to say, the following conjuction of the literals:

$$\tau = \bigwedge_{i=1}^{n} l'_i \quad ■$$

We agree to call *start characteristic state* the characteristic state associated with s_0. Notice that τ_0 is an empty conjunction.

Let S' be a program, $\mathcal{T}$ is the deductived theory associate with S' and (h,c,τ) is a pcs. If $\vdash(h \wedge c)$ is valid in $\mathcal{T}$, then $\vdash\tau$ too. In $\mathcal{T}$, one or more pcss can be valid at the same time. If (h,c,τ) is a pcs valid in $\mathcal{T}$, then in τ it is not all that it is valid in $\mathcal{T}$.

Definition 40.- Let q be a conjunction of the positive literals formed with derivate predicates, and let S be a program which is associated with a deductive theory $\mathcal{T}$. The conjunction q is a *generic state* if verify itself:

a) $\vdash$ q in $\mathcal{T}$

b) For any positive literal formed with some derivate predicate, if $l \notin \{q\}$, then $\vdash l$ is not valid. ■

Similary, a *start generic state* which correspond at start characteristic state it is introduced, this start generic state is denotated by q_0, which represent the empty conjunction.

Definition 41.- Let S be a correct pure program, $\mathcal{T}$ a deductive theory

associated with S, c a conjunction of the positive literals formed by primordial generic facts, and q a generic state. The double (c,q) is a *potential generic state*, if in $\mathcal{T}$ it satisfy itself:

$$\vdash (c \Longrightarrow q). \quad \blacksquare$$

Also we will introduce the *start potential generic state* which is formed by an empty conjunction c_0 and the start generic state q_0. This start potential generic state is noted by (c_0,q_0).

2.6 Diagram of the Generic States.

The potential generic state (c,q) expresses a possible situation of a program S, since the generic state c are all the positive literals formed by derivate predicates which are valid, if the conjunction c is valid too.

Definition 42.- Let S be a correct pure program. We can say that a directed graph labelled doubly is a *diagram of the potential generic states* (DPGS) associated with S, if

a) The nodes of the DPGS are labelled with potential generic states. One node is labelled by (c_0,q_0),

b) The arches are labelled by conjunctions of the primordial generic facts.

c) A path exists from the node (c_0,q_0) to the node (c,q), for any node (c,q) of a DEGP. Further c is a conjunction of the label of the arches in this path. ■

Definition 43.- A DPGS is *complete* if any potential generic state which it can define in S, thus this is in the DPGS. ■

Teorem 9.- Each correct pure program S is associate with a complet DPGS.

Proof.- We propose an algorithm to construct the DPGS from the set ξ in a finite number of the steps, where ξ is the set of potential characteristic states associated with the program S. We use the next notations:

a) The potential characteristic states are represented as $(h,c\wedge P,\tau)$, where c is a conjunction of the literals which are formed by primordial generic facts, and P is a conjunction of the literals which are formed by recursive derivate predicates.

b) Let b be a conjunction of the literals, then $\{b\}^-$ is the set of the atoms which corresponds with the negative literals in the conjunction b.

c) Ω_i is the set of the nodes of the DPGS which is generated until the i-esim step.

The algorithm is:

FERST STEP, i=0:

to generate the start node: (c_0,q_0);

For each $(h,c\wedge P,\tau)\in\xi$ such that $\{P\}^+=\emptyset$, $\{c\}^+\neq\emptyset$, $\{\tau\}^+\neq\emptyset$,

do

If $(h\wedge c^+,\tau^+)\notin\Omega_0$,

then

to generate a new node labelled with: $(h\wedge c^+,\tau^+)$;

end-if;

to generate a new arch labelled with:$h\wedge c^+$ from (c_0,q_0) to this node;

end-do;

STEP i, i>0:

For each $(c,q)\in\Omega_{i-1}$,

do

While $\exists(h,c\wedge P,\tau)\in\xi$ | $\exists\Xi[c\wedge q,h\wedge c\wedge P\wedge\tau]$ gmgcs and $(\{c\}-\{h\Xi\})\subseteq\{c\Xi\}^+$, $\{c\}\cap\{c\Xi\}^-=\emptyset$, $\{P\Xi\}^+\subseteq\{q\}$, and $\{P\Xi\}^-\cap\{q\}=\emptyset$ are accomplished

do

if $\{q\}\cap\{\tau\Xi\}^-\neq\emptyset$,

then

to obtain the set: $\{q'\} = \{\tau\Xi\}^+\cup(\{q\}-\{\tau\Xi\}^-)$;

else

to obtain the set: $\{q'\} = \{\tau\Xi\}^+\cup\{q\}$;

end-if;

to generate the generic state $q'= \wedge_{l\in\{q'\}} l$;

if $(c\wedge h\Xi\wedge c\Xi^+,q')\notin\Omega_i$,

then

to generate a new node labelled $(c\wedge h\Xi\wedge c\Xi^+,q')$, that it belong to Ω_i.

end-if;

to generate a arch from (c,q) to $(c\wedge h\Xi\wedge c\Xi^+,q')$, such that it is labelled by $\wedge_{l\in(\{c\Xi\}^+-(\{c\}-\{h\}))} l$;

end-do;

end-do.

FINAL STEP: The algorithm stops at a step n, if $\forall(c,q)\in\Omega_{n-1}$ is not some $(h,c\wedge P,\tau)\in\xi$, such that $\exists\Xi[c\wedge q,h\wedge c\wedge P\wedge\tau]$ mggcs and $(\{c\}-\{h\Xi\})\subseteq\{c\Xi\}^+$, $\{c\}\cap\{c\Xi\}^-=\emptyset$, $\{P\Xi\}^+\subseteq\{q\}$, and $\{P\Xi\}^-\cap\{q\}=\emptyset$ is accomplished.

It is evident to prove that the DPGS is complete, in virtue of the form as it has been constructed. ■

Henceforth the set of the nodes in DPGS is denoted as Ω. Thus Ω is the set of the potential generic states which can be defined in a program *S*.

Ours objective is to define a machine of the states, where the inputs are the primordial generic facts, the states are the generic states, such that it simulates the behaviour of the kernel of a logic program. This machine can change of state when it reads a primordial generic fact.

We can think that a complete DPGS of a program S can represent a diagram of the transition to this machine if the arches in the DPGS are labelled with one literal.

Lemma 2.- Let S be a correct pure program, and DPGS a complete diagram of the potential generic states. A directed and doubly labelled exists and is denoted as E-DPGS if it satisfied the following conditions:

a) Each arch is labelled by one literal.

b) The set of the nodes DPGS Ω is included in the set of the nodes in E-DPGS.

c) For any pair of the nodes (c,q), $(c',q')\in\Omega$ such that an arch exists from (c,q) to (c',q') and is labelled by $h_1 \wedge h_2 \wedge \ldots \wedge h_n$, then a paths exist in EDPGS with n arches from (c,q) to (c',q') such that these arches are labelled by $h_1, h_2, \ldots, h_n$ respectively.

Proof.- It is evident, if we include n new nodes between (c,q) and (c',q'). We need to include n new auxiliar predicates in P_D, and to extend the kernel of S. ■

Definition 44.- Let S be a correct pure program. We denominate an *extended diagram of the potential generic states associated with S*, E-DPGS, if this satisfies the conditions in lemma (2). ■

A program S can be rewritten as S', such that S' is associated with a diagram of the potential generic states, which is extended. the S' is denominated as an *expanded form of the S*.

We denote:

- Q as the set of the generic states associated with the form expanded of S.

- $\mathcal{F}\subset Q$ as the set of the generic states which cannot pass to other generic states.

Theorem 10.- Given a correct pure program S, a finite state machine exists $M_g(S)=(\Sigma_g,Q_g,q_0,F_g,\delta_g)$, where Σ_g is the finite set of allowable tape symbols (input symbols), Q_g is the finite set of states, $q_0\in Q_g$ is the start state, $F_g\subseteq Q_g$ is the set of final states, and $\delta_g:\Sigma_g\times Q_g \longrightarrow Q_g$ is the transition function, such that M_g emulate the behavior of the kernel of S when the kernel is stimulated by the occurrence of primordial generic facts.

Proof.- Let E-PGSD be an extended potential generic state diagram associated with S. We define

- Σ_g as the set of primordial generic facts in S

- $Q_g = Q$,

- q_0 as the start generic state,

- $F_g = \mathcal{F}$, and

- $\delta_g : \Sigma_g \times Q_g \longrightarrow Q_g$ as a partial function such that, for $h \in \Sigma_g$, $q_1, q_2 \in Q_g$, $\delta_g(h,q_1) = q_2$ if two nodes exist (c_1,q_1), (c_2,q_2) and an arc, labelled h, between them in E-PDGS.

The properties of an E-PDGS guarantee that the $M_g(S)$ machine emulate the behavior of the kernel of S. ■

Definition 45.- Let S be pure correct program in expanded form, and $M_g = (\Sigma_g, Q_g, q_0, F_g, \delta_g)$ a machine of the finite states. We can say that $(c, \mathcal{A}, \mathcal{B})$ is a *transactional operation activated by a primordial generic fact* h if q_1, q_2 exists in Q_g such that

a) $\delta_g(h,q_1) = q_2$,
b) $\mathcal{A} = \{q_2\} - (\{q_1\} \cap \{q_2\})$,
c) $\mathcal{B} = \{q_1\} - (\{q_1\} \cap \{q_2\})$,
d) $c = q_1 \wedge (\wedge_{l \in \mathcal{A}} \neg l)$ ■

Let h be a primordial generic fact, the set of the transactional operations activated by h be denoted as $\mathcal{O}(h)$, and the set of the transactional operations associated with a program S as $\mathcal{O}(S)$.

Definition 46.- A *derivate fact* is a basic atomic formula formed by a derivate predicate. ■

Let $\mathcal{D}(S)$ be the set of the derivate facts associated with a program S, and $W_i(S) \subseteq \mathcal{D}(S)$ the set of the derivate facts, which has been deduced after the occurrence of the i facts.

The set $W_i(S)$ can be obtained by a process of the deduction of the minimum or standard model, or by applying the sequential of the i facts over the set of the transactional operations associated with S.

The process of apply a fact over the set of the transactional operations associated with a program S is:

Let $h_1(a)$ be the first fact which take place, such that $a \in \mathbf{A}^n$ and $h_1(a)$ are the specialization of the primordial generic fact $h_1(x)$ for $x \in \mathbf{V}^n$. Let $(c_1, \mathcal{A}_1, \mathcal{B}_1)$ be the operation in $\mathcal{O}(h_1(x))$, such that it is actived by $h_1(a)$. Thus the specialization of this operation will be obtained to apply the *unification* of $h_1(a)$ with $h_1(x)$ over c_1 and all members in $\mathcal{A}_1$ and $\mathcal{B}_1$. In consequence, all the literals of the c_1 and all members in $\mathcal{A}_1$ and $\mathcal{B}_1$ are unified, and $W_1(S) = \mathcal{A}_1^a$, where $\mathcal{A}_1^a$ expresses the set of literals unified with a in $\mathcal{A}_1$.

Let $h_2(b)$ be the second fact, and $(c_2, \mathcal{A}_2, \mathcal{B}_2) \in \mathcal{O}(h_2)$ is a transaction operation

which is activated, depending on the h_2 and $W_1(S)$. The following process is needed to determine this operation.

1) Unifying the $h_2(x)$ with $h_2(b)$, (this unification must be applied over the literals of the c_2 and the members in $\mathcal{A}_2$ and $\mathcal{B}_2$) we obtain $c_2^{b'}$, $\mathcal{A}_2^{b'}$, and $\mathcal{B}_2^{b'}$ such that they are partially unified.

2) Unifying the $c_2^{b'}$ with the members in $W_1(S)$, all the variables in $c_2^{b'}$, $\mathcal{A}_2^{b'}$, $\mathcal{B}_2^{b'}$ are substituted with a constant. Then

$$W_2(S) = (W_1(S)-\mathcal{B}_2^b) \cup \mathcal{A}_2^b$$

where $\mathcal{A}_2^b$ and $\mathcal{B}_2^b$ express the members of the $\mathcal{A}_2$ and $\mathcal{B}_2$ full, unified.

Theorem 11. *Theorem of the equivalent between resolutions.*- Let S be a correct pure program, and $\mathcal{H}_n = \{h_1(a_1), h_2(a_2), \ldots, h_n(a_n)\}$ a set of the n facts in S. Then the set of the derivate facts $W_n(S)$ obtained to apply the sequence of the transactional operations which are actived by the sequence of the facts $h_1(a_1)$, $h_2(a_2)$, ... ,$h_n(a_n)$, and the minimum model of Herbrand or standard model H_0 are equals.

Proof.- Firstly, we will prove that any derivate predicate $p(a) \in W_n(S)$ is in H_0, and after that all derivate facts in H_0 are in $W_n(S)$ too.

The first part, is evident, since executing the transactional operations is equal to advancing in the E-DPGS.

In the second part we consider any $p(a) \in H_0$. If p(a) exists in H_0, then a clausula unless in the form $p(x) \leftarrow \Phi(z)$ must exist in S too. This clausula must prove a potential attandence or absence, and in consequence, must be included in E-DPGS and the transactional operations. This is evident. ■

3. Application of the Potential Effects Method (PEM) ontothe Deductive Conceptual Model.

The PEM cannot be applied over a specification of IS by DCM directly since this is not a logic program, although a relational language is used for the specification of the IS by DCM. The DCM needs to include the times for specification of the behavior, and this provide a order over sequence of facts which is allowed.

In this section we present a logic program deduced from a specification of IS by DCM, and this logic program is temporally normalized, specified in a temporal language, thus the PEM is extended over this program.

3.1 Specification of the Information Systems by DCM.

The DCM has usually been distinguished as having six components: declaration of domains, declaration of basis and derivate predicates, integrity constraints (IC), derivation of rules (DR), and inputs and outputs.

We assume that the IC and DR of the DCM are specified in a relational language **L'**. This language **L'** is defined as **L'**=($\mathbb{A}$',$\mathbb{F}$'), where $\mathbb{F}$' is the set of well formed formulas in **L'**, $\mathbb{A}$'= **A** ∪ $\mathbb{T}$ ∪ **P** ∪ **S** ∪ **V** ∪ $\mathbb{V}$, such that **A** expresses a set of constants, $\mathbb{T}$ is a set of the instants of time, **S** is a set of logic symbols, **V** is a set of variables, $\mathbb{V}$ is a set of temporary variables, and **P** is a set of predicates with an arity such that the time is included as the last argument in every predicate; Further the binary relational predicates are in **P**.

We will suppose that IC and DR are composed of Horn clausulas with negation in the body, also they are well formed formulas (wff) in **L'**, closed formulas and the temporal constraints are included in every clausula. The temporal constraints provide the relation between the temporal domains in the formulas.

We will consider the IC and DR as the essential components of DCM. The IC and DR expresses the behavior of systems and all the elements that constitute the systems, and the relation between the elements. We will consider that the DCM is formed by the components IC and DR only, and that the other components are not important to our work.

The IC is expressesed in denial form, and it are considered as a special derivation rules which is formed as $i(t) \Leftarrow \phi(x,t)$, where i is an integrity predicate in **P**. The DR and IC can be considered as the kernel of logic program, and the set of events occuring in the instants of time is the set of facts.

From now on we suppose a DCM as a logic program $\mathcal{G}$ in **L'** with allowed and closed formulas, and this program is hierarchical or stratified and it is normalized by the transformations proposed in LLoyd and Topor [LLO87]. Moreover, that this program is pure and well constructed is assumed.

In an instant σ we define an instantaneous interpretation $\mathbf{I}_\sigma$ over the relational language **L'** as triplets (D',I'_c,I'_v), where:

$D'=D\cup\mathbb{T}_\sigma$ such that D is the semantic domain over the constant, and $\mathbb{T}_\sigma=\{\sigma_0,\sigma_1,\ldots,\sigma_n,\sigma\}$ is the semantic domain over $\mathbb{T}$, $\mathbb{T}_\sigma$ as an ordered set.

I'_c is a set of functions such that let p be a predicate in **P** with arity n, then $I'_c: D^{n-1}\times\mathbb{T}_\sigma \longrightarrow \{T,F\}$, (T is true, and F is false). And $\forall a\in(A\cup\mathbb{T})$, then $I'_c(a)\in D'$.

I'_v is such that $\forall x\in V$ and $\forall t\in\mathbb{V}$, then $I'_v(x)\in D$ and $I'_v(t)\in\mathbb{T}_\sigma$.

Under the interpretation $\mathbf{I}_\sigma$ it must verify:

If $a\in A\cup\mathbb{T}$, $\mathbf{I}_\sigma(a)=_{def} I'_c(a)$.

If $x\in V\cup\mathbb{V}$, $\mathbf{I}_\sigma(x)=_{def} I'_v(x)$.

We define $\mathbf{I}_\sigma(p(k_1,k_2,\ldots,k_n,t) =_{def} I'_c(p)(\mathbf{I}_\sigma(k_1),\mathbf{I}_\sigma(k_2),\ldots,\mathbf{I}_\sigma(k_n),I(t))$.

If $\mathbf{I}_\sigma(k=l)=_{def}$ T (true) if $\mathbf{I}_\sigma(k)=\mathbf{I}_\sigma(l)$.

If $\mathbf{I}_\sigma(t<t')=_{def}$ T (true) if $\mathbf{I}_\sigma(t) < \mathbf{I}_\sigma(t')$.

If $\mathbf{I}_\sigma(t\leq t')=_{def}$ T if $\mathbf{I}_\sigma(t)\leq\mathbf{I}_\sigma(t')$:

Let $A(t,t')\wedge t<t'$ be a wff in L' with two temporary domains, then

$$I_\sigma(A(t,t')\wedge t<t') =_{def} I_\sigma(A(t,t') \text{ y } I_\sigma(t)<I_\sigma(t') \text{ y } I_\sigma(t=\sigma),$$

if $I_\sigma(A(t,t')\wedge t<t')$ get one or more values, then the major value in the time is assumed [SIS87].

Let $\mathfrak{G}$ be a logic program in L', and let l be a literal in a sentence *s* of the program $\mathfrak{G}$. Then $I_\sigma(l)$ only obtains a value which is not the same that value obtained by $I_\sigma(l)$ in $I_\sigma(s)$. This is in virtue of which the time included in every predicate and in all sentences of the program which have temporary constraints. This makes the manipulation of the literal in the sentences difficult.

We define a temporal language which supplies the manipulation of these literals, since it clusters the literals under the same the temporary domains. The literals in the scope of the same temporary domains can be applied to the calculus of predicates without temporary constraints.

The temporal language is defined as $L_t=(\mathbb{A}_t,\mathbb{F}_t)$, where $\mathbb{F}_t$ is the set of the wffs in L_t, $\mathbb{A}_t = A\times\Xi \cup P \cup V \cup S \cup O$, such that A is the set of constants in L', Ξ is a subset in $\mathbb{N}^+$ which expresses the set of the states, P is the set of predicates but with arity without temporal argument, V is the set of variables, S is the set of logic symbols, and O= {P, PC, PP, AP} is the set of the temporal operator where the operator P expresses the *past*, PC expresses the *past or current*, PP expresses the *recent past*, AP expresses the *always past*, and without operators expresses the *present*.

The concepts: terms, atomic formula, and basic formula on L_t are simular to the same concepts in the relational languages.

The rules to define the wffs on L_t are:

1.- All atomic formula are a wff.

2.- Let A and B be wffs, then
¬A, A∧B, A∨B, A⇐B, A⇔B, P(A), PP(A), PC(A), and AP(A) are wffs too.

3.- Let A be a wff, *x* is a variable in A, then ∀*x*A, and ∃*x*A are wffs.

4.- The rest of the formulas are not wffs. ■

This temporal language has been defined similar to the modal and temporal logic language.

A interpretation I_t on the temporal language L_t is defined as $I_t = (\mathfrak{E},R,D,I_c, I_v)$, where:

D is the domain of the interpretation,

I_c is a set of functions, such that p is a predicate in P with arity n, then $I_c(p):D^n \longrightarrow \{T,F\}$. If $s\in\mathfrak{E}$, and $l\in A_t$, $I_c^S(l)\in D$.

I_v is as $x\in V$ and let s be a member in $\mathfrak{E}$, then $I_v^S(x)\in D$.

$\mathfrak{E}$ is a finite and enumerate set of elements which expresses the states in time. Further, a partial order relation is defined on $\mathfrak{E}$.

R is a function of the predecessor relation on $\mathfrak{E}$. The $(\mathfrak{E},R)$ define the frame of interpretation.

Below is the definition of I_t, and for each s in $\mathfrak{E}$ the following formulas are satisfied:

If $x \in V$, then $I_t(s,x) =_{def} I_v^S(x)$ (1).

If $l \in A_t$, Then $I_t(s,l) =_{def} I_c^S(l)$ (2).

If p is a predicate such that $p \in P$ with arity m, and $t_1,t_2,\ldots,t_m$ are terms on p, then

$I_t(s,p(t_1,t_2,\ldots,t_m)) =_{def} I_c(p)(I_t(s,t_1),I_t(s,t_2),\ldots,I_t(s,t_m))$ (3).

If t and r are terms, then

$I_t(s,t=r) =_{def} T$, si $I_t(s,t) = I_t(s,r)$ (4).

Let A and B be wff on F_t, then to must accomplish:

$I_t(s,A \wedge B) =_{def} I_t(s,A) \wedge I_t(s,B)$ (5),

$I_t(s,A \vee B) =_{def} I_t(s,A) \vee I_t(s,B)$ (6),

$I_t(s,\neg A) =_{def} \neg I_t(s,A)$ (7),

$I_t(s,(A \Rightarrow B)) =_{def} I_t(s,\neg A) \vee I_t(s,B)$ (8),

$I_t(s,P(A)) =_{def} I_t(R^i(s),A)$ for some $i \geq 1$, (9)

$I_t(s,PP(A)) =_{def} I_t(R(s),A)$, (10)

$I_t(s,PC(A)) =_{def} I_t(s,A) \vee I_t(s,P(A)) = I_t(R^i(s),A)$, for some $i \geq 0$, (11)

$I_t(s,AP(A)) =_{def} I_t(R^i(s),A)$ for any $i \geq 1$, (12)

$I_t(s,\forall x A) =_{def} T$, if for all x/d and $I_v(x)=d$, $d \in D$. (13)

$I_t(s,\exists x A) =_{def} T$, if for some x/d and $I_v(x)=d$, $d \in D$. (14)

$I_t(s,\neg P(A)) =_{def} I_t(R^i(s),\neg A)$ for any $i \geq 1$, (15)

$I_t(s,\neg AP(A)) =_{def} I_t(R^i(s),\neg A)$ for some $i \geq 1$, (16)

$I_t(s,\neg PC(A)) =_{def} I_t(s,\neg A) \wedge I_t(s,\neg P(A)) = I_t(R^i(s),A)$, for any $i \geq 0$ (17)

$I_t(s,\neg PP(A)) =_{def} I_t(R(s),\neg A) = I_t(s,PI(\neg A))$, (18)

$I_t(s,P(P(A))) =_{def} I_t(R^i(s),A)$ for some $i \geq 2$, and $s \geq 2$ (19)

$I_t(s,P(PP(A))) =_{def} I_t(R^i(s),A)$ for some $i \geq 2$, and $s \geq 2$ (20)

$I_t(s,P(AP(A))) =_{def} I_t(R^j(R^i(s)),A)$ for some $i \geq 1$ and for any j from i, such as $j \geq i+1$ (21)

$I_t(s,PP(P(A))) =_{def} I_t(R^i(s),A)$ for some $i \geq 2$, y $s \geq 2$ (22)

$I_t(s,PP(AP(A))) =_{def} I_t(R^i(s),A)$ for any $i \geq 2$, y $s \geq 2$ (23)

$I_t(s,PP(PP(A))) =_{def} I_t(R^2(s),A)$, and $s \geq 2$ (24)

$I_t(s,SP(P(A))) =_{def} I_t(s_0,A)$, where s_0 is first state (25)

$I_t(s,SP(PI(A))) =_{def} I_t(R^i(s),A)$ for any $i \geq 2$, (26)

$I_t(s,AP(AP(A))) =_{def} I_t(R^i(s),A)$ for any $i \geq 2$, (27) ■

A axiomatic systems $\mathbf{K}_t$ on the temporal language $\mathbf{L}_t$ is defined. This is composed by a set of the basic axioms, and a set of inference rules.

Let A, B, and C be wffs in $\mathbf{L}_t$, then the set of basic axioms are:

1.- $(A \Rightarrow (B \Rightarrow A))$
2.- $(A \Rightarrow (b \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
3.- $(\neg A \Rightarrow \neg B) \Rightarrow (B \Rightarrow A)$
4.- $(\forall x A) \Rightarrow A$ if x is not free in A.
5.- $(\forall x (A \Rightarrow B) \Rightarrow (A \Rightarrow \forall x B))$ if x is not free in A.
6.- $=(x,x)$
7.- $=(t_1,u) \Rightarrow (p(t_1,t_2,\ldots,t_1,\ldots,t_n) \Rightarrow p(t_1,t_2,\ldots,u,\ldots t_n))$.
8.- $PC(B) \Leftrightarrow (P(B) \vee B)$
9.- $P(A \vee B) \Leftrightarrow P(A) \vee P(B)$
10.- $PP(A \vee B) \Leftrightarrow PP(A) \vee PP(B)$
11.- $AP(A \vee B) \Leftrightarrow AP(A) \vee AP(B)$
12.- $P(A \wedge B) \Rightarrow P(A) \wedge P(B)$
13.- $P(A \wedge B) \wedge P(A \wedge C) \wedge P(C \wedge B) \Leftrightarrow P(A \wedge B \wedge C)$
14.- $PP(A \wedge B) \Leftrightarrow PP(A) \wedge PP(B)$
15.- $AP(A \wedge B) \Leftrightarrow AP(A) \wedge AP(B)$
16.- $\neg P(A) \Leftrightarrow AP(\neg A)$
17.- $\neg AP(A) \Leftrightarrow P(\neg A)$
18.- $\neg PP(A) \Leftrightarrow PP(\neg A)$
19.- $\neg(\neg A) \Leftrightarrow A$
20.- $P(P(A)) \Leftrightarrow P(PP(A))$
21.- $P(PP(A)) \Leftrightarrow PP(P(A))$
22.- $PP(AP(A)) \Leftrightarrow AP(PP(A))$
23,- $AP(AP(A)) \Leftrightarrow PP(AP(A))$
24.- $AP(PP(A)) \Leftrightarrow PP(AP(A))$
25.- $AP(A) \Rightarrow A_{(S_0)}$, where $A_{(S_0)}$ expresses the formula A in the first state.
26.- $AP(P(A)) \Leftrightarrow A_{(S_0)}$
27.- $AP(A) \Rightarrow P(A)$
28.- $AP(A) \Rightarrow PP(A)$
29.- $PP(A) \Rightarrow P(A)$. ■

If the systems in the state m, then the inference rules are:

a) Extension of the modus ponens rule:

1.- If A is in the state m, and $A \Rightarrow B$, then B in the state m is inferred.
2,- If A is in a state $n<m$, and $P(A) \Rightarrow B$, then B in the state m is inferred.
3.- If A is in the state m-1, and $PP(A) \Rightarrow B$, then B in the state m is inferred.
4.- If A is in a state $n \leq m$, and $PC(A) \Rightarrow B$, then B in the state m is inferred.
5.- If A is in any states $n<m$, and $AP(A) \Rightarrow B$, then B in the state m is infered.

b) The rules of the generalization:

1.- If y is not free in A, and $A \Rightarrow B(y)$, then $A \Rightarrow \forall x B(x)$ is inferred.
2.- If y is not free in A, and $A(y) \Rightarrow B$, then $\exists x A(x) \Rightarrow B$ is inferred.

The similar mode as the concept of deductive theory is defined in a relational language, further the concept of deductive temporal theory is defined in a temporal language. In the equal mode, let $\mathcal{P}$ be a program in the state m, when this is expressesed in a temporal language, then a temporal theory can be

associated in the state m.

Let $\mathbf{L}_t$ be a temporal language, $\mathbf{I}_t$ is an interpretation on $\mathbf{L}_t$, $\mathbf{K}_t$ is a axiomatic system over $\mathbf{L}_t$, then the axiomatic system $\mathbf{K}_t$ applied on a deductive temporal theory is consistent with respect to the interpretation $\mathbf{I}_t$. This is not difficult to prove [VIL92].

Our next goal is to specify in a temporal language a program $\mathcal{G}$ what this provided by a DCM of the an IS.

The concept reduction temporary is included to supply the translation of the program $\mathcal{G}$ from the relational language $\mathbf{L}'$ to the temporal language $\mathbf{L}_t$.

We can say that a program $\mathcal{G}$ is temporary reduced, if all the sentences in $\mathcal{G}$ are temporary reduced, and a sentence is reduced if there are one or two temporary domains only.

All program $\mathcal{G}$ can be translated to other program $\mathcal{G}'$ which are temporarily reduced and are semantically equivalent. This has been proved in [VIL91].

Thus, we explain the rules to translate a program from a relational language to a temporal language.

let $\mathbf{L}'=(\mathbb{A}',\mathbb{F}')$ be a relational language, with $\mathbb{A}'=\mathbf{A} \cup \mathbb{T} \cup \mathbf{P} \cup \mathbf{S} \cup \mathbf{V} \cup \mathbb{V}$, and let $\mathbf{L}_t=(\mathbb{A}_t,\mathbb{F}_t)$ be a temporal language, with $\mathbb{A}_t= \mathbf{A}\times\Xi \cup \mathbf{P}_t \cup \mathbf{S} \cup \mathbf{V} \cup \mathbf{O}$, such that for any predicates p with arity n in $\mathbf{P}$, then the predicate p with arity n-1 exists in $\mathbf{P}_t$, further, a biyectve correspondence is defined between $\mathbb{T}$ and Ξ. Let $\mathcal{G}$ be a program in $\mathbf{L}'$ which is pure and temporarily reduced, the rules to translate an other program $\mathcal{G}'$ in $\mathbf{L}_t$ are:

a) $p(x,t)\Leftarrow\phi(y,t)\wedge\phi'(z,t')\wedge t'<t$ is translated by $p(x)\Leftarrow\phi(y)\wedge\mathrm{P}(\phi'(z))$.
b) $p(x,t)\Leftarrow\phi(y,t)\wedge\phi'(z,t')\wedge t'\leq t$ is translated by $p(x)\Leftarrow\phi(y)\wedge\mathrm{PC}(\phi'(z))$.
c) $p(x,t)\Leftarrow\phi(y,t)\wedge\phi'(z,t-1)$ is translated by $p(x)\Leftarrow\phi(y)\wedge\mathrm{PP}(\phi'(z))$.
d) $p(x,t)\Leftarrow\phi(y,t)$ is translated by $p(x)\Leftarrow\phi(y)$.
e) If $a_1, a_2, \ldots, a_n \in \mathbf{A}$, and $\sigma\in\mathbb{T}$, then $p(a_1, a_2, \ldots, a_n, \sigma)\Leftarrow$ is translated by $p(l_1, l_2, \ldots, l_n)\Leftarrow$, where the set of constants $\{l_1, l_2, \ldots, l_n\}$ expresses the set of constants $\{a_1, a_2, \ldots, a_n\}$ in the state s, such that s is associated with an appropriate instant σ. ■

If $\mathcal{G}$ is normalized by transformation of LLoyd, then in the program translated $\mathcal{G}'$ the temporary operator AP is not present.

Let $\mathcal{G}$ be a program in $\mathbf{L}'$, $\mathcal{T}'(\mathcal{G})$ is the deductive theory associated with $\mathcal{G}$, $\mathcal{G}'$ is the program which has been translated from $\mathcal{G}$, $\mathcal{T}_t(\mathcal{G})$ is the deductive temporal theory associated with the program $\mathcal{G}'$, and Δ and Δ' denote some formula in $\mathbf{L}'$ and $\mathbf{L}_t$ respectively, such that Δ' is the translation of the Δ, and viceversa. If any formula Δ is valid in $\mathcal{T}'(\mathcal{G})$, then Δ' is valid in $\mathcal{T}_t(\mathcal{G}')$ too, and viceversa. This is a theorem which can be proved.

We can consider the specification of an information system as a program $\mathcal{G}$ expressesed in a temporal language $\mathbf{L}_t$.

The next objective is to review the concepts which have been included in PEM, and they will be redefined in the temporal language with the conditional, and where all feasible propriety of PEM will be retained.

In the rest of this section, all logic expressions are wffs in $\mathbf{L}_t$, that is to say, the temporary operators can appear in the expressions.

The concepts of the atom, basic atom, literal, derivate predicate, basic predicate, fact, and generic fact remain.

If (l_1,c,l_2) is a potential attandence or absence in $\mathbf{L}_t$, then l_1 and l_2 are literals in scope of the present temporary operator, which can be null, and c is an expressesion in $\mathbf{L}_t$.

The concepts of the potential chain remain too, such that $(h,c,\{l_1, l_2, \ldots, l_n\})$ is an element in G(h), where h and $\{l_1, l_2, \ldots, l_n\}$ are formed by literals in the scope of the present operator.

Under the temporal operator, a process of the simplification is introduced. In virtue of the axioms (10), (11), (13), and (14) in $\mathbf{K}_t$, the process of simplification clusters all subformulas under the scope of the same temporal operator (PP, AP) and gives one formula composed of all subformulas in the scope of the temporal operator. However, the subformulas in the scope of the operator P are not simplified.

The concept of the relation of the inclusion is extended to a formula simplified under the temporal operators, and it is redefined as:

Let c_1 and c_2 be two conjunctions of the literals in $\mathbf{L}_t$, and $\{c_1\} = \{c_1\}_c \cup \{c_1\}_{pp} \cup \{c_1\}_{ap} \cup \{c_1^1\}_p \cup \{c_1^2\}_p \cup \ldots \cup \{c_1^n\}_p$ and $\{c_2\} = \{c_2\}_c \cup \{c_2\}_{pp} \cup \{c_2\}_{ap} \cup \{c_2^1\}_p \cup \{c_2^2\}_p \cup \ldots \cup \{c_2^m\}_p$ the sets of the literals associated with these conjunctions. We can say that c_1 is included in c_2, which is denote as $c_1 \subset_t c_2$, if the following conditions are accomplished:

a) $\{c_1\}_a \subseteq \{c_2\}_a$.
b) $\{c_1\}_{pp} \subseteq \{c_2\}_{pp}$.
c) $\{c_1\}_{sp} \subseteq \{c_2\}_{sp}$.
d) For any i $\{c_1^i\}_p$ a $\{c_2^j\}_p$ exists, such that $\{c_1^i\}_p \subseteq \{c_2^j\}_p$. ■

Note: $\{c\}_a$ expresses a subformula under present, $\{c\}_p$ under past, $\{c\}_{pp}$ under recent past, and $\{c\}_{ap}$ under always past.

It is evident that the relation of the inclusion retains the same property as the relation of the inclusion over conjunctions in a relational language.

The concept of the more general closed substitution must be extended to determine the subformula over which it is applied. This concept is redefined as *ij-more general closed substitution over the predicate p and in the scope*

of the same temporal operator, which is denoted as $\theta_p^{ij}\#[\Delta(x),\Gamma(z)]$, where # is a temporal operator which delimits the subformulas where the predicate p can be found. In the event of the appearance of the symbol ≈, then the complete formula is considered, and if it is not, some symbols then we will assume the current operator.

Similarly, the concepts more general closed global substitute is extended with the temporal operator symbol, that is to say, $\Xi\#[\Delta(y),\Gamma(z)]$, where # represents a temporal operator symbol.

The concepts: derivate predicate, basic predicate, logic program, generic fact, level of the potential chains, pure program, and well constructed program remain the same as in the relational language.

3.2 Set of Potential Attandences and Absences Associated with an MCD of the Information System.

Our objective is to obtain the set of the potential chain associated with each generic fact, which has take place in present. Firstly, the potential attandences o absences which cause the occurrence of a generic fact in present must be obtained.

When a generic fact h(x) happens, at level zero, the effects can be deduced by two motives:

a) The sets $E_+^0(h(x))$ and $E_-^0(h(x))$ which represent all potential attandences and absences obtained when the generic fact h(x) appears under the temporal current operator.

b) Independent of the generic facts, and when formulas such as $p(y) \Leftarrow \&(\phi(z))$ appear, where & can be the temporal operators: P, PP, AP, then a potential attandence can be deduced, which is denominate as spontaneous potential attandences, and the set of the spontaneous potential attandence is denoted as $E^0(\varepsilon)$. $E^0(\varepsilon)$ is defined as:

$$E^0(\varepsilon)=\{(null,C,p(y)) \mid \exists(p(y)\Leftarrow\&(\phi(z))) \text{ in the program, } \&\in\{P,PP,AP\}, \text{ and } C \text{ is } \&(\phi(z))\}$$

at the upper level, in the event of (a), we can obtain the $E_+^n(h(x))$ and $E_-^n(h(x))$, and the event of (b) the $E^0(\varepsilon)$ causes $E_+^1(\varepsilon)$, $E_-^1(\varepsilon)$, ...,$E_+^m(\varepsilon)$, $E_-^m(\varepsilon)$. The number of levels is finite too.

3.3 Set of operations associated with a generic fact.

Let E and Eε be the sets of all potential attandences and absences associated with a program Θ, then a set of the potential chains can be obtained from E and Eε, such that any member in E∪Eε is associated with a potential chain. Let (l,c,i) be a member in E∪Eε, if i is a literal formed by an integrity predicate, then this member is included in the element of the potential chain.

The concept of the operation is defined similar to the form in the previous section.

In the set of the operations associated with a generic fact,the operations deduced from the potential chains which have been obtained by members in Eε must be included .

In the set of the operations associated with a generic fact, we can distinguish the subset of the permitted operations and the subset of the non-permitted operations.

Let (h(x),C,$\mathcal{A}$,$\mathcal{B}$) be an operation: this operation is not permitted if a literal formed by an integrity predicate is in the set $\mathcal{A}$. In $\mathcal{B}$, the literals formed by an integrity predicate can be omitted.

In the set of the operations, the relation of the inclusion can be defined in a similar way. For the minimum conditions to be certain or not, the atom formed by a derivate predicate can be found.

The concept of the correction of a logic program and the process of the reduction are maintained. The process of the reduction can be applied over the set of the potential chains too.

3.4 Specification of the DCM by a State Machine.

The concepts: potential characteristic state and characteristic state are retained. In the condition of any potential characteristic state, the temporal operators can appear over the literals which constitute this condition, and an order relation is established under time.

This new order relation over the literal causes a constraint on the set of the generic states which can be expected.

The non-permitted generic states are included. A generic state is a non-permitted generic state if a literal formed by an integrity predicate appears in this state.

With these considerations, the state machine is constructed in a similar form. Further, the transactional operations are defined.

From the state machine we can deduce when a generic fact cannot take place, if a transactional operation is executed and a non-permitted generic state has taken place.

4. Conclusions.

A process to translate the kernel of the logic specifications or program to a set of the transactional operations has been proposed. This process provides an other way to obtain the minimum or standard model of the Herbrand associated with this logic program. Further, we have extended this process to the DCM of the information system.

REFERENCES

[APT88] **Apt,K.R.; Blair,H.A.; Walker,A.** *"Towards a Theory of Declarative Knowledge"*, in [MIN88].

[LLO84] **Lloy,J.W.; Topor,R.W.**"*Making Prolog more expressive*", Journal of Logic Programming, 1(3), 225-40, 1984.

[LLO87] **Lloy,J.W.**"*Foundations of logic programming*", Springer-Verlag, 2nd, 1987.

[LLO87A] **Lloyd,J.W.; Sonenberg,R.W.; Topor,R.W.** "*Integrity Constraint Checking in Stratified Databases*", Journal Logic Programming 4,4 Dec, 1987.

[MIN88] **Minker J. (ed).** "*Foundations of Deductive Databases and Logic Programming*", Morgan Kaufmann Publishers, Inc, 1988.

[OLI85] **Olivé, A.** "*Modelización conceptual de sistemas de información*", VII Escuela de verano de informática, AEIA 1985.

[OLL82] **Olle,T.W.;Sol,H.G.; Werrijn-Stuart A.A.(Eds.)** "*Information systems design methodologies: A comparative review*", North-Holland, 1982.

[OLL83] **Olle,T.W.; Sol, H.G.; Tully C.J. (Eds).** "*Information systems design methodologies: A feature Analysis*", North-Holland, 1983.

[SAN90] **Sancho M.R.** "*Deriving an internal events model from a deductive conceptual model*", Report interno de la Unv. Pot.de Cataluña, 1990.

[SIS87] **Sistac,J.** "*Constr u ccio automàtica de prototipus de sistemes D'informació a partir de models conceptuals deductius, descrits ambs el i.lenguatge 'DADES'*", Tesis Doctoral de Unv. Poltec. Cataluña, 1987.

[THY88] **Thayse, A.(ed).** "*From Standard Logic to Logic Programming*", Johm Wiley & Sons, 1988.

[VIL91] **Villena, S.; Clares, B.** "*Transformacion de especificaciones lógicas de sistemas de información a unas pseudo-operacionales*", ProDe'91, 2 a 4 Octubre Torremolinos (Málaga), 1991.

[VIL92] **Villena,S.**"*Definición de un lenguaje temporal para expresar la especificación de un Sistema de Inforación mediante una Modelización Conceptual Deductiva*", Informe interno del Dpto. Lenguajes y Sistemas Informáticos de la U.Granada, 1992.