On some inferences based on stratified forward chaining: An application to e-Government

El-Hassan Bezzazi

IREENAT, University of Lille 2, France `bezzazi@univ-lille2.fr`

Summary. This paper introduces first an expert system shell based on stratified forward chaining. The stratified forward chaining was proposed as a generalization for the inheritance networks with exception since it allows more than one antecedent in the premises part of a rule. Expert systems built with this tool are rule based and the user interface is web based. The practical use of this chaining is presented here through the three chaining methods: forward chaining, backward chaining and mixed chaining. Our current aim is to make use of these inference mechanisms to help the user in his interaction with administration to identify in an efficient way the relevant information he needs. This is done through a question-response dialog to determine the user profile. The paper considers also in a second part a special kind of inference called deontic inference. The rationale behind this kind of inference is the possible presence of deontic rules which may impose a change in the base of current facts to make it comply with the obligations they prescribe.

1 Introduction

This paper is twofold. It introduces first an expert system shell whose inference engine is based on stratified forward chaining presented in detail in the paper [1]. The stratified forward chaining (hereafter *sfc*) was proposed as a generalization for the inheritance networks with exception since it allows more than one antecedent in the premises part of a rule. In [2] *sfc* was tuned to what was called specific stratified forward chaining. Precisely, rules which have more literals in their body are said to be more specific than those with a subset of these literals in their body under some conditions. This way, more specific rules can easily be added to the rulebase and will prevail over more general ones in case they have conflicting conclusions. Expert systems built with this tool are rule based and the user interface is web based. The chaining *sfc* allows to manage the application of conflicting rules (i.e. those whose conclusions are opposed) according to the same intuition as the one in inheritance networks and which is expressed by the rule of preemption [5]. The practical use of *sfc* is presented through the three chaining methods: forward chaining, backward chaining and mixed chaining. Our current aim is to make use of these inference mechanisms to help the user in his interaction with public administration to identify in an efficient way the relevant information he needs. For example, when a user apply for some administrative document, the system would help him know the required documents [3]. This is done through a question-response dialog to determine the user profile. This kind of dialog exempts the user from spending time for extracting the relevant information out of the general available documentation. On the other hand, the fact that the user-system interaction is mainly based on yes/no questions makes it particularly well suited for mobile government, when using mobile phones which have, in general, small screens. The GPL programming language PHP [4] was used to implement the inference engines in our prototype. In its second part, this paper considers also a special kind of inference built on *sfc* called deontic inference. The rationale behind this kind of inference is the possible presence of deontic rules which may impose a change in the base of current facts to make it comply with the obligations they prescribe. The paper is organized as follows. In section 2 we present the language used for knowledge representation and in a rather informal way the specific stratified forward chaining. In sections 3 and 4 we define respectively the stratified backward chaining *sbc* and the stratified mixed chaining *smc* which comply with *sfc*. An example of dialog using *smc*. Section 5 begins the second part of this paper by introducing deontic rules and deontic inference as well as a measure to evaluate how far or how close is a situation described by a set of literals to the ideal situation computed by deontic inference. In section 6 we discuss a more general characterization for deontic inference. In section 7 we give some examples of deontic rules and inferences.

2 Knowledge representation, Cautious union and specific stratified forward chaining

A literal is an atom or an atom preceded by ! which represents negation, it is then a negative literal . Currently, an atom is simply a symbol but atoms with more structure such as comparisons could be easily incorporated in the system. The shape of a rule is: $l_1\ l_2 ... l_n > l_m$ where $l_1\ l_2 ... l_n$ and l_m are literals. Literals are separated by one space or more in the body of the rule and the conclusion is a single literal.
A knowledge base is identified by a name and has a base of rules and a base of facts.
The function *lit* identifies and puts in the set *literals* the literals which appear in the knowledge base.
The function *clean* removes from a set of literals all opposing literals. The function *cu* is a non commutative operation that carries out the cautious union of two sets of literals by adding to the first set only those literals in the second set which do not have their opposites in the first set and cleaning the

resulting set. Let L, L' be two sets of literals. $clean(L) = L - \{l, !l : l, !l \in L'\}$ $cu(L) = clean(L \cup \{l \in L' :!l \notin L\})$
The stratification of a rulebase is carried out by the function s. The stratification mechanism consists in computing for each literal of the rulebase its stratum. The stratum of a literal is, roughly speaking, the biggest number of one-step forward chaining that infers it [1]. The stratum of a rule is the biggest stratum of its body. Note that a stratified rulebase is necessarily acyclic when thought of as a graph. We give, in what follows, an informal description for this inference based on two partial orders $\prec_1$ and $\prec_2$. A rulebase is said to be stratified by the mapping $s : lit(R) \longrightarrow [1, n]$ if and only if for any rule in R, $l_1, ., l_n > l_m$ we have $s(l_i) < s(l_m)$ and there is no other such mapping $s' : lit(R) \longrightarrow [1, n']$ with $n' < n$. This stratification induces a partial ordering on R defined by $r \prec_1 r'$ iff $max(s(body(r)) < max(s(body(r'))$. Let $\prec_2$ be the partial ordering that specificity defines, $r \prec_2 r'$ if and only if $body(r') \subseteq body(r)$ and $r \not\prec_1 r'$. We combine these two orders to define a new partial order $\prec_p$ as $r \prec_p r'$ iff $r \prec_1 r'$ and $r' \not\prec_1 r$. In other words, $r \prec_p r'$ iff $r \prec_1 r'$ and either r is more specific than r' or they do not compare to each other. Let $\prec_t$ be any total ordering on R which is consistent with $\prec_p$, *i.e*, if $r \prec_p r'$ then $r \prec_t r'$. Specific *sfc* inference, starting from a rulebase $R = r_1, ..., r_n$ with $r_1 \prec_t ... \prec_t r_n$ and a set of initial facts L, proceeds as follows: Among these totally ordered rules we look for the first rule r which can be fired by L, *i.e.* such that $body(r) \subseteq L$. If such a rule does not exist, the process is done, otherwise the process continues with the new totally ordered rulebase $R - \{r\}$ and the set of facts $L \cup \{head(r)\}$. Notable properties for sfc are, given two sets of literals L, L':

- Non-monotony *i.e.*, in general $sfc(L) \nsubseteq sfc(L \cup L')$
- $L \subseteq sfc(L)$
- If L is consistent then so is $sfc(L)$

The function sfc computes the literals inferred using stratified forward chaining and retains at the same time the rules which were applied with relevance in the inference, i.e. the rules which were applied and whose conclusion was actually kept (and not rejected during the cautious union by the opposite literal previously inferred). The relevant rules will help us define in the sequel the stratified backward chaining.

3 Stratified Backward Chaining

Backward chaining is a bottom up method used in expert systems. It starts with a literal called a goal to see, in case it is not an initial fact, if there are rules that support it and so on for the literals in the body of these rules which become sub-goals to be confirmed. The goal is confirmed when all the ultimate sub-goals are among the initial facts. Because of the fact that this method is goal-driven, it will not use all the rules that would have been used to check a given goal by forward chaining. However, attention must be paid to possible cycles that may exist in the rulebase to guarantee the termination of the process. Fortunately, this would never be the case as far as we are dealing with stratified rulebases. This is the classical definition of backward chaining with respect to classical forward chaining. The definition we are about to give now of backward chaining which we will call stratified backward chaining (*sbc* in short) is made with respect to *sfc*. Unlike the definition of classical backward chaining which does not use forward chaining, *sbc* does use *sfc*. Indeed *sfc* allows the selection of the relevant rules *i.e.* those which conclusions are inferred with respect to *sfc* and the initial facts. Besides stratified backward chaining uses two functions *sbc* and *prove* which are executed according to a cross recursion (they call mutually each other). In fact, the function *sbc* selects among the relevant rules those which are especially relevant for the goal being processed.

Consider the following fragment of a rulebase which describe documents required to apply for a passport. There is an exception which is stated in case the application is made for a modification, fore example a new born child to be added into the passport. In this case the stamp, denoted by doc_7, is no more required.

```
passport_renewal > passport_application;
passport_modification > passport_application;
passport_application > documents;
documents > doc\_1;
.........................................
documents > doc\_9;
passport_modification > !doc\_7;
```

If we take as initial facts the only fact passport_modification and as a goal the fact doc_7, the classical backward chaining will conclude on a success whereas we would rather like it to conclude on a failure in accordance with *sfc*. The solution consists in checking before concluding on a success with the classical backward chaining that the rules which made it possible to deduce the goal are relevant for *sfc*.

```
function prove(list_of_goals)
{if (list_of_goals is empty) then return(true);
else if (sbc(head(list_of_goals)))
return(prove(tail(list_of_goals)));
else return(false);}

function sbc(goal,list_of_relevant_rules)
{if (goal in current_facts)) then success=true
else if (list_of_relevant_rules is empty) then success=false;
      else if (goal!=conclusion(r))  then success=(sbc(goal,tail(list_of_rules)));
             else {if (prove(premises(r))) then success=true;
                 else success=(sbc(goal,tail(list_of_rules)));
 }
return(success)}
```

list_of_relevant_rules is the set of relevant rules in the rulebase for *sfc*. current_facts stocks the facts entered by the user and which initially constitute the fact base.

Proposition 1. *Given a stratified rulebase,* $sbc(goal, 0)$ *returns a success if and only if* $goal \in sfc(current\_facts)$.

4 Stratified Mixed Chaining

Mixed chaining is an inference method using the forward chaining to infer new facts and backward chaining to confirm facts possibly by questioning the user. The stratified mixed chaining carried out by the function *smc* that we present here combines the forward and backward stratified chaining. In general, it works through a dialog with the user during its execution. The dialog relates exclusively to the truth value of certain facts among the initial facts. It is the function *backward_all* which chooses the facts to be questioned on. The function *sfc* is called in the treatment of the form submitted as the answer to each question. It is interesting to notice here that the forms sent to the user during this dialog save at the same time the execution environment of the function *smc*. Indeed, the fact that the http protocol is stateless, the html page sent to the user carries as hidden fields all the data necessary to save the recursion context. Questions are put only about positive literals i.e. if it is a negative literal on which the user must be questioned, the question will be put rather on the opposite literal which then gives the answer for the literal in question. The questioning to be sent to the user through an html form is prepared by the function *to_ask_user*.

```
function smc(list_of_rules,goal)
{n=0;dialog=false;
while (n<number_of_rules and !dialog);
{r=list_of_rules[n];
if (conclusion(r)=goal)
if ((backward_all(premises(r)))) return(true);
n=n+1;
}
return(false);}

function backward_all(list_of_goals)
{dialog=false;
if (list_of_goals is empty) return(true);
else {fact=head(list_of_goals);
  if (fact in result) return(backward_all(tail(list_of_goals));
  else if (inverse(fact) in result) return(false);
    else if (initial(fact) and not(asked(fact) or asked(inverse(fact))))
{to_ask_user(fact);sfc();dialog=true;}
 else if(smc(fact)) return(backward_all(tail(list_of_goals)));
 else return(false);}}
```

number_of_rules is the number of rules in the rulebase.
dialog is a boolean variable used to simulate the mutual recursion between the two functions.
result stocks the facts deduced with the function sfc.
initial returns true if the literal occurs only in the premises false otherwise.
asked returns true for a literal which the user was asked about its truth during program running.

Proposition 2. *Consider a session of smc which has concluded either on a success or not and let answers denote the set of literals added to current_facts through the session dialogue by the user's answers. Given a stratified rulebase, smc(goal) returns a success if and only if goal $\in$ sfc(current_facts $\cup$ answers).*

We give here an example of a dialog which the reader will be able to test at the address: http://droit.univ-lille2.fr/eadministration/exp.php We give the rulebase worked out starting from the government texts published in the web site: http://vosdroits.service-public.fr/particuliers/N360.html. The rule

```
renewal out-of-date<2 childs childs<=4  > validity-5-years
```

must be read as follows. If the case is about a renewal of an out-of-date passport since less than two years and that the children must be recorded there and that their number is does not exeed four then the validity duration of the passport will be of five years.

```
emergency > !stamp_60;
emergency > stamp_30;
emergency > validity_6_months;
emergency > documents;
renewal out_of_date<2 childs childs<=4 > documents;
renewal out_of_date<2 childs childs<=4 > docs_childs;
renewal out_of_date<2  childs childs<=4 > valid_5_years;
renewal out_of_date<2  childs !childs<=4 > documents;
renewal out_of_date<2  childs !childs<=4 > docs_childs;
renewal out_of_date<2  childs !childs<=4 > demand_elders;
renewal out_of_date<2  childs !childs<=4 > valid_5_years;
renewal out_of_date<2  !childs > documents;
renewal out_of_date<2  !childs > validity_10_years;
renewal !out_of_date<2  > first_demand;
modification > documents;
modification > !stamp_60;
modification > current_validity;
documents > form;
documents > 2_photos;
documents > stamp_60;
documents > proof_residence;
documents > identity;
```

The following rules are used to define the final facts among which the inferred facts should be chosen for the answer.

```
form > goal; 2_photos > goal;
stamp_60 > goal; stamp_30 > goal;
proof_residence > goal;
identity > goal;
tablissement > goal; demand_elder > goal;
documents_childs > goal;
valid_10_years > goal; valid_5_years > goal;
valid_6_months > goal;
current_validity > goal;
```

An example of a session.

```
emergency ?
no
renewal ?
no
out_of_date<2 ?
yes
childs ?
yes
childs<4 ?
yes
```

The dialog will stop at this point and the system will conclude on the following facts:

```
documents_childs valid_5_years form 2_photos
stamp_60 proof_residence identity
```

5 deontic rules

Deontic logic is a logic to reason about ideal and actual behavior [11]. Applications of deontic logic can go beyond legal analysis and legal automation to cover other domains like the specification of security policies and fault tolerant systems [10]. Traditionally, this logic is developed as a modal logic with, essentially, a modal operator o to define obligation which can be used in its turn to define permission and prohibition. However Instead of using such an operator and in order to stay at a propositional level, we enrich the language by considering a new kind of literals which are intended, if inferred, to describe an obligation to be respected. These literals, we shall call deontic literals, are of the form $o(l)$ or $!o(l)$ where l is a descriptive literal, *i.e.* a literal defined as in section 2. Let O be the set of deontic literals and Lit_D the set of obligated literals *i.e.* $l \in Lit_D$ if and only if $o(l) \in O$. The function ω returns the obligated literal of a deontic literal *i.e.* $\omega(o(l)) = l$ and the function o returns the deontic literals of a set of literals, $o(L) = L \cap O$. An operation of cautious

union *cu'* is defined for two sets of deontic literals L_D, L'_D in the following way. $cu'(L_D, L'_D)$ adds to the first set L_D only those literals in L'_D which do not have their opposites in the first set or which oblige a literal whose opposite is obliged in L_D.
A deontic rule is a rule where the body is made out of descriptive literals and the conclusion is a deontic literal. Therefore, we shall deal from now on with two kinds of rulebases: A descriptive rulebase R and a deontic rulebase D. Note that the set of rules $R \cup D$ is stratified since R is stratified and the heads of the newly added rules never occur in the rules body.

5.1 deontic inference and legal sets

Given a set of literals L, we define elementary deontic inference $edi(L, R \cup D) = sfc(L, R \cup D)$. In the sequel, we shall drop the parameter $R \cup D$ from both $edi(L, R \cup D)$ and $sfc(L, R \cup D)$ in order not to encumber the notation. The result of this inference is a set of literals L' which may contain deontic literals. A set of literals L is said to be legal or equivalently D-consistent when $sfc(L)$ does not contain a positive deontic literal which is violated by one of its descriptive literals or conflicted by a contrary obligation: L is legal if and only if $o(l) \in L \Rightarrow !l \notin L \wedge o(!l) \notin L$. A set of literals which is not legal is called illegal. inference from an illegal set L yields an illegal set L' since $L \subset L'$. An elementary legalization $\lambda(L)$ for L is an update of L by replacing l with $!l$ whenever $l \in L$ and $o(!l) \in sfc(L)$, in other words $\lambda(L) = cu(\omega(o(sfc(L))), L)$.

The legalization of an illegal set L consists in a sucession of elementary legalizations. In order to to be able to cumulate along this process the deontic literals in accordance with their cautious union, we need to keep track of them.

$$\begin{cases} \lambda^{(0)}(L) = L, \Delta^0 = \emptyset \\ \lambda^{(n+1)}(L) = cu(\omega(\Delta^{(n+1)}), \lambda^{(n)}(L)), \Delta^{(n+1)} = cu'(\Delta^{(n)}, \omega((o)(sfc(L)))) \end{cases}$$

Since the set Lit of all literals is finite, the sequence $\{\lambda^{(n)}(L)\}_n \in N$ is such that there is necessarily an integer i s atisfying $\lambda^{(i)}(L) = \lambda^{(i')}(L)$ for some $i' > i$. Let k and k' be the smallest integers satisfying $\lambda^{(k)}(L) = \lambda^{(k')}(L)$ with $k' > k$. Two cases are to be distinguished:
1) The case where $k = k'$ which means that the sequence $\{\lambda^{(n)}(L)\}_{n \in N}$ is stationary. We define then the deontic inference of L to be $di(L) = edi(\lambda^{(k)}(L))$
2) The case where $k \neq k'$ which means that the sequence $\{\lambda^{(n)}(L)\}_{n \in N}$ is cyclic.
This will be considered as an inconsistency and the deontic inference is undefined $di(L) = \perp$. This means that the set of deontic rules D does not provide the means for handling the case described by L.
Deontic inference defined this way incorporate defeasible reasoning thanks to stratified forward chaining which operates at the descrive level as well as at the deontic level. The issue of dealing with conflicting and conditional obligations in deontic logic at the light of non-monotonic reasoning has been discussed

in several papers [6][8][9]. In particular, in [8] the author recommends the use of an already existing non-monotonic and in [9], the author underlines the difference between an obligation which is defeated and an obligation which is violated. logic with a deontic logic instead of a built-in non-monotonic deontic logic. The elementary legalization plays a crucial role in deontic inference. it takes the deontic output of sfc and reuses it to correct the input, possibly altering it by replacing some of its literals by their opposites, to make it comply to the obligations of the rulebase. Making a difference between the declarative statements and the norms is an approach already singled out in [7] where the set of conditional norms is seen as a black box which transforms the input into output following some basic rules.

5.2 Legality degrees

Let us consider now the case where $di(L) \neq \perp$. $di(L)$ describes the ideal situation to which L must comply whereas $sfc(L)$ describes the actual situation. The similariry between the actual situation and the ideal situation can be evaluated with well known similarity measures for finite sets such as the jaccard measure. We define the degree of legality of a set of literal L with respect to the rulebases R and D as

$$\delta(L) = \frac{|sfc(L) \cap di(L)|}{|sfc(L) \cup di(L)|}$$

.

6 A more general characterization for legal sets

Let L, R, D, Lit be respectively a set of literals, a descriptive rulebase, a deontic rulebase and the set of all literals appearing in these sets. Let Lit_D be the set of obligated literals *i.e.* $l \in Lit_D$ if and only if $o(l) \in Lit$. The issue we investigate in this section is, given a consistent set of literals L, in case it is not legal, which legal sets of literals, if any, could be proposed as "legalisations" for it. A model for L with respect to $R \cup D$ is defined as being any consistent set $\mathcal{L} = cu(L_D, L)$ of literals with $L_D \subseteq Lit_D$ such that:
(1) $sfc(mathcalL)$ is D-consistent and
(2) for any $l \in L_D$ there is $L_l \subseteq L_D$ such that $l \in \omega(sfc(\mathcal{L}_\updownarrow, R \cup D))$ where $\mathcal{L}_\updownarrow = cu(L_l, L)$

Proposition 3. *Let L be a consistent set of descriptive literals. The legalized set obtained by deontic inference when defined is a model for L.*

Proof. The legalized set given by deontic inference is $\mathcal{L} = \lambda^{(k+1)}(L) = cu(\omega(\Delta^{(k+1)}), \lambda^{(k)}(L))$ for some $k \in N$. We put $L_D = \Delta^{(k+1)} = cu'(\Delta^{(k)}, \omega(o(sfc(L))))$.

$\mathcal{L}$ is D-consistent since it is the legalization of L by di. On the other hand, property (2) is satisfied since the sequence $(\Delta)_{i\in[0,k]}$ is an increasing chain.

```
a>o(c)
b>o(!c)
```

there is no model for $\{a, b\}$.

```
a>o(!b)
b>o(!a)
```

$\{a, !b\}, \{!a, b\} and \{!a, !b\}$ are models for $\{a, b\}$. Indeed, the last one for example is the result for $cu(\{!a, !b\}, \{a, b\})$, on the other hand $sfc(\{!a, !b\}) = \{!a, !b\}$ and $!a \in sfc(\{!a, b\}) = \{!a, b, o(!a)\}$ and $!b \in sfc(\{a, !b\}) = \{a, !b, o(!b)\}$

6.1 Examples

We comment in this paragraph some examples on the use of deontic inference and legal sets.

No smoking

Consider a building where smoking is forbidden. However, smoking is allowed in the room number 4 of this building.

```
room4 > building;
building > o(!smoking);
room4 > !(o(!smoking));
smoking > pollution;
```

The sets {room4, smoking}, {room4, !smoking}, {building, !smoking} and {room4, building, smoking} are legal sets. The set{building, smoking} is an illegal set and its legalization yields {building, !smoking}.
Note that both of $\mathcal{M}_\infty = \{building, !smoking, !pollution\}$ and $\mathcal{M}_\in = \{building, !smoking, pollution\}$ are models for $R \cup D$. Indeed $sfc((L_1)) = \{building, !smoking, !pollution\}$.
Consider the set $L = \{building, smoking\}$. $sfc(L, R \cup D)$ is not D-consistent. Let $\mathcal{L} = cu(!smoking, L) = \{building, !smoking\}$.
$\mathcal{L} = \{building, !smoking, o(!smoking)\}$ is D-consistent.

Privacy

Let bob_privacy denote personal data of Bob. Agency A is not allowed to have access to Bob privacy unless Bob is okay. If it happens that Agency A did have access to Bob privacy, it is committed not to communicate it to others. Note that as long as the information cannot be erased from his mind, there must be a rule in the rulebase asserting that no obligation can be made to change this state. Corrective actions as sanctions may be prescribed in this case. This is an example of the so-called contrary-to-duty norms in deontic logic where violation of some obligations must be tolerated in favor of other appropriate obligations which become operative.

```
agency_A > o(!bob_privacy);
agency_A bob_okay > !o(!bob_privacy);
agency_A bob_privacy > !o(!bob_privacy);
agency_A bob_privacy > o(!communicate);
agency_A bob_privacy !bob_okay> o(sanction);
```

Cyclic

A simple example of a cyclic case is the following one where D states that light must be on if it was off and the switch pressed and similarly must be off if it was off and the switch pressed.

```
light press > o(!light)
!light press > o(light)
```

Considering $L = \{light, press\}$ as our starting set of literals, the legalisation of L by the first elementary deontic inference gives $\{!light, press\}$, the following step gives $\{light, press\}$ which shows the starting of a cyclic inference. This is explained by the fact that the literal press hide some crucial information to put the light on or off. Actually it should be replaced by two literals $press_0$ and $press_1$ for stating respectively the facts of putting off and putting on the light.There is no model for L. This example shows a need to enrich the language by the concept of action to deal with discrete event systems.

7 Conclusion

The first purpose of this paper was to introduce an effective implementation of inference engines based on a non-monotonic logic. Our system enjoys of the conceptual simplicity of systems based on propositional logics. As a matter of fact the examples given in the literature to introduce and motivate the concerns of non-monotonic logics are mainly written in a propositional language, this is why we restrict our definitions to propositional logics in addition to the fact that interesting classical expert systems based on propositional logics do exist and are effectively used. However, we project to investigate on the use of a kind of first order predicate logics as in prolog clauses for writing the knowledge base rules. A methodology should also be investigated to help define the specificity of rules with respect to each other during the rulebase construction. The second part of this paper investigated the use of deontic literals to handle normative statements. It was noticed that a set of literals may have several sets as candidates for its legalization in the interpretation structure and that the legalization preferred in some cases by deontic inference was severe in comparison to other possible legalizations. This issue must be investigated further in order to make the interpretation more adequate for a possible completness theorem to hold. Another promising issue is to consider the concept of action in the language as long as the system encompass the concept of change in the inferred literals.

References

1. Bezzazi, E-H, Revision and Update based on Stratified Forward Chaining in Frontiers in Belief Revision, edited by Mary-Anne Williams and Hans Rott, 2001 Kluwer Academic Publishers. (2001) 315–332
2. Bezzazi, E-H, Specific Stratified Forward Chaining, Proceedings of the International Conference on Artificial Intelligence, Las Vegas, Nevada, USA, (2000) 1455–1460
3. Bezzazi, E-H, Workflows et Systèmes experts dans l'administration électronique, Communication au Colloque International Administration électonique et qualité des préstations administratives, Lille, France, 11/19/2004
4. The official site of PHP: http://www.php.net
5. Horty, J., Thomason, R., Touretzky, D., A skeptical theory of inheritancein nonmonotonic networks, Artificial Intelligence, **42** (1990) 311–348
6. Horty, J., Nonmonotonic foundations for deontic logic, In Nute, D., ed., Defeasible Deontic Logic. Kluwer. 17–44.
7. Makinson D., Van Der Torre L., Input-output logics, Journal of Philosophical Logic, vol. 29, 2000, p. 383-408
8. Prakken, H., Two Approaches to the Formalisation of Defeasible Deontic Reasoning, Studia Logica 57 (1), pp. 73-90, 1996.
9. Van Der Torre L., Violated obligations in a defeasible deontic logic, In A.Cohn (ed.), Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI'94), pages 371-375, John Wiley & Sons, 1994
10. Wieringa R.J., MeyerJ.-J.Ch., Applications of Deontic Logic in Computer Science: A concise Overview, In Deontic Logic in Computer Science, pp 17-40, John Wiley & Sons, Chichester, England, 1993
11. Von Wright G.H., An Essay in Deontic Logic and the General Theory of Action, Acta Philosophica Fennica, Fasc. 21. North-Holland, 1968.