Specifying a Security Policy: A Case Study

Frédéric Cuppens and Claire Saurel
ONERA-CERT
2 Av. E. Belin
31055, Toulouse Cedex
France
email: {cuppens,saurel}@cert.fr
fax: +(33) 62 25 25 93

Abstract

The objective of this paper is to assist the security administrators, in their attempt to specify, define and formalize security policies suited to a given high risk environment. It is then possible for the administrators to automatically derive consequences of these policies. In particular, we want to provide users with the following functionalities:

- Query a given security policy.
- Verify that properties such as consistency and completeness are enforced by a given policy.
- Verify that a given situation does not violate the security policy.
- Investigate interoperability problems between several security policies.

In this paper we more precisely focus on the problem of security policies formalization. We want to get a generic approach, being as much domain-independent as possible. In order to achieve the above goals, we have chosen a logic-based approach. It combines a deontic logic to model the concept of permission, obligation and prohibition with a modal logic of action. It also includes the possibility to deal with additional concepts such as role, responsibility and delegation. We shall illustrate this approach through a case study: a regulation whose purpose is to define means to protect secret data related to the National Defense.

1. Introduction

Given a system to be designed, the final issue is to provide the security administrators with security policy specifications, in order to efficiently and sufficiently protect the system against any threat of its security properties (confidentiality, integrity and availability). Not surprisingly, several successive steps are necessary for a security methodology:

- The first one consists of a risk analysis: which are the threats we want the system to be protected against? What are their characteristics? This first work is generally not an easy task: a lot of information may be implicit and informal, so that it may be quite long and fastidious to extract an exhaustive set of threats.
- Given a set of threats, the second step consists in specifying a security policy to combat them. A security policy may have various abstraction level according to ITSEC classification [6]: it may be a Corporate Security Policy, a System Security Policy, or a Technical Security Policy. All organizations generally have security standards that apply to all systems within the organization and define the security relationship between the organization and the outside world. Using the ITSEC terminology, these standards represent the Corporate Security Policy. The System Security Policy then specifies the security measures that regulate how sensitive information and other resources are managed, protected and distributed within a specific system. These security measures are to be defined in a way which is consistent with the Corporate Security Policy. Finally, the measures of a System Security Policy that apply to the hardware and software of an Information Technology system may be separated from the remainder of the System Security Policy and defined in a separate document: the Technical Security Policy.

The work related in this paper focuses on the second step, especially the formalization and analysis of an already defined security policy. We want our methodology to be as much domain-independent as possible, and reusable. We
also want it to fit with all levels of abstraction a security policy may have. In particular, it should be applied to specify a Corporate, a System or a Technical Security Policy.

In our approach, we assimilate a security policy to a specific case of regulation. A regulation may be viewed as composed of agents, events and objects of the system to be regulated; it aims at defining what actions the agents are permitted, obliged or prohibited to do. We distinguish between two classes of constraints:

1. The constraints to be enforced by the agents when they perform actions on the system objects.
2. The constraints to be enforced by agents when they interact with other agents. In this case, the regulation may use various concepts such as the concepts of responsibility, delegation, hierarchical authority and so on...

In order to meet all former constraints, we choose a logic-based approach because of the following triple advantage:

- It represents a universal formal language, useful for expressing any kind of knowledge and data. We shall illustrate how it may be used in the case of a regulation.
- The possibility of formal calculus methods it may support, useful to help users to reason about the regulation.
- The existence of running tools which support a logical theory (several kinds of PROLOG and other theorem provers).

The remainder of this paper is organized as follows. We first introduce a case study we shall use to illustrate our approach. We define the deontic language we use to formalize policies, and show how our case study is formalized within it. We then present several types of practical applications that can be developed using such a formalism. We finally propose a comparison with other related works and a discussion of some difficulties that raised along this work of formalization, and which represent further work that remains to be done.

2. A case study

Our approach is illustrated with a case study which may be viewed as an example of Corporate Security Policy. It applies to the context of the multilevel security policy, deals with protection of both sensitive information and people cleared at a given security level having a need to know of some sensitive information. It is quite a large set of paragraphs which defines various concepts necessary for the protection, and gives rules to be applied in order to protect vulnerable people or information. Its size is about 60 pages in natural language. Our illustration only deals with the information protection part.

What are the main concepts used to specify this security policy? Actually, as the document uses a quite large subset of natural language, its expressiveness is very rich. Most of classical deontic concepts, such as obligation, permission and prohibition, are present. We give some examples:

- The transmitter of a classified document is obliged to update the document classification as soon as it is possible, i.e. immediately after the transmitter estimates the document classification is obsolete (that means, the result of its classification evaluation does not fit with the classification of the document).
- Any agent who is not the transmitter of a classified document is prohibited to change the classification of this document.
- The holder of a classified document is permitted to ask the transmitter to revise the classification of this document.

The instruction uses other organizational concepts such as the concepts of responsibility and delegation. For instance:

- Every organization which holds some secret documents is obliged to designate an agent who is responsible for preserving these documents.
- If it is decided to send a classified document, then the sending of office is obliged to establish a consignment note of this document. The head of this office is obliged to sign this note. The head of this office is permitted to delegate the obligation to sign the note to one of his representatives.

We also have to represent the concepts of actions and events. In our model, the execution of an action is associated with the occurrence of an event, in the sense that, by performing an action, an agent causes an event to occur, which in turn will possibly cause other events to also occur. For instance:

- Every time a document classified at the secret level is destroyed (action), the holder of this document and a witness of the destruction (corresponding event) are obliged to sign the destruction minutes.
- When a meeting is finished (event: meeting), its organizer is obliged to incinerate all the preparatory documents of the meeting.

One can also notice that a temporal representation is necessary: we have to represent temporal structure to represent
intervals such as before, during, and after the execution of an action. For instance:

- **Before** every meeting, the organizer of this meeting is obliged to establish a list of all participants in this meeting.

- **During** the time an agent is elaborating a document classified at the secret level, he is obliged to work in a protected area.

- **After** a secret document is destroyed, the document holder is obliged to notify the document author of this destruction.

Finally, there are rules in our case study which deal with situations of violation. For instance:

- If the organizer of a meeting does not incinerate all the preparatory documents of this meeting when the meeting is finished, then this organizer is obliged to bring it about that these documents are kept in a safe.

This last rule corresponds to a situation where the above primary obligation to incinerate the preparatory documents is violated.

### 3. Language definition

This section describes the language we propose to deal with all the notions we found in our case study and their associated axiomatics. Due to space limitation, we do not develop a semantics for this language in this paper but it is partially described in [8].

Our language is based upon a first order modal logic. To represent a regulation, we have to represent objects, events, actions and agents. We also need to introduce the concept of delegation. Then we have to represent the deontic concepts of obligation, permission and prohibition. For this purpose, we propose to use a deontic logic because of its advantage on other logical formalisms, for instance alethic logic (necessary and possible) or temporal logic, to reason about a situation where a rule is violated. The last rule of section 2 provides an example of such rules. They are generally called contrary to duty norms (see for instance [17, 4]). We actually found many examples of this type of rules in our case study and deontic logic was well suited to model them.

#### 3.1. Objects

We take our inspiration from the object-oriented concepts. An entity is represented by an object. Any object belongs to a “class of objects” which is an abstraction of it. The set of classes is structured in a hierarchical way; a class may inherit from another class. By doing so, we can represent different abstraction levels. A class of objects is defined by:

- its name,

- the names of the classes it possibly inherits from,

- its attributes; attribute values are objects (so that we can define structured objects),

- and a set of methods one can apply to the objects, in order to manipulate the object or to create another object.

In our logical language, a class of objects is represented as follows:

- name: a predicate symbol,

- attribute: a function; its signature provides the type (i.e. the object class) of the attribute value,

- method: an action; its signature provides the types of input and output parameters of this method.

Given a regulation, we shall represent its agents and objects as entities. For instance, we give the (partial) definition of the class *Classified_Document*:

**Class:** Classified_Document

**Inherits:** Document

**Attributes:** Classification : Level

Consignment Note : Note

**Methods:**

- Change_Classification : Classified_Document -> Level
- Establish_Consignment_Note : Classified_Document -> Note

Level and Note are supposed to be two already defined object classes. In our language, a particular entity will be represented by a constant symbol. For instance, if D337 is a particular classified document, then we have: Classified_Document(D337).

#### 3.2. Events and temporal structure

The concept of event is often used in the domain of natural language analysis. We follow Allen [1] and take the position that events are primarily linguistic or cognitive in the sense that the world does not really contain events. An event may be viewed as a means to classify how the world may change, i.e. the different ways to update the state of the world. As a consequence, any execution of an action may be viewed as an event.
In our language, events are represented as objects. There is a predefined class \textit{Event} which describes the characteristic common to every event. In particular, this class \textit{Event} contains the attribute \textit{Date} whose value is an interval of time. Therefore, if \(e\) is an event, then \(\text{Date}(e) = (t_1, t_2)\) means that the event \(e\) occurs during the interval \((t_1, t_2)\).

To represent our case study, we need to express at least three other temporal notions in order to specify the temporal context of a sentence, namely before, during and after the occurrence of an event. This actually turns to be enough to specify our case study. This quite simple temporal structure is logically defined by introducing three boolean attributes in the \textit{Event} class – \textit{Before}, \textit{During} and \textit{After} – with the following axioms, where \(e\) is considered to be an event:

- \(\forall e, \text{Before}(e) \iff \exists t_1, t_2, \text{Date}(e) = (t_1, t_2) \land \text{Current\_Date} < t_1\)
- \(\forall e, \text{During}(e) \iff \exists t_1, t_2, \text{Date}(e) = (t_1, t_2) \land t_1 \leq \text{Current\_Date} \leq t_2\)
- \(\forall e, \text{After}(e) \iff \exists t_1, t_2, \text{Date}(e) = (t_1, t_2) \land t_2 < \text{Current\_Date}\)

\textit{Current\_Date} is supposed to be a predefined object; its value represents the actual date. Let us notice that we consider that the evaluation of any formula is related to \textit{Current\_Date}.

Similarly to entities, the set of events is associated with a hierarchical structure, the class \textit{Event} being the top of this hierarchy. For instance, we can define a sub-class of events \textit{READ} and associate with this class two attributes \textit{Actor} and \textit{Dest} to respectively represent the agent who causes the occurrence of the event \textit{READ} and the entity read during the occurrence of this event.

### 3.3. Actions

Actions are applied to objects or agents of the regulation. They are actually associated with methods in an object-oriented environment. For instance, if \textit{Read} is the name of an unary action and if \(d\) is a document, then \textit{Read}(\(d\)) represents the action of reading the document \(d\).

In order to be homogeneous with the above temporal structure, we introduce three corresponding binary predicates: if \(a\) is an agent and if \(\alpha\) is an action, then:

- \(\text{Before\_Exec}(a, \alpha)\) is true if the value of \textit{Current\_Date} is previous to the time of the execution of \(\alpha\) by \(a\) (this means, \(a\) has not yet executed \(\alpha\)).
- \(\text{During\_Exec}(a, \alpha)\) is true during the execution of \(\alpha\) by \(a\).
- \(\text{Exec}(a, \alpha)\) is true after \(a\) has executed \(\alpha\). \text{Exec} is actually an abbreviation for \textit{After\_Exec}.

Actions and events are connected through these temporal predicates. For instance, for the predicate \textit{Before\_Exec}, the action \textit{Read} and the corresponding event \textit{READ}, we have the following axiom:

- \(\forall a, \forall d, \text{Before\_Exec}(a, \text{Read}(d)) \iff \exists e, (\text{READ}(e) \land \text{Actor}(e) = a \land \text{Dest}(e) = d \land \text{Before}(e))\)

where symbol variables \(d, e, a\) respectively denote a document, an event, and an agent.

For convenience reasons and in order to increase the expressive power of our language, we have also introduced another predicate, namely \textit{Res\_Exec} to deal with the result \textit{res} of an action \(\alpha\). If \(a\) is an agent and \(\alpha\) is an action, then \(\text{Res\_Exec}(a, \alpha, \text{res})\) is to be read \(a\) has executed the action \(\alpha\) and \(\text{res}\) is the result of this execution. We have the following axiom:

- \(\forall a, \forall \alpha, \forall \text{res}, \text{Res\_Exec}(a, \alpha, \text{res}) \rightarrow \text{Exec}(a, \alpha)\)

### 3.4. Bring it about

Regulations often contain references to implicit actions: they only mention expected results of such actions, without explicitly naming them. Moreover, regulations may also contain references to indirect actions: they only mention the agent who ensures or influences the action to be performed, and not the agent who actually performs the action. This is generally the case when the regulation mentions the person who is responsible for doing something instead of the person who will actually perform the action of doing something.

To deal with such notions, we introduced a modal operator \textit{Do} in our language. Let \(a\) be an agent, and \(p\) a formula. The expression \textit{Do}(\(a, p\)) intuitively means that \(a\) brings it about that (or sees to it that) \(p\) is the case (\(p\) describes the expected effect of the implicit action in the resulting state).

In order to define an axiomatics for this modality, we follow the proposals of Kanger [12] and Pörn [16]. In their approaches, \textit{Do}(\(a, p\)) is true if the two following conditions are satisfied:

1. It is necessary for something which \(a\) has done that \(p\), i.e. \(p\) would be true in all hypothetical situations in which \(a\) has done the same actions as in the actual situation.
2. But for \(a\)’s action, it might not be the case that \(p\), i.e. if \(a\) has behaved differently from what he has done in the actual situation, then it might be the case that \(p\) would be false.
For instance, if a first agent $a$ executes the action of delivering a document $d$ to a second agent $b$, then one can conclude that $a$ brings it about that $b$ becomes the holder of this document:

- $\text{Exec}(a, \text{Deliver}(d, b)) \rightarrow \text{Do}(a, \text{Holder}(d) = b)$

We accept the following axiomatics for the $\text{Do}$ modality (previously proposed by Pörn):

1. $\text{Do}(a, p) \rightarrow p$
   
   This axiom says that the modality $\text{Do}$ is a success in the sense that bringing it about that $p$ implies $p$.

2. $\neg \text{Do}(a, \text{True})$
   
   This axiom says that nobody brings it about that something is a tautology. This is because we consider that the truth of a tautology is not caused by anybody.

3. $\text{Do}(a, p) \land \text{Do}(a, q) \rightarrow \text{Do}(a, p \land q)$
   
   This axiom says that if $a$ brings it about that $p$ and brings it about that $q$, then $a$ actually brings it about the conjunction of $p$ and $q$.

4. $\frac{\text{Do}(a, p) \land \text{Do}(a, q)}{\text{Do}(a, p \land q)}$
   
   i.e. if $p \leftrightarrow q$ is a tautology, then bringing it about that $p$ is equivalent to bringing it about that $q$.

3.5. Deontic Modalities

In order to specify a regulation, we need to express norms, i.e. rules which say what must, may or must not be done. For this purpose, we introduce classical deontic modalities as $\text{O}$, $\text{F}$, and $\text{P}$ (corresponding respectively to Obligation, Prohibition and Permission). These deontic modalities take formulae as arguments (and not, for instance, actions) because of the expressiveness needs of our application (see section 5.3 below). So, let $p$ be a formula, then:

- $\text{Op}$ is to be read $p$ is obligatory. We accept the following axiomatics for the $\text{O}$ modality:
  
  1. $\text{O}(p \land q) \leftrightarrow \text{Op} \land \text{Oq}$
     
     This axiom says that the obligation to do the conjunction of $p$ and $q$ is equivalent to the obligation to do $p$ and the obligation to do $q$.
  
  2. $\neg (\text{Op} \land \text{O}\neg p)$
     
     This axiom excludes the possibility to have a conflicting situation in which it is obligatory to do $p$ and the negation of $p$.

3. $\frac{p \rightarrow q}{\text{Op} \rightarrow \text{Oq}}$
   
   i.e. if $p \rightarrow q$ is a tautology, then the obligation to do $p$ should imply the obligation to do $q$.

- $\text{Pp}$ is to be read $p$ is forbidden. In our model, prohibition to do something is simply defined as the obligation not to do something. Therefore, we have the following definition which relates the $\text{F}$ and $\text{O}$ modalities:
  
  1. $\text{Fp} \leftrightarrow \text{O}\neg p$

- $\text{Pp}$ is to be read $p$ is permitted. We consider that $\text{P}$ is an explicit permission, so that we do not accept $\text{Pp} \leftrightarrow \neg \text{Fp}$, but only $\text{Pp} \rightarrow \neg \text{Fp}$. This is the reason why this modality is associated with a specific axiomatics distinct from the axiomatics associated with the obligation:
  
  1. $\text{P}(p \land q) \rightarrow \text{Pp} \land \text{Pq}$
     
     This axiom says that the permission to do $p$ and $q$ should imply the permission to do $p$ and the permission to do $q$. Notice that the converse is generally not true. For instance, if you have your driving licence, you are permitted to drive a car. You may also be permitted to drink a bottle of wine. However, it is prohibited to drive a car and to drink a bottle of wine, at the same time.

2. $\text{Pp} \land \text{Oq} \rightarrow \text{P}(p \land q)$
   
   This axiom says that if it is permitted to do $p$ and obligatory to do $q$, then it is permitted to do the conjunction of $p$ and $q$. In particular, this axiom implies that:

3. $\text{Op} \rightarrow \text{Pp}$
   
   i.e. the obligation to do $p$ should imply the permission to do $p$.

4. $\text{Pp} \rightarrow \neg \text{Fp}$
   
   i.e. if $p$ is explicitly permitted, then $p$ should not be prohibited.

5. $\frac{\text{Fp} \rightarrow \text{Fq}}{\text{Pp} \rightarrow \text{Pq}}$
   
   i.e. if $p \rightarrow q$ is a tautology, then the permission to do $p$ should imply the permission to do $q$.

3.6. Responsibility and delegation

In order to formalize the organizational concepts of responsibility and delegation, we must first define a special object class $\text{Mission}$ with the following structure:

**Class** Mission

**Attributes:** Responsible : Agent

Controller : Agent

Obligations : set of Action

**Methods:** Designate.Responsible : Mission $\rightarrow$ Agent
Therefore, if \( m \) is a mission and \( a \) and \( b \) are two agents, then \( \text{Responsible}(m) = a \) means that \( a \) is responsible for the mission \( m \) and \( \text{Controller}(m) = b \) means that \( b \) is the agent who is in charge of controlling the execution of \( a \)'s mission. In particular, this means that \( a \)'s responsibility is to agent \( b \), i.e. a responsibility relationship exists between \( a \) and \( b \). Finally, \( \text{Obligations}(m) \) represents the set of actions the responsible agent \( a \) has to discharge with respect to the responsibility he holds.

The method \( \text{Designate	extunderscore Responsible} \) enables a responsibility relationship to be created between two agents. We shall accept the two following axioms:

\[
\forall a, \forall \text{mission}, \forall b, \\text{Res}\text{exec}(a, \text{Designate	extunderscore Responsible}(\text{mission}), b) \\
\text{\quad } \rightarrow \text{Do}(a, \text{Responsible}(\text{mission}) = b) \\
\text{\quad } \land \text{Do}(a, \text{Controller}(\text{mission}) = a)
\]

i.e. if \( a \) designates \( b \) responsible for the mission \( m \) then \( a \) and \( b \) respectively become controller and responsible for this mission.

\[
\forall a, \forall m, \forall \alpha, \\text{Mission}(m) \\
\text{\quad } \land \text{\text{Obligations}}(m) = a \\
\text{\quad } \land \text{Responsible}(m) = a \\
\text{\quad } \rightarrow \text{O} \text{Exec}(a, \alpha)
\]

i.e. if \( a \) is responsible for the mission \( m \) and \( \alpha \) is an action associated with the obligations of \( m \), then \( a \) is obliged to execute \( \alpha \).

We shall also use the following action constructors:

- **Delegate.** If \( a \) is an agent and \( \alpha \) is an action, then \( \text{Delegate}(a, \alpha) \) corresponds to the action of delegating to the agent \( a \) the obligation of executing the action \( \alpha \). We shall accept the following axioms:

\[
\forall a, \forall b, \forall \alpha \\
\text{Agent}(a) \land \text{Agent}(b) \\
\text{\quad } \land \text{Action}(\alpha) \\
\text{\quad } \land \rightarrow \text{O} \text{Exec}(a, \alpha) \\
\text{\quad } \rightarrow \text{F} \text{Exec}(a, \text{Delegate}(b, \alpha))
\]

i.e. any agent \( a \) is forbidden to delegate an obligation he does not have.

\[
\forall a, \forall b, \forall \alpha \\
\text{Exec}(a, \text{Delegate}(b, \alpha)) \\
\text{\quad } \rightarrow \exists m, (\text{Mission}(m) \\
\text{\quad } \land \text{Responsible}(m) = b \\
\text{\quad } \land \text{Controller}(m) = a \\
\text{\quad } \land \text{Obligations}(m) = \{a\})
\]

i.e. \( a \) delegates to \( b \) the obligation to do \( \alpha \) means that a mission \( m \) is created whose responsible is \( b \), controller is \( a \) and associated obligations are to do \( \alpha \).

- **Authorize:** \( \text{Authorize}(a, \alpha) \) corresponds to the action of giving an agent \( a \) an explicit authorization to perform \( \alpha \). Note that we consider that an authorization is not equivalent to a permission: an authorization may be necessary, but not sufficient to get an explicit permission. We shall provide an example of such a situation in section 4.

### 3.7. Other concepts

Our case study raised other expressiveness needs; this is the reason why we introduced other modalities, which we shall now describe. Let \( p \) be a formula and \( a \) an agent, then:

- **Estimate\((a, p)\)** is to be read \( a \) estimates that \( p \) is true. \( \text{Estimate} \) is a doxastic modality, i.e. it behaves like a belief modality\(^1\). This means that we do not have: \( \text{Estimate}(a, p) \rightarrow p \), because we may have \( \text{Estimate}(a, p) \) being true while \( p \) is actually false in the real world. We actually accept an axiomatics of type KD45 for the modality \( \text{Estimate} \):

1. \( \text{Estimate}(a, p) \land \text{Estimate}(a, p \rightarrow q) \rightarrow \text{Estimate}(q) \)

   This axiom says that if \( a \) estimates that \( p \) is true and also that \( p \rightarrow q \) is true then \( a \) is able to conclude that \( q \) is true.

2. \( \neg(\text{Estimate}(a, p) \land \text{Estimate}(a, \neg p)) \)

   i.e. \( a \) does not estimate inconsistently.

3. \( \text{Estimate}(a, p) \rightarrow \text{Estimate}(a, \text{Estimate}(a, p)) \)

   i.e. if \( a \) estimates that \( p \) then he also estimates that he estimates that \( p \).

4. \( \neg\text{Estimate}(a, p) \rightarrow \text{Estimate}(a, \neg\text{Estimate}(a, p)) \)

   i.e. if \( a \) does not estimate that \( p \), then he estimates that he does not estimate that \( p \).

5. \( \frac{p}{\text{Estimate}(a, p)} \)

   i.e. \( a \) estimates every tautology.

- **Know\((a, p)\)** is to be read \( a \) knows \( p \). The axiomatics for the modalities \( \text{Know} \) and \( \text{Estimate} \) are similar except that we accept the following additional axiom for the modality \( \text{Know} \):

\[ \text{Know}(a, p) \rightarrow p \]

i.e if \( a \) knows \( p \) then \( p \) is true in the real world.

\(^1\)We do not intentionally use the classical \( \text{Belief} \) modality; in our mind, there is a difference between \( \text{Belief} \) and \( \text{Estimate} \), as an agent can change its belief about a given piece of information but, in the context of our case study, an agent never changes its estimation.
We also introduced several action constructors in order to fit with other expressiveness needs of our case study, namely \textit{Decide}, \textit{Ask} and \textit{Notify}. If \( \alpha \) is an action, then:

- \textit{Decide}(\( \alpha \)) corresponds to the action of deciding to perform \( \alpha \). We shall accept the following axiom for \textit{Decide}:
  \[
  \text{Exec}(\alpha, \text{Decide}(\alpha)) \rightarrow \text{Do}(\alpha, \text{Estimate}(\text{Before}_\text{Exec}(\alpha, \alpha)))
  \]
  i.e. \( \alpha \) decides to execute \( \alpha \) means that \( \alpha \) brings it about that he estimates that the current date is before the execution of \( \alpha \) by \( \alpha \).

- \textit{Ask}(\( \alpha, \alpha \)) corresponds to the action of asking an agent \( \alpha \) to perform the action \( \alpha \). We shall accept the following axiom for \textit{Ask}:
  \[
  \text{Exec}(\alpha, \text{Ask}(b, \alpha)) \rightarrow \text{Do}(b, \text{Know}(\alpha, \text{Before}_\text{Exec}(b, \alpha)))
  \]
  i.e. \( \alpha \) asks \( b \) to perform \( \alpha \) means that \( \alpha \) brings it about that \( b \) knows that \( \alpha \) estimates that \( \alpha \) should execute \( \alpha \).

- \textit{Notify}(\( \alpha, \alpha \)) corresponds to the action of notifying \( \alpha \) of the execution of the action \( \alpha \). We assume that an agent \( b \) who executes the action \( \text{Notify}(\alpha, \alpha) \) is trusted in the sense that \( b \) will not notify \( \alpha \) without having executed the action \( \alpha \). Thus, we shall accept the following axiom for \textit{Notify}:
  \[
  \text{Exec}(b, \text{Notify}(\alpha, \alpha)) \rightarrow \text{Do}(b, \text{Know}(\alpha, \text{Exec}(b, \alpha)))
  \]
  i.e. \( b \) notifies \( \alpha \) of the execution of \( \alpha \) means that \( b \) brings it about that \( \alpha \) knows that \( b \) has executed \( \alpha \).

4. Application to a case study

In our case study, many paragraphs may be viewed as conditional rules with normative conclusions. In other words, they can be translated by: if this condition is satisfied, then this agent is obliged, permitted or prohibited to do something.

We shall now provide several examples to show how rules in natural language are translated in our formalism.

4.1. Rules with normative conclusions

As said above, the rules which specify a regulation generally have normative conclusions. We give some examples:

- The transmitter of a classified document is \textit{obliged} to update the document classification as soon as it is possible, i.e. immediately after the transmitter evaluates that the document classification is obsolete.

\textbf{Rule R1}:
\[
\begin{align*}
\text{Classified Document}(d) \\
\land & \; \text{Transmitter}(d) = t \\
\land & \; \text{Res}_{\text{Exec}}(t, \text{Evaluate}_{\text{Classification}}(d), \text{level}) \\
\land & \; \text{Classification}(d) \neq \text{level} \\
\rightarrow & \; \text{O}_{\text{Exec}}(t, \text{Change}_{\text{Classification}}(d))
\end{align*}
\]

- Any agent who is not the transmitter of a classified document is \textit{prohibited} to change the classification of this document.

\textbf{Rule R2}:
\[
\begin{align*}
\text{Classified Document}(d) \\
\land & \; \text{Agent}(a) \\
\land & \; \text{Transmitter}(d) \neq a \\
\rightarrow & \; \text{F}_{\text{Exec}}(a, \text{Change}_{\text{Classification}}(d))
\end{align*}
\]

- The holder of a classified document is \textit{permitted} to ask the transmitter to revise the classification of this document.

\textbf{Rule R3}:
\[
\begin{align*}
\text{Classified Document}(d) \\
\land & \; \text{Transmitter}(d) = t \\
\land & \; \text{Holder}(d) = h \\
\rightarrow & \; \text{P}_{\text{Exec}}(h, \text{Ask}(t, \text{Change}_{\text{Classification}}(d)))
\end{align*}
\]

4.2. Rules with organizational concepts

- Every organization which holds some secret documents is \textit{obliged} to designate an agent who is \textit{responsible} for preserving these documents.

\textbf{Rule R4}:
\[
\begin{align*}
\text{Organization}(o) \\
\land & \; \text{Classified Document}(d) \\
\land & \; \text{Classification}(d) = \text{Secret} \\
\land & \; \text{Holder}(d) \in \text{Employees}(o) \\
\rightarrow & \; \text{O}_{\text{Exec}}(\text{Head}(o), \text{Designate}_{\text{Responsible}}(\text{Document}_{\text{Preservation}}(o)))
\end{align*}
\]

\( \text{Document}_{\text{Preservation}}(o) \) corresponds to the mission of preserving the classified documents held by the organization \( o \). Of course, there are other axioms whose purposes are to define the obligations associated with this mission. Due to space limitation, these axioms are not presented in this paper.

---

\(^2\)Notice that we guess it is possible to improve this first translation. As a matter of fact, a more precise interpretation would be: \textit{when the regulation is enforced}, then if this condition is satisfied, then this agent is obliged, permitted or prohibited to do something. However, the language becomes more complicated and the rules more difficult to read. Therefore, for the sake of simplicity, we prefer to use the above interpretation.
• If it is decided to send a classified document, then the sending office is obliged to establish a consignment note of this document. The head of this office is obliged to sign this note. The head of this office is permitted to delegate the obligation to sign the note to one of his representatives.

**Rule R5**:  
Organization(o)  
\&  
Sending_Office(o) = s  
\&  
Classified_Document(d)  
\&  
Exec(o, Decide(Send(d)))  
→  
O Exec(s, Establish_Consignment_Note(d))

**Rule R6**:  
Organization(o)  
\&  
Sending_Office(o) = s  
\&  
Classified_Document(d)  
\&  
Exec(o, Decide(Send(d)))  
\&  
Consignment_Note(d) = n  
→  
O Exec(Head(s), Sign(n))

**Rule R7**:  
Organization(o)  
\&  
Sending_Office(o) = s  
\&  
Classified_Document(d)  
\&  
Exec(o, Decide(Send(d)))  
\&  
Consignment_Note(d) = n  
\&  
r ∈ Representatives(Head(o))  
→  
P Exec(Head(s), Delegate(r, Sign(n)))

If the head of s uses this permission, then one of his representatives r would become responsible for a mission associated with one obligation: the obligation to sign the note n; the head of s becomes controller of this mission.

• Anybody who wants to visit a protected area is obliged to get an authorization from the head of this area. Moreover, the visitor is obliged to be supervised by an agent who is especially designated for this task.

**Rule R8**:  
Protected_Area(a)  
\&  
Head(a) = h  
\&  
Person(p)  
\&  
p ∉ Employees(a)  
\&  
¬ Exec(h, Authorize(p, Visit(a)))  
→  
F Exec(p, Visit(a))

It is interesting to note the negative turn we used to formalize the first sentence: somebody, who did not get from the head of a protected area the authorization to visit this protected area, is prohibited to do so.

**Rule R9**:  
Protected_Area(a)  
\&  
Head(a) = h  
\&  
Person(p)  
\&  
p ∉ Employees(a)  
\&  
Exec(h, Authorize(p, Visit(a)))  
\&  
Exec(h, Designate_Responsible(Supervision(p)))  
→  
P Exec(p, Visit(a))

This is an example of a rule in which an authorization is not sufficient to get an explicit permission. To get this explicit permission, the visitor p must be supervised. In this rule, Supervision(p) is a mission whose associated obligations is to supervise p.

### 4.3. Rules with events

We shall use the following convention: events are denoted with upper case letters, and actions with small letters.

• Every time a document classified at the secret level is destroyed (action), the holder of this document and a witness of the destruction (corresponding event) are obliged to sign the destruction minutes.

**Rule R10**:  
Classified_Document(d)  
\&  
Classification(d) = Secret  
\&  
Holder(d) = h  
\&  
DESTROY_DOCUMENT(e)  
\&  
Actor(e) = h  
\&  
Destroyed_Object(e) = d  
\&  
Witness(e) = w  
\&  
After(e)  
\&  
DestructionMinutes(d) = m  
→  
O Exec(h, Sign(m)) ∧ O Exec(w, Sign(m))

In this rule, e is an event belonging to the class DESTROY DOCUMENT. This event class has the following attributes: Actor to represent the agent who performs the document destruction, Destroyed_Object to represent the document to be destroyed and Witness to represent a witness of this destruction.

• When a meeting is finished (event: meeting), its organizer is obliged to incinerate all the preparatory documents of this meeting.

**Rule R11**:  
MEETING(m)  
\&  
Organizer(m) = o
After(m)
\[ d \in \text{Preparatory\_Documents}(m) \]
\[ \rightarrow O \text{Exec}(o, \text{Incinerate}(d)) \]

- If the organizer of a meeting does not incinerate all the preparatory documents of the meeting when it is finished, then he is obliged to bring it about that these documents are kept in a safe.

**Rule R12 :**
MEETING(m)
\[ \wedge \text{Organizer}(m) = o \]
\[ \wedge \text{After}(m) \]
\[ \wedge d \in \text{Preparatory\_Documents}(m) \]
\[ \wedge \neg \text{Exec}(o, \text{Incinerate}(d)) \]
\[ \rightarrow O \text{Do}(o, \exists s, (\text{Safe}(s) \land d \in \text{Content}(s))) \]

### 4.4. Temporal rules

- **Before** every meeting, the organizer of this meeting is obliged to establish a list of all participants in this meeting.

**Rule R13 :**
MEETING(m)
\[ \wedge \text{Organizer}(m) = o \]
\[ \wedge \text{Before}(m) \]
\[ \rightarrow O \text{Exec}(o, \text{Establish\_Participants\_List}(m)) \]

- **During** an agent is elaborating a document classified at the secret level, he is obliged to work in a protected area.

**Rule R14 :**
Agent(a)
\[ \wedge \text{Classified\_Document}(d) \]
\[ \wedge \text{Classification}(d) = \text{Secret} \]
\[ \wedge \text{During\_Exec}(a, \text{Elaborate}(d)) \]
\[ \rightarrow O \exists z, ((\text{Protected\_Area}(z) \land \text{During\_Exec}(a, \text{Work}(z))) \]

- **After** a secret document is destroyed, the document holder is obliged to notify the document author of this destruction.

**Rule R15 :**
Classified\_Document(d)
\[ \wedge \text{Classification}(d) = \text{Secret} \]
\[ \wedge \text{Holder}(d) = h \]
\[ \wedge \text{Exec}(h, \text{Destroy}(d)) \]
\[ \wedge \text{Transmitter}(d) = t \]
\[ \rightarrow O \text{Exec}(h, \text{Notify}(t, \text{Destroy}(d))) \]

5. Using our formalism

Our formalism may be used to develop many applications to help the persons who are in charge to define a security policy. In this section, we more precisely present two of these applications. The first application provides functionalities to query a given security policy in order to know which norms apply to a given situation. The second one enables the policy consistency to be checked. In both cases, we need to define means to reason about the regulation. This was the purpose of the axiomatics we developed in section 3.

### 5.1. Querying a regulation

Based on this axiomatics, we can develop a tool which enables a user to query a security policy. For instance, let us assume that a user wants to ask the following query:

- **Who is permitted to change the classification of a classified document?**

This query is translated in the following logical formula:

- \[ \exists d, \text{Classified\_Document}(d) \]
- \[ \wedge \text{PExec}(a, \text{Change\_Classification}(d)) \]

Notice that this user generally does not expect to obtain a list of persons who are actually permitted to change the document classification but instead a formula which corresponds to a sufficient condition to satisfy the query. This technique of query answering is called *intentional answer* in [5]. When applied to the above query, this technique would lead to the following answer:

- **True if**

\[ Transmitter(d) = a \]
\[ \wedge \text{Res\_Exec}(a, \text{Evaluate\_Classification}(d), \text{level}) \]
\[ \wedge \text{Classification}(d) \neq \text{level} \]

i.e. the transmitter of a document is permitted to change the classification of a document if he evaluates that this classification is obsolete.

### 5.2. Checking the regulation consistency

Similarly, it is also useful to develop a tool which enables a user to check the security policy consistency. In the case of a regulation, we may actually define three different cases of inconsistency:

1. **Pp \land \neg Fp**
   i.e. there is a formula \( p \) which is both permitted and prohibited.
2. \( Op \land Fp \)
   i.e. there is a formula \( p \) which is both obligatory and prohibited.

3. \( Op \land O \neg p \)
   i.e. there is a formula \( p \) which is obligatory and whose negation is also obligatory.

However, due to the axiomatics we propose for the deontic modalities, it is easy to verify that the first case is weaker in the sense that it is implied by the two last cases. So our objective in developing a tool to check the policy consistency is to find condition \( c \) and situation \( s \) such that:

\[ c \rightarrow Ps \land Fs \]

In this case, it is also useful to apply the intentional answering technique to provide the user with formulae which characterize the condition \( c \) and the situation \( s \) which lead to an inconsistency.

### 5.3. Problematical rules

Not surprisingly, there are several examples of rules with complex structures in our case study. Some of them are currently preventing our objective of a complete automatic treatment of our case study in a PROLOG-based system. We give some examples:

- Rules with disjunctions in the range of a deontic modality. For instance:
  
  *As soon as a classified document becomes useless, its transmitter is obliged to incinerate it, or tear it into shreds.*

  **Rule R16**:  
  \[
  \begin{align*}
  \text{Classified Document}(d) \\
  \land \text{Transmitter}(d) = t \\
  \land \text{Useless Document}(d) \\
  \rightarrow O(\text{Exec}(t, \text{Incinerate}(d)) \\
  \lor \text{exec}(t, \text{Shred}(d)))
  \end{align*}
  
- Rules with existential variables in the range of a deontic modality. Rules R12 and R14 already provide examples of this kind of rules. We can give another example:
  
  *If nobody is currently using a given classified document, then the agent who is responsible for preserving this document is obliged to bring it about that this document is kept in a safe.*

  **Rule R17**:  
  \[
  \begin{align*}
  \text{Classified Document}(d) \\
  \land \text{Holder}(d) \in \text{Employees}(o) \\
  \land \forall x, \text{Agent}(x) \rightarrow O(\text{During Exec}(x, \text{Use}(d))) \\
  \rightarrow O(\text{Responsible Preservation}(o), \\
  \exists s, (\text{Safe}(s) \land d \in \text{Content}(s)))
  \end{align*}
  
  Actually, this rule was initially specified in the document as follows: *If nobody is currently using a given classified document, then it is obligatory that this document be kept in a safe.* However, we decided to complete this rule by specifying the agent who is in charge of this obligation. The help of an expert in regulation analysis was very useful to perform this kind of completion.

- Rules with nested deontic modalities. For instance:
  
  *Once a year, the head of every organization is obliged to bring it about that all employees are obliged to perform the inventory of every secret document they hold.*

  **Rule R18**:  
  \[
  \begin{align*}
  \text{Organization}(o) \\
  \land x \in \text{Employees}(o) \\
  \land \text{Classified Document}(d) \\
  \land \text{Classification}(d) = \text{Secret} \\
  \land \text{Holder}(d) = x \\
  \land \text{Current Date} = \text{Annual Inventory Date}(o) \\
  \rightarrow O(\text{Head}(o), O(\text{Exec}(x, \text{Inventory}(d))))
  \end{align*}
  
  Notice that we have translated the condition “once a year” by:

  “\( \text{Current Date} = \text{Annual Inventory Date}(o) \)”

  This implicitly means that every head of an organization is obliged to bring it about that there exists a date for the annual inventory:

  **Rule R19**:  
  \[
  \begin{align*}
  \text{Organization}(o) \\
  \rightarrow O(\text{Head}(o), \exists \text{date}, \\
  \text{Annual Inventory Date}(o) = \text{date})
  \end{align*}
  
### 5.4. Other functionalities

There are several other functionalities that we plan to investigate in the future. We only mention two of them:

- Functionality for checking if a given situation does not violate the security policy. In this case, we have to check the formulae \( p \land Fp \) and \( \neg p \land Op \). It would be also interesting to derive which sanction applies when a policy violation occurs.
6. Comparison with related work

Deontic logic, because of its ability to specify the concepts of obligation, permission and prohibition, is an attractive candidate for expressing security policies. It was first used by Glasgow and McEwen to specify confidentiality policies [10]. In this case, the language combines a deontic logic with an epistemic logic. In [2, 3], we used this approach to propose a new security property called causality as a candidate for expressing security policies. It was first used by Glasgow and McEwen to specify conﬁdentiality and in [7], we show how the properties of non-interference [11] and non-deducibility [19] actually correspond to different definitions of the concepts of permission and prohibition to know. Finally, in [9], we show how the so-called inference problem, which occurs in multilevel databases when a user cleared at a low level can infer higher-classified information, may be viewed as a problem of security policy consistency.

In this paper, we propose a more general language which combines a deontic logic with a logic of action. This enables to express more general security requirements and we show how it may be used to specify a Corporate Security Policy. As observed by Wieringa et al. [22], notice that there are two ways of interpreting the sentence “it is forbidden to change the classification of a document”. It may be the observation that a rule exists or the promulgation of the rule itself. The first one is a proposition with a truth value, while the other is a norm with the effect of a command. The deontic logic we develop in this paper is related to the first approach.

We also guess there is a clear connection between our work and the approach proposed by Strens and Dobson in [18]. They show how responsibility modelling can be used as a means to specify policy requirements, how obligations define what a responsibility holder must do and how the process of delegation may be used to transfer obligations and create new responsibility relationships. Moffett and Sloman also proposed a series of paper which investigate several concepts discussed in our paper: responsibility [14], delegation of authority [13] and policy consistency [15]. They do not propose a formal definition for these concepts but it is their intuition that deontic logic is the correct approach to follow if one wishes to set up a theory of policies on a sound basis. We guess the language and the associated axiomatics we proposed in this paper is a step in this direction.

7. Discussion

Our case study is not a toy example. So, an intrinsic difficulty is related to the quite large size of the document and to the natural complexity and ambiguity of a text in natural language. Consequently, the first difficulty we had to face when we started to formalize our case study, was to clarify all subtleties contained in a document written in an administrative natural language.

Faced with this difficulty, the first step of our work was to get a correct and as precise as possible interpretation of the text of this case study, by using classical deontic concepts and a smaller subset of natural language (in comparison with the language initially used to write this case study). We discarded every piece of information that was neither a definition nor a normative rule, but only a comment or advice. Another domain-dependent task was to identify which agents are concerned with each norm of the instruction: this information was often left implicit in the regulation.

We are not specialists in this administrative domain, so this first part of our work was very time consuming. Fortunately, we enjoyed very much to be successfully helped, in clarifying the initial document, by an expert in regulation analysis. Let us also note that we performed this first task completely by hand and we do not see any simple means to automatize it. Another step was a modeling task, consisting in extracting regulation concepts. The large size and the natural complexity of the document we had to model made this step quite fastidious. Knowledge modeling assistance tools such as KADS-tool [21], KOD [20] or other similar tools would probably provide helpful support: maybe we would have wasted less time, especially in getting a homogeneous set of concepts. Nevertheless, this modeling work may take advantage of being assisted by an expert, in order to exactly reflect the semantics of the domain. In our case study, an additional difficulty comes from an intentional ambiguity of the text: the lack of precision of many paragraphs was deliberate in order to make the regulation applicable by a large variety of organizations.

Another difficulty comes from paragraphs containing meta-information. A regulation can be viewed as a set of definitions and normative rules. Frequently these definitions or rules mention the overall regulation or parts of it. In our case study, we did not deal with such sentences. For instance: *Any holder of a classified information is obliged to respect the regulation protection rules associated with the information classification.*

On the other hand, our case study raised up several interesting and real-world problems which would require further theoretical work. In particular, we are not fully satisfied by...
our proposals for several concepts, especially the concepts of responsibility and delegation; these concepts are clearly worth deeper investigation.

In order to improve inference performances, we also need to make our model more exhaustive. All semantic links between concepts have to be fully explained. For instance, in our case study, we had to express semantic links between the actions to disclose and to know (if information is disclosed, then somebody will know it). Building such an exhaustive list of axioms is a difficult but necessary task if we want to be sure to infer all expected conclusions. Some theoretical difficulties may also come from the intrinsic content of the regulation we had to model. For instance, it is an additional difficulty to draw inferences with logical formulae when there is a disjunctive norm in the conclusion; however you cannot avoid this without betraying the initial sense of the regulation.

Finally, let us notice that, in our logic-based approach, we did not choose to distinguish different granularity levels of a regulation. According to user needs, this may be interesting or not. A logic-based approach is perhaps not the best way for easily structuring information. But, this was not the matter in this paper.

8. Acknowledgements

This work was carried out with the support of the CELAR (Contract No. 0164194). We are especially indebted to Jacques Abdelaziz for the interest he takes in our work and for his help in clarifying the case study. Without his support, this work would not be possible. Also, comments by Nora Cuppens-Boulahia on earlier drafts of this paper have resulted in a number of improvements. We finally want to thank Laurence Cholvy and Robert Demolombe for their helpful suggestions.

References