Delegation of Obligations and Responsibility

Meriam Ben Ghorbel, Frédéric Cuppens, Nora Cuppens-Boulahia, Daniel Le Métayer, Guillaume Piolle

Delegation of Obligations and Responsibility

Meriam Ben Ghorbel-Talbi[1], Frédéric Cuppens[1], Nora Cuppens-Boulahia[1], Daniel Le Métayer[2], and Guillaume Piolle[3]

[1] Institut TELECOM/Télécom Bretagne
2, rue de la Châtaigneraie, 35576 Cesson-Sévigné Cedex - France
{meriam.benghorbel,frederic.cuppens,nora.cuppens}@telecom-bretagne.eu
[2] INRIA Rhône-Alpes
Inovallée, 655 avenue de l'Europe, 38334 Saint-Ismier Cedex - France
daniel.le-metayer@inria.fr
[3] Supélec
Avenue de la Boulaie, CS 47601, 35576 Cesson-Sévigné Cedex - France
guillaume.piolle@supelec.fr

Abstract. In this paper, we discuss the issue of responsibilities related to the fulfillment and the violation of obligations. We propose to formally define the different aspects of responsibility, namely causal responsibility, functional responsibility, liability as well as sanctions, and to examine how delegation influences these concepts. Our main aim is to identify the responsibility of each agent that is involved in the delegation of obligations. More precisely, we try to answer to the following questions: who is responsible for the obligation fulfillment? When a violation occurs, which agents are causally responsible for this violation? Who is liable for this violation and to whom? And finally, who must be sanctioned?

Keywords: Responsibility, Obligations, Delegation

1 Introduction

Obligations are important means to specify security control, in particular usage control [14,15,3]. Obligations must usually be fulfilled by a fixed deadline, otherwise violations occur and punitive sanctions are inflicted upon agents (for instance through the activation of prohibitions or new obligations). Yet, agents can violate their obligations due to various causes that can be related to agents themselves (e.g. lack of time or competence), or to other agents who have performed (or not) actions such that they have blocked out the fulfillment of the obligation, or finally to system faults, such as a system dysfunctioning or insufficient authorization/resource [10]. For these reasons, it is necessary to have means to clearly identify the responsibility of agents that are involved in the obligation violation, especially when obligations are delegated to one or more other agents.

Indeed, identifying the responsibility of agents in the case of violations is a fundamental part of security and is central to the determination of liability and sanctions. For this purpose, we focus here on these two issues and we propose,

in section 2, a formal model that defines different levels of responsibilities [4], namely functional responsibility which is the operational aspect of an obligation, causal responsibility which expresses the link of causality between an agent's actions and a given fact, and liability which is related to the notion of blame, sanction or damage reparation. In section 3, we propose a model of the concept of obligation delegation. We examine how to deal with the different kinds of responsibilities, and we give a concrete example to illustrate our approach. We give, in section 4, a discussion on related work and concluding remarks.

2 Logical model of obligation and responsibility

In the following, propositions will be noted by lower case italic letters (a, b, p ...), variables by roman strings starting with a capital letter (Var) and litterals by strings in fixed width font (`litt`). Indifferent variables are noted with an underscore (_), using Prolog-like notation.

Basic structure of the model Our framework is based on the notion of organization. Organizations will be noted $a, b, c \ldots \in \mathbb{O}$. They do not have any kind of property, but we will introduce a way to define arbitrary binary relations between them. This allows to nest organizations, to define roles and other high-level concepts. We choose to represent agents and organizations at the same level, by considering that an agent is itself an organization. Another core component is the notion of obligation. We consider that obligations always come from a normative source (noted $x \in \mathbb{X}$), which is an object shared by a set of organizations. It can be a contract, an order, a law or any kind of normative document. For instance, an organization can publish internal regulations, or several organizations can agree on a contract. Normative sources will be used as references for obligations and associated concepts. The logic does not make any distinction between obligations to be and obligations to do, nor between actions, events and states. These distinctions are abstracted away by the notion of fact (noted $p \in \mathbb{P}$). A fact is a proposition describing a situation or an action. It can be an observation of the system or the object of an obligation. In the remaining of section 2, we will consider a simple obligation (*i.e.* without any delegation), between two agents (or organizations) a and b, coming from a normative source x. We will present the various constructs related to obligations and responsibility, before discussing the impact of obligation delegation in section 3.

Obligations and organizations Obligation is represented by the modality class O, differentiated in a four-parameter predicate. $O(a, p, b, x)$ represents the fact that a has the obligation, towards b, to ensure p, and that this obligation comes from the normative source x. As our goal is to model the various kind of underlying responsibilities in a fine-grained way, the obligation modality has been emptied of most of its usual meaning, and is best described as the representation of a speech act, the acknowledgment

that an obligation has been expressed. Formally, each tuple (O, a, b, x) is a monadic obligation modality, applied to facts. They are defined like in SDL, with a KD axiomatics [21]. The abstract relation structure is brought by a *relation* predicate. *relation*(`relationName`$, a, b$) means that a is in relation `relationName` with b. Binary relations on $\mathbb{O} \times \mathbb{O}$ can be introduced this way. For instance, *relation*(`playsRole`$, a, r, b$) can mean that a plays a given role r in the organization b.

Functional responsibility is the operational aspect of an obligation, the fact that the obligated agent is actually expected to perform a task itself. We note $FR(a, p, b, x)$ the fact that a has the functional responsibility, for which it is accountable to b, to ensure p, and that this responsibility comes from normative source x. In simple cases, functional responsibility is directly derived from the expressed obligation: if an agent is obliged to ensure p then it has the corresponding functional responsibility. This is why, in this simple (delegation-free) version of the framework, functional responsibility is formally equivalent to obligation (eq. 1). The predicate is introduced to make a distinction between the responsibility and the mere speech act.

$$FR(a, p, b, x) \stackrel{def}{=} O(a, p, b, x) \tag{1}$$

Causal responsibility is not necessarily derived from an obligation, but will contribute to the definition of more complex notions. It expresses the link of causality between an agent's actions and a fact, without any assumption of any kind of "fault". We note $CR_a\ p$ the fact that agent a is causally responsible for the fact p. It implies p itself. It means that a has contributed, in some way to the fact that p is true: there is a causality link between a's behaviour and p. It does not mean that a is the sole responsible agent for p.

We choose to distinguish "material causal responsibility" ($MCR_a\ p$) from "causal responsibility by direct influence" ($CRDI_a\ p$). The former means that p occured, and that there is a causality link between a's actions or inaction and the fact p. The latter means that a made another agent or organization b do something (by the means of an obligation) which made b causally responsible for the fact p. In other words, a used its influence to cause p. Material causal responsibility can be more precisely specified in many ways, by introducing complex relations between the actions and their results. In the version of the formalism presented here however, the notion remains abstract and the individual actions are hidden, because our only need here is to decide wether the causal link exists or not. In the context of this presentation, MCR_a will therefore be considered a primary operator. Yet, we should keep in mind that it is possible to distinguish between various grades of material causal responsibility, which might lead to various grades of other kinds of responsibility. Causal responsibility by direct influence, on the other hand, is defined on the basis of an obligation and of the material causal responsibility of another agent or organization (2). To conclude, this first version of causal responsibility is simply the disjunction of material causal responsibility and causal responsibility by direct influence (3).

$$CRDI_a\ p \stackrel{def}{=} O(b, p, a, x) \wedge MCR_b\ p \tag{2}$$

$$CR_a\ p \stackrel{def}{=} MCR_a\ p \vee CRDI_a\ p \tag{3}$$

Liability We understand liability with respect to an undesirable fact as the possibility, for an agent or an organization, to be blamed for the fact, to be imposed a sanction. This notion is inspired from the legal concept of liability as it appears in the French legal context, for instance, where a person is held liable if its (faulty) behaviour is causally related to a damage (to another agent or to society). In our model, the damage is represented by a fact p, the fault by a violated interdiction on p and the causal relation by our dedicated operator. It means that if an agent has not violated any norm, then it cannot be blamed or sanctioned. Therefore it may be considered that the system contains very general norms, such as the obligation not to cause a harm or loss to another agent. In our language, $L(a, p, b, x)$ means that a is liable for p towards b, because of obligations coming from normative source x. In a first version its direct form ($DL'(a, p, b, x)$), it is defined as the conjunction between a causal responsibility and a violated obligation (4).

$$DL'(a, p, b, x) \stackrel{def}{=} CR_a\ p \wedge O(a, \neg p, b, x) \tag{4}$$

This direct liability is personal in essence, but in some cases one may be liable for somebody else's actions. For instance, parents often bear civil liability in the name of their children. We need to take this kind of relationship into account, because it can also occur in many organizations, where employers, under certain circumstances, may be liable instead of their employees. In order to model this, we will use a relation `accountableFor`, which we need to be built-in. In short, if a is accountable for b, then we consider that a is liable when b should be. This allows us to define indirect liability $IL(a, p, b, x)$ as in (5). Overall liability (6) is therefore the disjunction between direct and indirect liability, where direct liability DL is redefined as the conjunction between DL' and the absence of a relation `accountableFor`.

$$IL(a, p, b, x) \stackrel{def}{=} CR_c\ p \wedge O(c, \neg p, b, x) \wedge relation(\texttt{accountableFor}, a, c) \tag{5}$$

$$L(a, p, b, x) \stackrel{def}{=} DL(a, p, b, x) \vee IL(a, p, b, x) \tag{6}$$

If different levels of causal responsibility are defined, then different levels of liability will arise. For instance, one can imagine a weaker causal responsibility CR^1 (denoting a partial responsibility) and a stronger one CR^2 (denoting a full, exclusive responsibility). CR^1 and CR^2 could give rise to two levels of liability L^1 and L^2. In some context, a L^1 liability could be considered too weak to give rise to a sanction, while an agent with L^2 liability would be considered "blamable". For simplicity, we will work only with one kind of causal responsibility here. However, several existing propositions could be useful in designing a gradation of causal responsibility, like constructions based on Pörn's D and D' modalities

[16], and in particular the recent proposal by Marek Sergot [19].

Sanction As mentioned above, in the case of obligation violation an agent or an organization has to make good for this violation. We use the predicate $sanction(s, c, p, x)$ to say that sanction s is associated to fact p by normative source x and may be imposed by agent c. Note that we use this predicate to define sanctions in the sense of punishment (penal responsibility), but also to define blame and the reparation of damage or loss (civil responsibility). We choose not to formally differentiate the two notions. In our language, sanctions are associated to the agent liable for the violation according to the normative source. $S(a, p, b, s, c)$ means that sanction s can be imposed by agent c to a following fact p, for which a is liable towards b:

$$S(a, p, b, s, c) \stackrel{def}{=} L(a, p, b, x) \wedge sanction(s, c, p, x) \tag{7}$$

Some discussion remarks One question that remains to be answered, for the sanction to be just: is the agent actually able to fulfill the obligation or to avoid its violation? For this purpose, one has to define the concept of the agents' *ability* [5,12] to fulfill a given obligation, as well as the parameters influencing this ability. Thus, when a violation occurs we can tell whether a liable agent was actually able to fulfill the obligation in that moment. Many concepts have to be defined and considered to define agents' ability. For instance, is the agent considered able to do some task if it is able to delegate it to another agent?

Another issue is the possibility of sanctions for agents which are causally responsible for the violation, but which bear no liability with respect to the current source of norms. For instance, organization b may deem agent a liable for a given violation with respect to a source of norms x, but agent c, belonging to a foreign organization on which b has no influence, may have a greater causal responsibility because it prevented a from doing its job properly. No liability of c towards b can apparently be derived, because c is not concerned by x and therefore it has not violated any norm of x. Yet, it would seem just that c could be blamed. No liability can be built upon x, but there may be other applicable normative sources. On the first hand, if a and c share a source forbidding an agent to harm another in the way c did, then a liability can be derived from that, and the corresponding sanction will be considered independently from x. It can also be the case that c broke one of its own norms and is sanctionned for that [10], but that its liability is not towards b. On the other hand, if c has not violated any norm applying to it, then it is not faulty in any way and has neither to be sanctionned nor to provide a reparation. In other words, an agent with no functional responsibility for a given fact cannot be judged liable and therefore cannot be blamed. It matches real world situations, in which a fault must be exhibited for a sanction to be applied. For instance, two shops operating in the same street may have a negative impact on each other's income, thus generating a damage, but as long as none of them breaks the general rules of commerce, no civil reparation or penal sanction can be sought.

3 Modelling obligation delegation

Now that the notions of obligation, causal responsibility, functional responsibility and liability are available, we will propose a model of the concept of obligation delegation and examine its influence on the former notions. We say that an obligated agent b delegates its obligation to another agent a when b obliges a to what b was initially obliged. Depending on the options of this delegation, this may or may not influence the functional responsibility and the liability of both a and b with respect to the obligated fact.

The delegation predicate The delegation of an obligation is represented by an instance of the *delOb* predicate. $delOb(a, p, b, c, x, \text{FRoption}, \text{Loption})$ means that b delegates to a the obligation on p that it had towards c, coming from the normative source x. The last two parameters are the options of the delegation related to functional responsibility and liability. Functional responsibility can be either shared (FRoption = `fr_share`) or forwarded (`fr_forward`). In the first case, both a and b have functional responsibility: they are both in charge of ensuring p. This is for instance the case if the obligation delegation is a request for help on a complex task. In the second case, a alone gets functional responsibility. b does not have to take actions anymore, it is a's role to actually ensure p. It is not possible that b keeps functional responsibility for itself alone, as the key idea about delegation is giving someone else something to do.

Liability can be kept (Loption = `l_keep`), shared (`l_share`) or forwarded (`l_forward`). If liability is kept, then the delegatee will accept no other liability than towards the delegator. It means that if b is liable towards c and delegates to a with `l_keep`, then a will not be liable to c, only b will. On the other hand, a will still be locally liable to b: it is a way to acknowledge that the speech act of delegation itself generates its own liability. If liability is shared, then both a and b will be liable to c (and a will still be "locally" liable to b). If liability is forwarded, then only b will be liable to c (and to a, locally).

For instance, in a conference program committee a reviewer a can delegate the obligation to review a given paper to an external reviewer b, using options `fr_forward` and `l_keep`. In this case, b has the functional responsibility to review the paper, a is liable to the PC chair if the review deadline is not met, and b is liable to a. We can also imagine the opposite situation: a PhD student delegates the obligation to review a paper to his/her advisor (obviously with his/her consent) using options `fr_share` and `l_forward`. In this case, the student transfers the obligation, i.e. he/she is no more liable to the PC chair, but will help his/her professor to review the paper. Note that there is a hierarchical authority between the professor and the student, therefore the student must request the consent of the professor before delegating the obligation (see [2] for more details about consent negotiation).

Formally, to be valid a delegation from a delegator b to a delegatee a on p necessitates the existence of a prior obligation $O(b, p, c, x)$ (i.e. an obligation to b towards another agent c), and it creates a new obligation for the delegatee a towards b. Note that this obligation is also coming from the same normative

source x. Equation (8) illustrates this derivation mechanism. The prior formula says that B delegates to A its obligation on P, coming from source X, with options LRoption and Loption. The derived formula is the new obligation of A, towards B, to ensure P according to normative source X.

$$\frac{\mathit{delOb}(\mathrm{A}, \mathrm{P}, \mathrm{B}, \_, \mathrm{X}, \mathrm{FRoption}, \mathrm{Loption})}{O(\mathrm{A}, \mathrm{P}, \mathrm{B}, \mathrm{X})} \tag{8}$$

Rights system Depending on the context of an obligation, it is not always possible or desirable to delegate it. The initial obligator may demand that the initial obligee keeps either liability or full functional responsibility, for instance. It is therefore necessary to install a rights system over obligation delegation: each normative source will also enacts a number of rights formulae, and depending on the active rules, a specific delegation will be authorized or not. More details about how to set up such contextual rights about delegation are given in [1]. It is currently assumed that normative sources properly define these rights, in that rights enacted by a given source should not interfer with the obligations coming from another one (see [8] for more details about conflict management).

Rights enacted by a normative sources are represented by *allow* and *deny* predicates. $\mathit{allow}(a, b, p, c, x, \mathrm{FRoption}, \mathrm{Loption}, \mathrm{Recursivity})$ means that an obligation $O(a, p, c, x)$ can be delegated to b with the options FRoption and Loption. If Recursivity = `recursive`, then this permission propagates to any delegated obligation. It does not if Recursivity = `nonrecursive` $\mathit{deny}(a, b, p, c, x, \mathrm{FRoption}, \mathrm{Loption}, \mathrm{Recursive})$ means that this same initial obligation cannot be delegated with these options. If *deny* is recursive, it means that any delegated obligation is also subject to it. It can be relevant, for instance, if another set of options is authorized for the delegation. Recursivity in the rights system is defined by (9).

$$\begin{aligned} &\mathit{deny}(a, b, p, c, x, \mathrm{FRoption}, \mathrm{Loption}, \texttt{recursive}) \\ &\quad \rightarrow \mathit{deny}(b, \_, p, a, x, \mathrm{FRoption}, \mathrm{Loption}, \texttt{recursive}) \\ &\mathit{allow}(a, b, p, c, x, \mathrm{FRoption}, \mathrm{Loption}, \texttt{recursive}) \\ &\quad \rightarrow \mathit{allow}(b, \_, p, a, x, \mathrm{FRoption}, \mathrm{Loption}, \texttt{recursive}) \end{aligned} \tag{9}$$

When an agent delegates an obligation, it does not directly instantiate *delOb*, but rather creates an instance of a *delObAttempt* predicate, which generates the corresponding *delOb* only if the delegation is valid according to the existing rights. It should be noted that *allow*s and *deny*s are terms which can be more or less instantiated (parameters can be ground litterals or uninstantiated variables), and thus more or less specific. By default, any delegation that is not allowed is forbidden, and *deny* has priority over *allow*. The overall rule for deriving an obligation delegation from a delegation attempt is described by (10).

$$\frac{\begin{array}{c}\mathit{delObAttempt}(\mathrm{A}, \mathrm{P}, \mathrm{B}, \mathrm{C}, \mathrm{X}, \mathrm{FRoption}, \mathrm{Loption}), \\ O(\mathrm{B}, \mathrm{P}, \mathrm{C}, \mathrm{X}), \mathit{allow}(\mathrm{B}, \mathrm{A}, \mathrm{P}, \mathrm{C}, \mathrm{X}, \mathrm{FRoption}, \mathrm{Loption}, \_), \\ \neg \mathit{deny}(\mathrm{B}, \mathrm{A}, \mathrm{P}, \mathrm{C}, \mathrm{X}, \mathrm{FRoption}, \mathrm{Loption}, \_)\end{array}}{\mathit{delOb}(\mathrm{A}, \mathrm{P}, \mathrm{B}, \mathrm{C}, \mathrm{X}, \mathrm{FRoption}, \mathrm{Loption})} \tag{10}$$

Obligation chains We have seen that an obligation can be delegated with or without delegating (or sharing) liability towards the original obligator. In order to decide whether an agent is liable towards another for a given obligation, one must know whether there is a chain of obligations (including both the initial one and the delegated ones) between them, and whether liability has been shared or forwarded at each step. This is what the *obChain* predicate does. $obChain(a, p, b, x, \text{L\_chain})$ means that there is a chain of obligations between b (obligator) and a (obligatee) about p, coming from the normative source x. The last parameter can be `l_propagated`, if liability has been kept (so that a may be liable to b), or `l_lost` if liability has been lost somewhere between a and b. This predicate is a convenience abbreviation defined as (11).

$$\begin{array}{l} obChain(a,p,b,x,\texttt{l\_propagated}) \stackrel{def}{=} \\ \quad \left\{ \begin{array}{l} O(a,p,b,x) \\ \vee \left(\begin{array}{l} obChain(c,p,b,x,\texttt{l\_propagated}) \\ \wedge \ \left(delOb(a,p,c,\_,x,\_,\texttt{l\_share}) \vee delOb(a,p,c,\_,x,\_,\texttt{l\_forward}) \right) \end{array} \right) \end{array} \right. \\ obChain(a,p,b,x,\texttt{l\_lost}) \stackrel{def}{=} \\ \quad \left\{ \left(\begin{array}{l} obChain(c,p,b,x,\texttt{l\_lost}) \\ \wedge \ delOb(a,p,c,\_,x,\_,\_) \end{array} \right) \vee \left(\begin{array}{l} obChain(c,p,b,x,\texttt{l\_propagated}) \\ \wedge \ delOb(a,p,c,\_,x,\_,\texttt{l\_keep}) \end{array} \right) \right. \end{array} \tag{11}$$

Functional responsibility (with delegation) Functional responsibility must be redefined in order to take obligation delegation into account. Now an agent or organization has functional responsibility for p if it is obliged to ensure p, but only if that it has not delegated this obligation with the `fr_forward` option (12).

$$FR(a,p,b,x) \stackrel{def}{=} obChain(a,p,b,x,\_) \wedge \neg delOb(\_,p,a,x,\texttt{fr\_forward},\_) \tag{12}$$

Causal responsibility by indirect influence (with delegation) Causal responsibility by influence is the only component of causal responsibility which is related to obligations, so it is the only one we need to reconsider in the light of obligation delegation. So far, we have only defined causal responsibility by direct influence, when the agent we have ordered to ensure p is itself materially responsible for it. We introduce causal responsibility by indirect influence, which captures the fact that this obligation can be further delegated. $CRII_a\ p$ (reading "a is causally responsible, by indirect influence, for p") means that there is a chain of delegated obligations on p between a and some agent b, that b is materially responsible for p, and that this is not a causal responsibility by direct influence (13). Causal responsibility by direct influence and by indirect influence are then grouped in a same "causal responsibility by influence" $CRI_a\ p$ (14).

$$CRII_a\ p \stackrel{def}{=} \neg CRDI_a\ p \wedge obChain(b,p,a,\_,\_,\_) \wedge MCR_b\ p \tag{13}$$

$$CRI_a\ p \stackrel{def}{=} CRDI_a\ p \vee CRII_a\ p \tag{14}$$

It can be interesting to introduce a variant operator: causal responsibility by primitive influence $CRPI_a\ p$, meaning that the issued obligation has not been inherited by delegation (15). Overall causal responsibility is then redefined as

the disjunction between material causal responsibility and causal responsibility by influence (16).

$$CRPI_a\ p \stackrel{def}{=} CRI_a\ p \wedge \neg delOb(a, p, \_, \_, \_, \_) \tag{15}$$

$$CR_a\ p \stackrel{def}{=} RCM_a\ p \vee CRI_a\ p \tag{16}$$

Liability (with delegation) The last notion to be redefined is liability, for which the *obChain* predicate has been specially tailored. An agent or organization a is directly liable for p towards b if and only if a is causally responsible for p, there is an obligation chain propagating liability from b to a, this liability has not been lost by delegation and no other agent is accountable for a (17). Indirect liability (18) can be defined in the same way, with overall liability remaining the disjunction of direct and indirect liability.

$$DL(a, p, b, x) \stackrel{def}{=} CR_a\ p \wedge PDL(a, p, b, x) \tag{17}$$

$$IL(a, p, b, x) \stackrel{def}{=} CR_c\ p \wedge PIL(a, p, b, x) \tag{18}$$

Concrete Example [4] Let us assume that agent a has the obligation, towards b, to fulfill p, and a is allowed to delegate this obligation with recursive option (figure 1). If the obligation to ensure p is violated then we have to identify agents that are responsible of this violation, namely, functional responsibility and liability (which is derived from causal responsibility). As shown in figure 1, agent a delegates the obligation to c and shares both the functional responsibility and the liability towards b, so we can derive that $FR(a, p, b, x)$ and $L(a, p, b, x)$. Agent c delegates the obligation to d with `fr_share` option and keeps the liability, so d has the functional responsibility towards b. Then, c delegates the obligation to e and forwards both the functional responsibility and the liability. Therefore, c is no more responsible towards b. Finally, agent e forwards the functional responsibility to f. Thus, e is liable for p and f has the functional responsibility towards b. To summarize, if the obligation $O(a, p, b, x)$ is violated then we have $FR(act, p, b, x)$, for act in $\{a, d, f\}$, and $L(act', p, b, x)$ for act' in $\{a, e\}$. Moreover, as mentioned above, agents are also "locally" responsible towards the agent who delegated to them the obligation (directly or indirectly), but only if they have not forwarded this responsibility to another agent. This is why, agent d has the functional responsibility towards agents c and a, and is liable towards c. Agent e is also liable towards agents c and a. Finally, agent f is liable towards e and has functional responsibility towards e, c and a. Note that if the obligation is fulfilled by an agent belonging to the obligation chain, then we consider that all the other agents have fulfilled their obligations [1]. In addition, we consider that only agents having functional responsibility are obliged to perform the obligation. Otherwise, the obligation is inactivated, i.e. the obligations of agents c and e. After identifying agents that are liable towards b, namely a and e, sanctions

[4] We give here a basic example to illustrate our approach. Real life situations will be given in the next section 4 to help to understand the issues of our work.

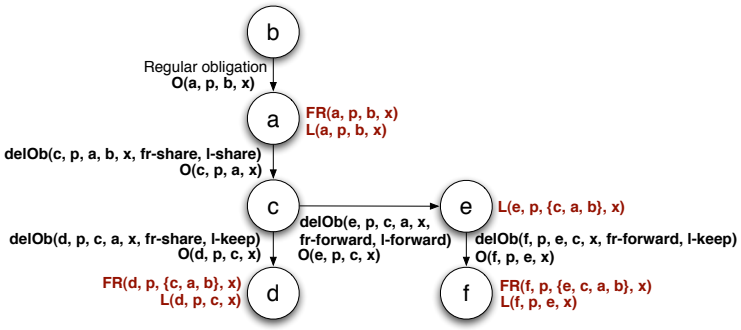

Fig. 1. The obligation chain

are derived. According to the sanction defined by norm x and according to the liability level (i.e. blameworthiness) and kind (i.e. penal or civil liability), a and e will be (or not) sanctioned, asked to repair a damage or a loss. "Local" sanctions can also be defined by norm x for agents who did not fulfill the delegation contract. Thus, agent d, which is "locally" liable to c, can be sanctioned by c if the obligation is violated. Obviously, this liability is inactivated if there is a hierarchical authority relation between the obligatee and the obligator (i.e. $relation(\texttt{bossOf}, d, c)$). In the same way for agents e and f.

4 Related work and Discussion

In the literature, some works [13,18,6] have studied the issue of the delegation of obligations, such as the share, the transfer or the split of obligations, others [5,17,20,12] have focused on the definition of responsibilities, such as direct, causal or task-based responsibility, and in [7] authors have addressed the issue of accountability within delegation protocols. But, none of them has explored the delegation of responsibility as we have done in this paper. These works have only considered the basic levels of delegation, namely the delegation of the obligation without responsibility or the transfer of the obligation together with the responsibility. In our work, we have proposed a distinction between functional responsibility and liability, in order to give agents the means to delegate their obligations according to their requirements and abilities. Moreover, this distinction allows us to identify, in the case of delegation, agents that are responsible for the violation, agents that are liable (or indirectly liable) for this violation and finally agents who are to be sanctioned.

Even though the word delegation has been used (and defined) in a technical sense in this paper, our notion of delegation can be applied in real life situations in which a proper distinction between functional, causal and legal responsibilities could help clarifying the issues and drawing appropriate conclusions. As an illustration, existing privacy protection regulations (such as the European

Directive) impose strong obligations on any entity which collects personal data (the "data controllers"). In particular, data controllers are responsible for the security of the data and must ensure that the data subject can effectively exercise their rights, for example their rights to get access to their data or to have them corrected in case of error, or deleted if they are no longer necessary for the purpose. If the data controller subcontracts some or all the treatment of the personal data, certain responsibilities are transferred when others are shared or kept by the data controller. For example the functional responsibility to ensure the security of the data is shared (the data controller must implement appropriate security measures for the collection of the data and their transfer to the subcontractor and the subcontractor must ensure the protection of the data storage and access) but the legal responsibility for security (w.r.t. the data subject) may remain with the data controller, so that the data subject could potentially sue the data controller for security breaches actually due to the subcontractor. On the other hand, the functional responsibility on the exercise of the rights of the subject may be completely transferred (if the data controller does not store any data by himself) and both the data controller and the subcontractor must address any request from the subjects concerning their personal data, thus sharing legal responsibility.

Another illustration of the generality of our framework is the application to software contracts and responsibilities for defective software. As stated in section 2, different notions of causal responsibility can be defined, which may correspond to different levels of severity. The notion of causality has also been studied for a long time in computer science, but it is usually seen essentially as a temporal property. In [9], we have defined several variants of logical causality allowing us to express the fact that an event e_2 (e.g. a damage) would not have occurred if another event e_1 had not occurred ("necessary causality") or the fact that e_2 could not be avoided as soon as e_1 had occurred ("sufficient causality"). These causality properties are expressed in terms of execution traces of the software components. We have shown that they are decidable and proposed trace analysis procedures to establish them. These notions of causality are examples of causal responsibilities relations CR which can be used to apply the framework presented here to software liability. This would make it possible to formalize legal aspects of the liability framework proposed in [11] and to distinguish, for example, the technical commitments of a subcontractor (e.g. providing a software component with a given functionality) and the cases of misbehaviour giving rise to a legal liability on his part (e.g. if the output of the component exceeds a given threshold, which might put the system or its environment at risk). An interesting avenue for further work to this respect is the introduction of group liability allowing us to make a distinction between "joint liability" (when each party is considered fully responsible for the obligation) and "several liability" (when the parties are responsible for their respective part of the obligation).

Acknowledgments. This research has been supported by the ANR 07 SESUR FLUOR project.

References

1. Ben-Ghorbel-Talbi, M., Cuppens, F., Cuppens-Boulahia, N.: An extended role-based access control model for delegating obligations. In: Trust, Privacy and Security in Digital Business. LNCS, Springer (2009)
2. Ben-Ghorbel-Talbi, M., Cuppens, F., Cuppens-Boulahia, N.: Negotiating and delegating obligations. In: International Conference on Management of Emergent Digital Eco-Systems (MEDES) (2010)
3. Bettini, C., Jajodia, S., Wang, X.S., Wijesekera, D.: Provisions and obligations in policy rule management. Network and Systems Management 11(3) (2003)
4. Cholvy, L., Cuppens, F., Saurel, C.: Towards a logical formalization of responsibility. In: 6th international conference on Artificial intelligence and law. ACM Press, Australia (1997)
5. Cholvy, L., Garion, C., Saurel, C.: Ability in a multi-agent context: A model in the situation calculus. In: Computational Logic in Multi-Agent Systems. Springer (2005)
6. Cole, J., Derrick, J., Milosevic, Z., Raymond, K.: Author obliged to submit paper before 4 july: Policies in an enterprise specification. In: Policies for Distributed Systems and Networks (2001)
7. Crispo, B., Ruffo, G.: Reasoning about Accountability within Delegation. In: Information and Communications Security. LNCS, Springer (2001)
8. Cuppens, F., Cuppens-Boulahia, N., Ghorbel, M.B.: High level conflict management strategies in advanced access control models. Electronic Notes in Theoretical Computer Science 186 (2007)
9. Gössler, G., Le Métayer, D., Raclet, J.B.: Causality analysis in contract violation. In: Runtime Verification (2010)
10. Irwin, K., Yu, T., Winsborough, W.H.: Assigning responsibility for failed obligations. In: IFIP Trust Management Conference (2008)
11. Le Métayer, D., Maarek, M., Mazza, E., Potet, M.L., Frénot, S., Viet Triem Tong, V., Craipeau, N., Hardouin, R.: Liability in software engineering - overview of the lise approach and illustration on a case study. In: 3rd International Conference on Software Engineering (2010)
12. Mastop, R.: Characterising responsibility in organisational structures: The problem of many hands. In: Deontic Logic in Computer Science. LNCS, Springer (2010)
13. Pacheco, O., Santos, F.: Delegation in a role-based organization. In: DEON. LNCS, Springer (2004)
14. Park, J., Sandhu, R.: The UCON$_{ABC}$ Usage Control Model. ACM Transactions on Information and System Security 7(1), 128–174 (2004)
15. Pretschner, A., Hilty, M., Basin, D.: Distributed usage control. Communications of the ACM 49(9), 39–44 (2006)
16. Pörn, I.: Action theory and social science: Some formal models. Synthese Library 120 (1977)
17. Royakkers, L., Grossi, D., Dignum, F.: Responsibilities in organizations. In: Computer Supported Activity Coordination (2006)
18. Schaad, A., Moffett, J.D.: Delegation of obligations. In: Policies for Distributed Systems and Networks. USA (2002)
19. Sergot, M.: Norms, action and agency in multi-agent systems. In: Deontic Logic in Computer Science. LNCS, Springer (2010)
20. Strens, R., Dobson, J.: How responsibility modelling leads to security requirements. In: Workshop on New Security Paradigms. United States (1993)
21. Wright, G.H.v.: Deontic Logic. Mind 60, 1–15 (1951)