Representing privacy regulations with deontico-temporal operators*

Guillaume Piolle† and Yves Demazeau‡

2011

Abstract

The aim of this study is to provide artificial agents with logical tools to reason specifically on privacy-related regulations, in order to comply with them. In order to express these regulations, the Deontic Logic for Privacy (DLP) is proposed. DLP is a deontic and temporal logic based on predicates dealing with personal data management. Illustrated by an example, it becomes obvious that specific operators are needed to express mixed deontic and temporal notions such as obligations with deadlines and maintained interdictions. A set of eight specific requirements is defined for such operators. The existing proposals in the field are then evaluated with respect to these criteria, and found insufficient. Two new operators are tailored to fit both the DLP formalism and the eight requirements. It is then shown how such operators can be used to translate typical privacy regulations in logical formulae via the DLP language.

Keywords deontic logic, temporal logic, deadlines, privacy

1 Introduction

Commercial service providers tend to put a stress everyday more significant on personalization, thus collecting more and more personal data from users. In consequence, they have to provide them with guarantees on the protection of their private information. The users' concern for privacy has indeed grown jointly with the thirst for personalization. Intelligent agents aiming at interfacing their human user with distant or distributed services should then be provided with tools for protecting private information, in application of the privacy regulations relevant to their execution context.

In the general domain of privacy, research has first focused on the so-called *Privacy-Enhancing Technologies* (PETs), the technical means used to protect individuals' information. The ISO, for instance, gave in 1999 a set of "common criteria" to be met by such systems, in terms of technical features [15]. In 2006, Deswarte classified existing Internet-oriented PETs with respect to their aims (identity management, IP anonymization...) [12] and a classification of privacy-related regulations along six dimensions (user information, user consent, data update and retractation, process justification, data retention and data forwarding) was proposed the same year [20]. Regarding the needed reasoning means on these regulations, much work has been done on the representation of security policies using deontic logic. One could cite for instance the early works of Ortalo [16] or the proposals of Cholvy and Cuppens on the consistency of such policies [8, 9]. These works seem to provide means to represent the normative context of an

*This document is an author version adapted from [19].

†Supélec, Campus de Rennes, CS 47601, Avenue de la Boulaie, 35576 Cesson-Sévigné Cedex, France – guillaume.piolle@supelec.fr

‡CNRS, Laboratoire d'Informatique de Grenoble, Maison Jean Kuntzmann, 110 avenue de la Chimie, Domaine Universitaire, F-38400 Saint-Martin-d'Hères – yves.demazeau@imag.fr

agent. So far, none of the logic-based proposals include privacy-specific reasoning features, even though some of them deal with closely related fields like information exchange policies [10].

The need for temporal dynamism is particularly prominent in privacy regulations, which often deal with notions such as durations and deadlines. Such norms may state, for instance, that a piece of information should be deleted before a given date or after a given duration, or that user information must take place at least N days before her data are used. Techniques for dealing jointly with deontic and temporal notions has often been studied, most notably by Åqvist [2], and some authors have been working "in the abstract" on the general notion of deadlines in deontic obligations [13, 11, 4]. This notion is actually crucial to the representation of privacy norms, and these contributions need to be thoroughly analyzed before developing a similar operator adapted to the specificities of privacy regulations.

What is proposed here is to build of a deontic and temporal logic dedicated to the representation of privacy regulations and to make sure that it is expressive enough for the deontic and temporal notions appearing in privacy-related regulations.

In the following section, the formalization needs already mentionned are explicited, and the existing proposals for dealing with mixed deontic and temporal notions (and most notably the notion of obligations with deadlines) in logical frameworks are examined. In the next section, the *Deontic Logic for Privacy* (DLP) is introduced as a formalism designed to deal with privacy-specific notions. In section 4, eight requirements for an "obligation with deadline" operator are introduced and discussed. The existing proposals are adapted to the DLP formalism and evaluated with regards to these requirements, before it is examined how a more suited operator can be built on this basis. The notion of interdictions maintained over time is also introduced and a corresponding operator is proposed in the DLP formalism. In section 5, these operators are used to translate fictional but representative privacy regulations into DLP norms, thus showing how they could be used by an autonomous agent.

In the following will be considered, as a generic applicative context, a **service agent**, in charge of a process (or service) involving personal information, and a **user agent**, willing to obtain that service and responsible for the personal information of its human owner.

2 Formalization needs in privacy regulation frameworks

In order to design an auto-adaptative privacy-preserving system, it is necessary to examine what the regulations potentially constraining the system can look like. Several authorities can enact such norms: international and local legal systems, corporate regulations, contracts, individual directions, user preferences... From the point of view of personal data protection, they all deal with the same concepts. It is on the basis of these concepts that the expressiveness requirements of the formal system must be established.

2.1 The six dimensions of privacy regulations

In the example of the European legal context, which will be used in this study, the generic notions of personal data protection have been introduced by the European directive 95/46 [26]. The directive 02/58 [27] represents an example of the specialization of these principles to a particular field, in this case electronic communications. These regulations have then been implemented in the national legal frameworks of the member states. In France, for instance, the original data protection law [23] was amended in 2004 [24] in order to cope with the European directives.

In these texts, one can observe that the regulations directly related to personal data protection are distributed along six dimensions, which can also be found in other sources of privacy-related norms [20]. These dimensions refer to pieces of personal data, containing information about their owner (who will be called "the user" from now on), the collection of these data, their processing, the entity responsible for those actions (which can be conveniently considered as a "service provider") and the other possible recipients of the data. The six dimensions define the subject of the regulations, but do not make assumptions about how restrictive they should be.

1. **Information:** Texts say whether, when and to which extent the user must be informed of the data collection and of their processing. Usually, save exceptional cases like juridic actions, national security and public health issues, the service provider must give explicit information (covering the other five dimensions) to the user.
2. **Consent:** It is usually required that the user give her consent about the data collection and processing before they take place. This consent can eventually be explicit, *a priori* or by default.
3. **Modification:** In some cases the user has the right to request access to the collected data and to ask for them to be corrected or deleted (if they appear to be erroneous, outdated or not needed anymore).
4. **Justification:** This dimension is the link between the collected data and its intended use. It covers the proportionality and minimality principles introduced by the 1995 directive, which requires that the collection of any data be justified by their intended use (thus forbidding the collection of unrelated information).
5. **Data retention:** There can be a minimum and more often a maximum to the duration of data retention. These constraints usually depend on the nature of the data and on the application.
6. **Data forwarding:** The forwarding of personal information by the service provider to third parties can be limited or forbidden by regulations. Most often, it is subject to an explicit authorization by the user.

The translation of human-readable regulations into formal norms is then to cover these six dimensions. Therefore, the language to be used must be designed on the basis of predicates able to deal with their semantics.

It is worth noting that these dimensions refer only to the privacy-related regulations, and not to the techniques and methods used to enforce them. There are other possible axes for such notions. The "Common Criteria" of the ISO/IEC [15] are an example. They define the four requirements of a privacy-enhancing technology for it to be acceptable as such: anonymity, pseudonymity, unlinkability and unobservability. One could note that these requirements do make assumptions about what is acceptable or not, whereas the present work stays as neutral and generic as possible. Deswarte and Aguilar-Melchor propose yet another kind of classification [12], by organizing Internet-based privacy-enhancing technologies into five categories (identity management, anonymous IP communication, anonymous access to services, privacy-preserving authorization and personal data management). These three organization schemes are complementary and do not relate to the same aspects of the issue. The focus of the present study being on regulation representation and reasoning, only the first set of dimensions (information, consent, modification, justification, retention and forwarding) is relevant here.

2.2 Mixing deontic and temporal notions

In order to represent laws and regulations, one has to model the notions of obligation and interdiction, at the very least. Various kinds of deontic logics have been designed with this purpose. The Standard Deontic Logic (SDL) [29] has the advantage of simplicity, being a normal modal logic with a KD operator Ob representing obligation and allowing the expression of interdiction (For) and permission (Per) by the means of abbreviations. Although plagued with quite a number of dilemmas and paradoxes, SDL is widely used as a basis for more complex deontic frameworks. However, if needed, deontic notions cannot alone represent the modalities of a regulatory text, especially in the domain of personal data protection. Indeed, such regulations almost always make reference to time-related notions. For instance, regulations regarding information may say how long in advance the user should be informed, modification-related norms may indicate how much time the service provider has to update data after a modification request, and retention instructions have to deal with time in a rather obvious way.

As an example, one could consider the following (fictional) norm, dealing with user information:

Users must be informed at least one week in advance of the nature of any process involving their personal information.

This sentence looks rather simple, and not uncommon. It could be part of a corporate regulation, for instance. In such norms there is a strong collaboration, if not an intrication, between the deontic and the temporal notions in order to represent the full semantics of the regulations. Submitted to this example norm, one could interpret it in two complementary ways:

- If one has the goal or intention to perform a process involving personal information at a given date situated more than one week in the future, then one must take into consideration an **obligation with deadline**: one must inform the owner of the data before a given date, which is situated one week before the process date;
- Otherwise and in any case, if one has not informed the owner of a given piece of personal information about a process in which it could be involved, then one faces a **maintained interdiction**: it is forbidden to perform such a process before at least seven days have passed. Of course, it does not mean that a permission to do so then arise by itself.

There are actually other cases to be examined by the agent, like when a process is scheduled less than one week in the future and the user has not been informed. In this case, the unavoidable violation is triggered by the first form of the norm. The reader can check that the remaining possibilities are covered by these two norm descriptions.

In conclusion, depending on whether one considers the norm with a teleologic or with a constraining point of view, it can be viewed as containing either an obligation with deadline or a maintained interdiction. These two notions can also be used to represent and interpret norms related to the other five dimensions of privacy-related regulations. They crystallize the needed relationship between obligation and time.

2.3 Existing proposals

Several research works have proposed formal models for dealing jointly with both deontic and temporal notions, either by setting up adapted reasoning framework, or more specifically by proposing operators for the modelling of the concept of obligation with deadline.

2.3.1 Deontico-temporal formalisms

In 1980, Chellas proposed the introduction of temporal modalities in SDL [7]. In his logic, deontic operators are closely linked with operators of temporal necessity and possibility. It has proved difficult for software agents to reason with these temporal and deontic notions, and the Propositional Linear-time Temporal Logic (PLTL, or simply LTL), introduced by Pnueli [21], has soon appeared as a more serious candidate for the expression of temporal notions, because of its ability to reason explicity on both future and past. In the following, it is assumed that the reader is familiar with the corresponding operators: P and F are the existential operators over past and future, respectively, H and G the corresponding universal modalities (that is $H\varphi = \neg P\neg\varphi$ and $G\varphi = \neg F\neg\varphi$), X^{-1} and X allow to access previous and next time steps and $\mathcal{U}$ and $\mathcal{S}$ read as "until" and "since". All these operators are understood in their strict version, *i.e.* excluding the present. Loose versions are noted with a minus in subscript (*e.g.* F^{-}).

Some STIT (for "See To It That") logics are also able to represent both deontic and temporal concepts [14]. However, they do not seem to constitute an adequate candidate because the STIT version of the deontic operator is deeply linked with the notions of ability and responsibility. Namely, it is impossible to express an obligation on some truth value that the agent does not control. Using such formalisms would forbid any representation of conflicting or absurd obligations, which may be needed to comprehend the normative context of an agent. Mixing the standard deontic and temporal notions seems to be a better, more flexible option.

The Normative Temporal Logic (NTL) proposed by Ågotnes *et al.* [1] is one example of logic based on computational tree logic and SDL. It has the advantage of differentiating normative authorities, but it intrisically forbids that they enact conflicting norms, which is not suitable

for the purpose of reasoning on arbitrary privacy-related regulations coming from independent authorities. It also restricts the joint use of deontic and temporal logic operators (for the purpose of cutting computational complexity down), thus seriously limiting their expressiveness potential.

In 2004, Åqvist proposed an analysis of the methods for mixing temporal and deontic modalities [2]. He distinguishes time indexation methods (like in Chellas's proposal) and the use of temporal operators (like in Ågotnes's proposal). He emphasizes the use of multi-dimensional modal logics and their associated multi-dimensional semantic structures (for instance, one dimention for deontic possibilities and one dimension for the axis of time). He introduces "systematic frame constants", allowing to manipulate a coordinate in the semantic grid. This concept allows the representation of dates, which is a necessary starting point if one wants to work with deadlines.

2.3.2 Proposals on obligations with deadlines

Once one is able to deal jointly with obligations and time, the concept of obligation with deadline arises naturally. Indeed, a pure obligation in a temporal framework is often considered meaningless. In the first possible interpretation, it is an immediate obligation, and then an agent cannot do anything in order to respect it: either it is already the case, or it is not and therefore it is too late, a violation has already occured. On the other hand, it could be an obligation to do something "at some point in the future" (which can be expressed as $Ob\ F\varphi$ with the standard operators of LTL and SDL). In that case, the agent has the ability to respect the obligation, but can also indefinitely postpone the corresponding action, if the axis of time is right-unbounded. Violations are then impossible to characterize. For these reasons, at least three research tracks have introduced the notion of obligations with deadlines. An operator of obligation with deadline is such that the target proposition must become true at some point after the obligation is first issued, and before the stated deadline. Otherwise, the violation can be characterized at the deadline.

It should be precised that here we consider violation as the failure to comply with the obligation within the time allowed. Therefore we do not make a distinction between "obligation violations" and "deadline violations" like in the proposal of Cardoso and Oliveira [5], which is based on different considerations allowing the introduction of deadline extensions or other kind of arrangements between the parties. This does not seem to be highly desibrable in the present model specifications because in many cases, the management of deadlines regarding personal data protection is strictly fixed by law.

In 2004, Dignum *et al.* [13] propose an operator for obligations with deadlines, formalized in a purely temporal framework based on CTL, to which is added a STIT operator E_a (expressing an action of agent a leading to a given truth value). The corresponding operator $O_a(\varphi < \delta)$ reads "a must make φ become true before δ becomes true". It is defined jointly with the corresponding violation operator. The main characteristics of this proposal is that the operators is not based on a deontic modality but on temporal chains of events, and that the violation is defined as a state. The latter is necessary here because the chosen set of operators does not permit any reasoning on the past.

In 2005, Demolombe *et al.* [11] construct a similar operator in a quite different formalism, based on an earlier proposal by Segerberg [25]. This formalism being very different from the classical modal logics of time and obligations, it might be easier to focus on its expression in natural language:

> There is an obligation with deadline $Ob_a(\varphi, \delta)$ if and only if for any possible future, there is an immediate obligation on agent a to perform φ before δ, holding until either φ or δ becomes true.

In this case the authors rely on an existing deontic notion, and define the violation separately.

In 2006, Brunel *et al.* [4] designed their operator in a modified LTL $\times$ SDL logic. They introduce operators such as $F_{\leq k}\varphi$, meaning that φ will become true in k time steps at most. The authors present these operators as an extension to the language, but since k is always finite, they can actually be defined in terms of standard LTL operators. The core extension in expressiveness is the ability to use explicit quantification over the number of time steps. This allow them to

reason on durations rather than on deadline propositions. In order to define an operator for obligations with deadlines, they first introduce a intermediate operator, to be used only when the obligation is first expressed. On this basis, they build a more complex operator $Ob_k(\varphi)$ (k being the distance to the deadline), described by the following sentence:

> k' time steps ago, there was an obligation to satisfy φ before $k + k'$ time steps, and the obligation has not been fulfilled yet.

This operator is defined only by its formal semantics, because it needs an existential quantification on k' (and on other numbers not showing in the natural language phrasing), along with comparisons between delays which do not translate directly in LTL.

2.4 Synthesis

To conclude this overview of the available tools, there is apparently no specialized formalism already proposed to deal with privacy-related norms. Yet, such regulations need very specific tools, like methods for dedicated reasoning about the six regulatory dimensions of privacy, or specialized operators dealing with obligations with deadlines and maintained obligations. The design of such operators must rely on both deontic and temporal notions, which should therefore be available in the logical framework.

The most generic and simple way of achieving this is probably to build a language around SDL and LTL. More complex logics such as STIT logics could also provide the necessary notions, but they also bring with them a number of restrictions on how the temporal and the deontic concepts must interfere with each other. Namely, working on impossible regulations and normative conflicts can be difficult or impossible in such languages, this is why a language with fewer presuppositions should be preferred.

3 The DLP logic

A new language, designed on the basis of existing work in the domain, is proposed to represent and reason on privacy-related regulations. The Deontic Logic for Privacy (DLP logic) is a normal deontic and (linear) temporal language, the corresponding modalities being applied on a base language $\mathcal{L}_{DLP}$ designed to represent information and actions about personal data usage and protection.

In the following, predicate functors will be in italic font (like *request*), litteral values in monospaced font (like `self`) and variables in standard font with a Prolog-like leading capital (like UserAgent).

3.1 Privacy-related information

In $\mathcal{L}_{DLP}$, processes making use of personal information are identified through a unique identifier (ProcessID). Processes have one or more action types (ActionType) to define their nature. Each process is related to a set of personal information chunks, each of them being identified by a unique identifier inside a process (DataID) and characterized by one or more data types (DataType). Data identifiers are local to a process, it means that a piece of information is uniquely identified by a couple (ProcessID, DataID). In the current version of the language, action types and data types take their values in the various sets and type trees defined by the P3P standard schemata [31]. For instance, `user.home-info.telecom.telephone` is a data type value representing the home telephone number of a user. $\mathcal{L}_{DLP}$ language is based on three sets of predicates, on which the connectives of propositional logic ($\neg, \vee$) are applied. **Informative predicates** are designed to represent the service agent informing a user agent with respect to a given process:

- *informActionType*(ServiceAgent, UserAgent, ProcessID, ActionType) says that ServiceAgent informs UserAgent that the process ProcessID is of the type ActionType;

- *informForward*(ServiceAgent, UserAgent, ProcessID, DataID, AgentList) says that ServiceAgent informs UserAgent that data DataID of process ProcessID may be shared with agents from AgentList;
- *informDuration*(ServiceAgent, UserAgent, ProcessID, DataID, Duration) says that ServiceAgent informs UserAgent that a piece of information may be kept for a duration Duration;
- *informContact*(ServiceAgent, UserAgent, ProcessID, ContactAgent) says that ServiceAgent informs UserAgent that ContactAgent is the agent to contact for queries related to the process ProcessID.

Performative predicates are designed to represent some actions and speech acts taken by the agents:

- *request*(ServiceAgent, UserAgent, ProcessID, DataType, DataID) is the request, from ServiceAgent to UserAgent, of an information of type DataType for the process ProcessID;
- *requestConsent*(ServiceAgent, UserAgent, ProcessID) is a request for the explicit consent of UserAgent;
- *consent*(UserAgent, ServiceAgent, ProcessID) represents this consent;
- *tell*(ServiceAgent, UserAgent, ProcessID, DataID) represents the actual transmission of the piece of information DataID of the process ProcessID;
- *perform*(ServiceAgent, ProcessID) represents ServiceAgent actually running the process ProcessID;
- *updateRequest*(UserAgent, ServiceAgent, ProcessID, DataID, UpdateID) represents a request (identified by UpdateID) from the user agent to update the piece of information DataID of the process ProcessID;
- *update*(ServiceAgent, ProcessID, DataID, UpdateID) represents the service agent actually updating the data (in response to the update request UpdateID);
- *forgetRequest*(UserAgent, ServiceAgent, ProcessID, DataID) represents a request from UserAgent to permanently delete the piece of information DataID of the process ProcessID;
- *forget*(ServiceAgent, ProcessID, DataID) represents ServiceAgent actually deleting the data;
- *forward*(ProcessID1, DataID1, ProcessID2, DataID2) represents the forwarding of the piece of information DataID1 of process ProcessID1 to another agent, through the correlation between two processes (ProcessID1 and ProcessID2).

State predicates are used to store information about processes and data, in the form of relational tuples:

- *actionType*(ProcessID, ActionType) records the type of a process;
- *dataType*(ProcessID, DataID, DataType) records the type of a piece of information;
- *forwardList*(ProcessID, DataID, RecipientAgent) represents the fact that RecipientAgent may have access to a piece of information;
- *duration*(ProcessID, DataID, Duration) records the stated retention duration of a given piece of information;
- *responsible*(ServiceAgent, ProcessID) is used to record that ServiceAgent in charge of process ProcessID;
- *contact*(ContactAgent, ProcessID) is used to record that ContactAgent is the contact agent for process ProcessID;
- *owner*(ProcessID, DataID, UserAgent) is used to record that UserAgent is the actual owner of data DataID of process ProcessID.

Any instance of these predicates can be true or false at a given instant in the timeflow, thus allowing a precise description of past and scheduled events. One must note though that $\mathcal{L}_{DLP}$ is not an agent communication language. Its only aim is to internally represent the information about the speech acts taken by the agents, not to embody them.

For each predicate in the language, each argument takes its value in a domain which is a finite or countable set. Therefore, the set of all possible $\mathcal{L}_{DLP}$ predicate instances is countable and can be mapped onto a set of (virtual) propositional atoms. In the case one of the arguments is not bound in an expression, then the predicate represents a propositional schema (like axiomatic schemata), covering a whole set of propositional atoms. One should note that there are no explicitly quantified formulae in the language. This is why, even though information is represented in the form of predicates, it is safe to consider a modal logic based on $\mathcal{L}_{DLP}$ as a propositional modal logic.

3.2 DLP Syntax

DLP is a language where the obligation modality Ob of Standard Deontic Logic is freely mixed with temporal operators from Propositional Linear-time Temporal Logic. The well-formed formulae φ of the DLP language are defined by the grammar in Eq. (1), where p is a $\mathcal{L}_{DLP}$ proposition. Of course, the abbreviations $\wedge$ and $\rightarrow$ (defined as usual) and the parentheses, although not formally included in the grammar, can be used in DLP formulae.

$$\begin{array}{rl} \varphi = & p \mid \varphi \text{ “}\vee\text{” } \varphi \mid \text{“}\neg\text{” } \varphi \mid \text{“}Ob\text{” } \varphi \\ & \varphi \text{“ }\mathcal{U}\text{ ” } \varphi \mid \text{“}H\text{” } \varphi \mid \text{“}X^{-1}\text{” } \varphi \ ; \end{array} \tag{1}$$

One should note that the logic presented here is actually a simplified version of DLP. The complete version [17] makes use of a class of obligation modalities Ob_a^ν (rather than the single one Ob used here), differentiated by agent (a) and by normative authority (ν). This is to allow normative conflict representation and arbitration, as it has been shown in a previous communication [18], but it does not bring anything essential to the problem addressed here.

The chosen temporal language allows much expressiveness on the future, through a strict “until” operator $\mathcal{U}$. A loose until $\mathcal{U}^-$, including the present, is defined as well in Eq. (2). Since this logic is to be used by an agent to reason about its own behaviour and the plans it chooses to execute, more expressiveness is needed in the future than in the past, where a “since” operator has not proved to be essential. DLP thus relies on an universal modality H on the past, and a X^{-1} operator pointing to the previous time step (DLP is to be interpreted over non-dense timeflows). Its symmetric operator in the future is a “next” modality X, defined as an abbreviation in Eq. (3). It could be convenient to use the usual temporal abbreviations G (universality in the future, Eq. (5)), F (existential dual of G, Eq. (4)) and P (existential dual of H, Eq. (6)), none of them including the present. Their loose counterparts G^-, H^-, F^-, P^-, including the present, are defined in Eq (7)-(10).

$$\varphi \,\mathcal{U}^-\, \psi \stackrel{def}{=} \psi \vee (\varphi \wedge \varphi \,\mathcal{U}\, \psi) \tag{2}$$

$$X\varphi \stackrel{def}{=} \bot \,\mathcal{U}\, \varphi \tag{3}$$

$$F\varphi \stackrel{def}{=} \top \,\mathcal{U}\, \varphi \tag{4}$$

$$G\varphi \stackrel{def}{=} \neg F \neg \varphi \tag{5}$$

$$P\varphi \stackrel{def}{=} \neg H \neg \varphi \tag{6}$$

$$G^-\varphi \stackrel{def}{=} \varphi \wedge G\varphi \tag{7}$$

$$H^-\varphi \stackrel{def}{=} \varphi \wedge H\varphi \tag{8}$$

$$F^-\varphi \stackrel{def}{=} \varphi \vee F\varphi \tag{9}$$

$$P^-\varphi \stackrel{def}{=} \varphi \vee P\varphi \tag{10}$$

It is also possible to design, on the basis of the existing temporal modalities, a class of indexed X^i operators, based on X and X^{-1}, which can be used to travel step-wise along a timeflow. They are designed in Eq. (11).

$$\begin{cases} X^0\varphi \stackrel{def}{=} \varphi \\ \forall n \in \mathbb{Z}_+^*, X^n\varphi \stackrel{def}{=} XX^{n-1}\varphi \\ \forall n \in \mathbb{Z}_-^*, X^n\varphi \stackrel{def}{=} X^{-1}X^{n+1}\varphi \end{cases} \tag{11}$$

The axiomatics of DLP logic will not be detailed here, since it is not crucial to the understanding of the current issues. The individual axiomatics of the operators are the ones classically used in SDL [29] and LTL [21] [3, chap. 7], with conversion axioms between the different temporal modalities. Conversion axioms between deontic and temporal modalities are not included in the logic, in order to allow the norms to be dynamic, independently from the framework's restrictions. The full axiomatization is available in [17, chap. 2 & 4] for the interested reader.

3.3 DLP Semantics

DLP formulae are interpreted over Kripke-like bi-dimensional structures, inspired by the proposals of Åqvist [2] and called DLP models.

Definition 1 (DLP model) *A DLP model $\mathcal{M}$ is a set $\{\mathcal{H}, \mathcal{T}, \rightsquigarrow, \mathbb{O}, v\}$ where:*

- *$\mathcal{H}$ is a countable set of objects ($h, h', h''... \in \mathcal{H}$) called histories;*
- *$\mathcal{T}$ is a countable set of objects ($t, t', t''... \in \mathcal{T}$) called dates. A couple from $\mathcal{I} = \mathcal{H} \times \mathcal{T}$ ($i, i', i''... \in \mathcal{I}$) is called an instant (or a world, for DLP interpretation is based on them);*
- *$\mathbb{O}$ is a (serial) binary relation among instants ($\mathbb{O} \subset \mathcal{I} \times \mathcal{I}$) called the deontic accessibility relation;*
- *$\rightsquigarrow$ is a binary relation (serial, left-unbounded, linear in both directions) among dates ($\rightsquigarrow \subset \mathcal{T} \times \mathcal{T}$) called the temporal accessibility relation;*
- *v is an application from $\mathcal{L}_{DLP}$ to the powerset $\wp(\mathcal{I})$ called the valuation function.*

A DLP model is then a bi-dimensional grid where an instant i is the point defined by a couple of coordinates (h, t). A particular date t_0 can be identified as an arbitrary reference point for the timeflow. The truth of $\mathcal{L}_{DLP}$ propositions and DLP formulae is defined for instants, which are the elementary worlds of the structure. For each instant $i = (h, t)$, the equivalent notations $\mathcal{M}, h, t \models \varphi$ and $\mathcal{M}, i \models \varphi$ read, respectively, "φ is true at date t in history h of model $\mathcal{M}$" and "φ is true at instant i of model $\mathcal{M}$". We define the relation $<$ (resp. $\leq$) among dates as the transitive (resp. transitive and reflexive) closure of $\rightsquigarrow$. DLP formulae are interpreted over DLP structures as defined in Eqs. (12)-(15) (with $p \in \mathcal{L}_{DLP}, \varphi, \psi \in DLP$).

$$\mathcal{M}, h, t \models p \text{ iff } (h, t) \in v(p) \tag{12}$$

$$\mathcal{M}, h, t \models \neg\varphi \text{ iff } \mathcal{M}, h, t \not\models \varphi \tag{13}$$

$$\begin{array}{c} \mathcal{M}, h, t \models \varphi \vee \psi \\ \text{iff } (\mathcal{M}, h, t \models \varphi \text{ or } \mathcal{M}, h, t \models \psi) \end{array} \tag{14}$$

$$\begin{array}{c} \mathcal{M}, i \models Ob\ \varphi \\ \text{iff } (\forall i' \in \mathcal{I},\ \text{if } i\mathbb{O}i' \text{ then } \mathcal{M}, i' \models \varphi) \end{array} \tag{15}$$

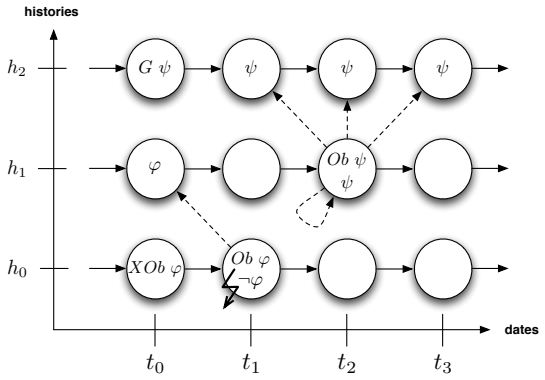

Figure 1: Example DLP model.

Thus, a deontic obligation on a formula at a given date of a given history means that this formula is true at all instants accessible from the given one via $\mathbb{O}$. Fig. 1 shows a portion of an example DLP model. In this illustration, histories are arranged horizontally, indexed by dates (histories extend both in the past and in the future). Places figure instants, arrows represent the $\rightsquigarrow$ temporal accessibility relation and dotted arrows represent the $\mathbb{O}$ deontic accessibility relation (which is only partially represented). The broken arrow represents the local violation, at (h_0, t_1), of an SDL obligation.

One could note that $\mathbb{O}$ is not a relation among histories: a deontically acceptable alternative is an instant in a history, and not the history as a whole. This gives a better granularity to the notion of norm violation. Eqs. (16)-(18) give the semantics of the temporal operators: $\varphi\ \mathcal{U}\ \psi$ is true at a given date of a given history if and only if, in the same history, ψ occurs at some point in the (strict) future, and φ is true from the next instant (tomorrow) until then. $H\varphi$ is true at an instant if and only if φ has been true at every past instant in the same history, and $X^{-1}\varphi$ is true at an instant if and only if φ was true at the previous instant in the same history. It is then straightforward to check that the semantics of temporal abbreviations is defined as usual.

$$\mathcal{M}, h, t \models \varphi\ \mathcal{U}\ \psi \text{ iff } \exists t' \in \mathcal{T}, \begin{cases} t < t', \\ \mathcal{M}, h, t' \models \psi \\ \quad \forall t'' \in \mathcal{T}, \text{ if } t < t'' < t' \\ \qquad \text{then } \mathcal{M}, h, t' \models \varphi \end{cases} \tag{16}$$

$$\mathcal{M}, h, t \models H\varphi \text{ iff } \forall t' \in \mathcal{T}, \quad \text{if } t' < t \text{ then } \mathcal{M}, h, t' \models \varphi \tag{17}$$

$$\mathcal{M}, h, t \models X^{-1}\varphi \text{ iff } \forall t' \in \mathcal{T}, \quad \text{if } t' \rightsquigarrow t \text{ then } \mathcal{M}, h, t' \models \varphi \tag{18}$$

4 Deontico-temporal operators in DLP

It must now be examined how obligations with deadlines and maintained interdictions can be defined in DLP. The first objective is to build an operator $Ob(\varphi, \delta)$ intended to mean: "it is obligated that φ becomes true strictly before δ". Later on its deontic counterpart, the maintained obligation operator $For(\varphi, \delta)$ (meaning "it is forbidden that φ becomes true from now until δ"), will also be discussed. The operators $violOb(\varphi, \delta)$ and $violFor(\varphi, \delta)$ are defined as the respective violations of these deontic operators.

4.1 Requirements

In order to represent deadlines, a special kind of propositions is added to the language, the *dated propositions*, similar in nature to Åqvist's "systematic frame constants" [2] or to the propositions used by Dignum [13] and Demolombe [11] in order to model deadlines. The additional predicate $dated(\delta)$ means that δ is a dated proposition. They are defined as propositions occuring once and only once in the timeflow, as stated by Eq. (19).

$$dated(\delta) \stackrel{def}{=} \begin{cases} \delta \wedge G\neg\delta \wedge H\neg\delta \\ \vee F(\delta \wedge G\neg\delta \wedge H\neg\delta) \\ \vee P(\delta \wedge G\neg\delta \wedge H\neg\delta) \end{cases} \tag{19}$$

Eight *requirements* are now proposed and discussed. These are the criteria that an operator for obligations with deadlines must match in order to properly represent the notion in DLP logic. Some of them are of very formal nature, designed for the operator to interact well with the native temporal and deontic modalities. Other ones are essentially philosophical and could be considered as choice points. Most of these requirements have been designed to formally describe the notion of obligation with deadline and its implications.

1. *The deadline must be a dated proposition*: for a deadline to bear a meaning, it must be defined in a clear and distinct fashion. It must exist and be unique in the timeflow. Dated propositions have been defined exactly for this purpose.
2. *The deadline must be in the future*: it is obvious for a human that an obligation with a present or past deadline does not leave any choice to schedule actions. Its fulfilment is independent from an agent's actions or capabilities at the time of utterance of the obligation: either the obligation is already fulfilled because of a past action or situation, or there is nothing the agent can do about it.
3. *Obligations on $\top$ must be tautologies*: this principle is introduced for the sake of coherence with the immediate deontic obligation, for which $Ob\ \top$ is a theorem. In the same way, $Ob(\top, \delta)$ should be a theorem.
4. *Obligations on $\bot$ should be antilogies*: for the same reasons, $\neg\, Ob(\bot, \delta)$ should be a theorem.
5. *Violated obligations should be dropped after the deadline*: this choice is rather philosophical in nature, and subject to debate (although it is a direct consequence of point 2). It seems reasonable to consider that if the agent would or could not cope with the obligation on time, then it is too late and the violation must be simply accepted and the obligation should not be derived anymore. Indeed, the original obligation with deadline says nothing about what should happen after the deadline. This principle is not an opposition to the existence of eventual contrary-to-duty norms attached to obligations with deadlines. This requirement allows an agent not to consider indefinitely obligations that it will never be able to fulfill and that might be obsolete for reasons linked to their meaning in the real world.
6. *Violations should be punctual*: in some formalisms [13, 11], an agent can be considered in a "state" of violation as soon as an obligation with deadline is not fulfilled on time. Given the temporal expressiveness allowed by DLP, and notably the ability to reason on past actions, it seems sufficient, more convenient and more correct to think of violations as events, rather than as states. It is indeed possible to pinpoint the date at which a given agent has failed to fulfill a given norm, which gives a better expressiveness than a flat violation state. Philosophically speaking, it also avoids the final characterization and stigmatization of an agent as a violator, thus opening the door to richer cognitive methods for trust management and recovery, like forgiveness mechanisms for instance [28]. Once again, this point is specific to the chosen formalism and subject to debate.
7. *The propagation principle must be respected*: Eq. (20) formally defines this requirement, graphically illustrated by Fig. 2. It says that the obligation with deadline must be maintained from instant to instant, until the obligation is respected or the deadline is reached.

$$Ob(\varphi, \delta) \wedge \neg(\varphi \vee \delta) \wedge \neg X\delta \rightarrow XOb(\varphi, \delta) \tag{20}$$

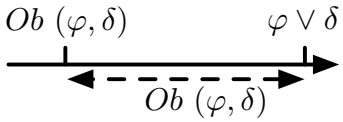

Figure 2: The propagation principle.

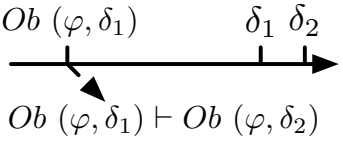

Figure 3: The monotony principle.

8. *The monotony principle must be respected*: Eq. (21) defines monotony as a purely logical necessity, by saying that if there is an obligation with a deadline, then an obligation with a further deadline can be derived. For instance, if an agent must delete some data before July 1st, then it is logical to consider that it must also delete it before August 1st and before any later date. This principle is illustrated by Fig. 3.

$$Ob(\varphi, \delta) \wedge F(\delta \wedge F^{-}\delta') \rightarrow Ob(\varphi, \delta') \tag{21}$$

The first six *requirements* have already been pointed out as choice points and discussed by Dignum *et al.* [13]. However, due to their different goals and formalisms, their opinion differs on some aspects. For instance, since they do not reason on past, they define violations as states, not as events. Furthermore, they reject obligations on $\top$ since they are not coherent with the STIT operator they use. On some other points (like the nature of deadlines), they do not make a clear choice and leave the final decision to the reader, depending on her needs.

Propagation and monotony requirements have been introduced by Brunel [4], and they have been adapted to the more strict interpretation of the notion of deadline discussed in the current work.

4.2 Evaluating some existing proposals

The three majors proposals, made by Dignum *et al.* [13], Demolombe *et al.* [11] and Brunel *et al.* [4], have been presented in section 2.3.2 and are now to be evaluated as candidates for modelling the notion of obligations with deadlines in DLP. Before being compared and matched against the eight requirements previously discussed, they need to be adapted to the SDL $\times$ LTL formalism. The adaptation of the existing operators will be noted, respectively, Ob^a, Ob^b and Ob^c. Since these operators appear in a different formalism in the original papers, as it has been said already, they may have slightly different properties in a different context. It is therefore necessary to pay attention to the translation step, in order to convey in priority the meaning of the original operators, rather than their form, while introducing as few extensions as possible in the language. Hence, it is the translation of the original operators (and their adaptation to DLP) which is evaluated, and not the operators themselves, although their main characteristics remain in the DLP formalism.

As already introduced, the operator Ob^a by Dignum is defined jointly with the notion of violation, and does not rely on a real deontic logic [13]. Its adaptation to the DLP formalism is

represented by Eq. (22)

$$\begin{aligned} Ob^a(\varphi,\delta) \stackrel{def}{=} & (\neg \text{ viol}(\varphi)\ \mathcal{U}^-\ \delta) \\ & \wedge\Big(\big[((\neg\varphi \wedge \neg\delta)\ \mathcal{U}^-\ \varphi) \\ & \quad \wedge(\neg\varphi\ \mathcal{U}^-\ (\varphi \wedge G^- \neg \text{ viol}(\varphi)))\big] \\ & \vee\big[((\neg\varphi \wedge \neg\delta)\ \mathcal{U}^-\ \delta) \\ & \quad \wedge(\neg\delta\ \mathcal{U}^-\ (\delta \wedge G^-\ \text{viol}(\varphi)))\big]\Big) \end{aligned} \tag{22}$$

The translation in natural language of this complex definition is:

> There is no violation before the deadline δ, and one of the following options is true:
>
> - Either a sees to it that φ becomes true before δ, and then a will never be in violation in the future;
> - Or a does not see to it that φ becomes true until δ, and then a is in a state of violation from δ and forever afterwards.

Here, an obligation with a deadline is therefore characterized by the absence of violation before the deadline and by the maintaining of a violation, or its negation, after the deadline, depending on whether the obligation has been fulfilled on time or not. Because of this strictly temporal definition, such obligations can be derived whenever they seem to be respected. This is quite an undesirale feature for building norm-directed agents. For instance, let us imagine an agent running a process `processID` involving personal information (action represented in DLP by *perform*(`agent`, `processID`), which will be the target formula φ here), before an arbitrary date δ, and let us consider an instant in the past of this process. Nothing indicates that the agent is in violation anytime, therefore $\neg$ viol$(\varphi)\ \mathcal{U}^-\ \delta$ is verified. It can also be checked that neither *perform*(`agent`, `processID`) nor δ occur before the process, so $((\neg\varphi \wedge \neg\delta)\ \mathcal{U}^-\ \varphi)$ is also derivable. At last, since nothing allows to conclude that the agent will ever be in a state of violation, the derivability of $(\neg\varphi\ \mathcal{U}^-\ (\varphi \wedge G^- \neg \text{ viol}(\varphi)))$ can easily be accepted. With all these elements, it is possible, by definition of the operator, to derive $Ob^a(\varphi,\delta)$ anytime before the process takes place, with is totally counter-intuitive: nothing seems to obligate the agent to perform this process.

Furthermore, the operator Ob^a is not monotonic at all: there are several cases where an obligation with a further deadline cannot be derived. Namely, if the obligation is fulfilled and then another obligation on the same formula is violated after the first deadline, or if the obligation is violated and then φ becomes true after the deadline, then it is impossible to derive obligations with further deadlines. It is also worth noting that deadlines with a value of $\top$ can be defined, resulting in an immediate obligation.

The Ob^b operator, adapted from Demolombe [11], translates better in a deontic and temporal logic, as shown in Eq. (23).

$$Ob^b(\varphi,\delta) \stackrel{def}{=} Ob(F^-(\varphi \wedge F\delta))\ \mathcal{U}^-\ (\varphi \vee \delta) \tag{23}$$

An obligation with a deadline is now the maintaining (until the deadline or the fulfilment of the obligation) of an obligation of anteriority between the target formula and the deadline. Propagation is here built-in in the operator, in the form of immediate obligations allowing us to derive obligations with deadlines. Regarding monotony, if an obligation $Ob^b(\varphi,\delta)$ is fulfilled on time, then it is possible to check that $Ob^b(\varphi,\delta')$ is true for any future δ'. However, if it is violated, it is not possible to derive an obligation with a further deadline. This property will be called **semi-monotony** and is illustrated in Fig. 4. It is interesting to see that this property has a nice side-effect: it prevents the derivation of multiple violations for the same initial obligation, whereas it still allows monotony if the obligation is fulfilled.

The Ob^c operator is based on two different dated obligations [4], expressed in a formalism already close to DLP. The first one, $Ob^{c'}$, defined in Eq. (24), is used when the obligation is initially activated, and does not allow propagation (although it is monotonic). It is an immediate obligation that φ become true before the deadline. Brunel, unlike precedent authors, reason on

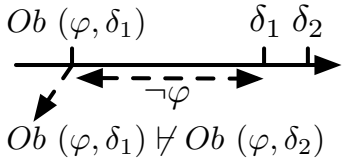

Figure 4: The semi-monotony principle.

Table 1: Evaluation of existing operators

Requirement	1	2	3	4	5	6	7	8
Ob^a		□	□		■		■	
Ob^b			■	□	■	□	■	□
Ob^c	■	■		■	■	□	■	

delays of k time steps ($k \in \mathbb{N}$) rather than on deadlines. The formalism will then be temporarily adapted, and δ_k will refer to a deadline associated with a delay of k time steps.

$$Ob^{c'}(\varphi, \delta_k) \stackrel{def}{=} X^k \delta_k \wedge Ob\left(\bigvee_{i=0}^{k-1} X^i \varphi\right) \tag{24}$$

The final operator, designed in Eq. (25), has the following meaning: an obligation with a deadline is active at the present time if there was in the past an obligation (of type $Ob^{c'}$, and not fulfilled yet) on φ with a deadline not reached already, and also that there was no obligation on the same formula with a shorter deadline.

$$Ob^c(\varphi, \delta_k) \stackrel{def}{=} \begin{cases} X^k \delta_k \\ \exists k' \in \mathbb{N}, X^{-k'} Ob^{c'}(\varphi, \delta_k) \\ \qquad \wedge \bigwedge_{i=1}^{k'} X^{-i} \neg\varphi \\ \nexists k'' < k, X^{k''} \delta_{k''} \\ \qquad \wedge X^{-k'} Ob^{c'}(\varphi, \delta_{k''}) \end{cases} \tag{25}$$

This more complex operator deals efficiently with propagation and is able to express subtle nuances of dated obligations. Yet, it has several noticeable drawbacks in the application context. First, it introduces new quantifiers on delays (when Ob^b relies on existing temporal quantifiers, for instance). For this reason it is not directly expressible in a normal modal language. Plus, the operator does not respect the monotony principle, although it deals properly with other requirements, by reasoning on delays rather than on deadlines.

Table 1 summarizes how the three operators match the eight requirements. A full square means that the requirement is natively matched by the candidate operator. An empty square means that the requirement is only partially satisfied, or under some conditions, whereas void means that the requirement is not met at all.

4.3 An operator adapted to DLP

Even though Ob^c satisfies more requirements than the other operators (and could thus be considered as the best candidate), both Ob^a and Ob^c imply too significant extensions to DLP logic to be directly usable (namely, the non-deontic nature of Ob^a and the explicit quantification on durations in Ob^c). The operator Ob^b seems to be a better trade-off, being much more compatible with the chosen formalism. It can be improved by the use of dated propositions, so that it

satisfies the requirements in a better way. For the reason already given, semi-monotony, rather than strict monotony, is deemed suitable for the final operator, which is now defined by Eq. (26). Its punctual violation, adapted from the persistent one proposed by Demolombe, is defined by Eq. (27).

$$Ob(\varphi,\delta) \stackrel{def}{=} \left\{ \begin{array}{l} dated(\delta) \\ F\delta \\ \quad Ob\ (F^-(\varphi \wedge F\delta)) \\ \qquad\quad \mathcal{U}^-\ (\varphi \vee \delta) \end{array} \right. \tag{26}$$

$$violOb(\varphi,\delta) \stackrel{def}{=} \left\{ \begin{array}{l} dated(\delta) \\ \delta \\ P\left(\begin{array}{l} Ob\ (\varphi,\delta) \\ \qquad \wedge\neg\varphi\ \mathcal{U}^-\delta \end{array} \right) \end{array} \right. \tag{27}$$

It is now easy to show that this adapted operator satisfies the eight requirements (with monotony replaced by semi-monotony) defined in section 4.1.

Theorem 4.1 *The operator $Ob(\varphi,\delta)$ and its associated violation $violOb(\varphi,\delta)$ have the following properties:*

1. *Deadlines are dated propositions (they occur once and only once in the timeflow);*
2. *Deadlines are always in the strict future;*
3. *$Ob(\top,\delta)$ can be derived for each properly defined deadline δ;*
4. *$\neg Ob(\bot,\delta)$ is a theorem;*
5. *Obligations are dropped after the deadline;*
6. *Violations are punctual:*
$$violOb(\varphi,\delta) \rightarrow \left\{ \begin{array}{l} G\neg violOb(\varphi,\delta) \\ H\neg violOb(\varphi,\delta) \end{array} \right.$$
7. *Propagation principle is respected;*
8. *Semi-monotony is respected:*
$$\left. \begin{array}{l} Ob(\varphi,\delta) \wedge dated(\delta') \\ \wedge\ F(\delta \wedge F^-\delta') \\ \wedge\ \neg(\neg\varphi\ \mathcal{U}^-\ \delta) \end{array} \right\} \rightarrow Ob(\varphi,\delta')$$

Proof elements follow:

- *Properties 1 and 2:* trivial, by definition of the operator.
- *Property 3:* $dated(\delta)$ and $F\delta$ being derivable, $Ob(\top,\delta)$ reduces to $Ob(F^-(\top \wedge F\delta))\ \mathcal{U}^-\ (\top)$, then to $\top\ \mathcal{U}^-\ \top$, which is $\top$.
- *Property 4:* $Ob(\bot,\delta)$ implies $Ob(F^-(\bot))\ \mathcal{U}^-\ (\delta)$. $F^-(\bot)$ implies $F(\bot)$, then $\bot$. By the D axiom of SDL obligations, the original expression then boils down to $\bot\ \mathcal{U}^-\ \delta$ and then to $\bot$.
- *Property 5:* $Ob(\varphi,\delta) \rightarrow F\delta$, so $F^-Ob(\varphi,\delta) \rightarrow F\delta$, then $\neg G^-\neg Ob(\varphi,\delta) \rightarrow F\delta$. $F\delta$ implies $\neg\delta$, so $(\neg G^-\neg Ob(\varphi,\delta)) \rightarrow \neg\delta$. *Modus tollens* then leads to the result.
- *Property 6:* $violOb(\varphi,\delta)$ implies δ, so it can be true only once in the timeflow.
- *Property 7:* the antecedent of the proposition allows to derive $Xdated(\delta)$ and $XF\delta$ (through $\neg X\delta \wedge F\delta$). $\neg(\varphi \vee \delta)$ tells that the deadline is not reached yet, so the whole $\mathcal{U}^-$ expression will be true at next time step. By conjunction of these points, $XOb(\varphi,\delta)$ is obtained.
- *Property 8:* the antecedent of the proposition gives $dated(\delta')$ and $F\delta'$. $\neg(\neg\varphi\ \mathcal{U}^-\ \delta)$ implies that φ will be true before δ, so before δ'. Therefore, $Ob(F^-(\varphi \wedge F\delta))\ \mathcal{U}^-\ (\varphi \vee \delta)$ implies $Ob(F^-(\varphi \wedge F\delta))\ \mathcal{U}^-\ (\varphi \vee \delta')$. On the other hand, $(\varphi \wedge F\delta)$ implies $(\varphi \wedge F\delta')$, therefore by the normal character of modalities $Ob(F^-(\varphi \wedge F\delta'))\ \mathcal{U}^-\ (\varphi \vee \delta')$. The antecedent therefore allows to derive $Ob(\varphi,\delta')$.

□

4.4 Maintained obligations

The second notion needed to translate the example norm in the proposed way is the concept of interdiction maintained over time. This notion has attracted less attention in the deontic logic community, and to the best of our knowledge, no previous work could be found of the subject. Indeed, the choice points here seam to be less significant and maybe trivial to address. It is much easier to define an operator with good properties for maintained interdictions than for obligations with deadlines. It is however interesting (at least for the sake of coherence) to observe what the eight requirements become in the case of the maintained obligation. Here, "deadline" is to be understood as the end of the period of interdiction, and $For(\varphi, \delta)$ is the interdiction on φ maintained until δ.

1. *The deadline must be a dated proposition*: the period of interdiction is defined by a start date (the date at which the norm is first issued) and an end date (the deadline). Like in the case of the obligation with deadline, this period must be sufficiently well-defined for an agent to safely reason about it.
2. *The deadline must be in the future*: When a maintained obligation is true at a given instant, the beginning of the period can eventually be somewhere in the past, but the end of it must be really clearly defined and situated in the strict future. The justification for it is the same as for obligations with deadlines.
3. *Maintained obligations on $\bot$ must be tautologies*: a maintained interdiction is essentially similar to an SDL interdiction, which is always true when applied to a logical contradiction.
4. *Maintained obligations on $\top$ must be antilogies*: again, this requirement is introduced for consistency with the immediate SDL interdiction modality.
5. *Violated interdictions must be maintained until the deadline*: this choice can seem (at first sight) slightly inconsistent with the corresponding criterion for obligations with deadlines, because of the inherent differences between the notions: when an obligation with deadline is violated, it means that the deadline has been reached and therefore the norm is not applicable anymore. In the case of maintained interdictions, however, a first violation does not mean that the interdiction is dropped, it remains active until the deadline and can possibly be violated again.
6. *Violations are defined only at the dates when the norm is violated*: as it has just been suggested, there is a need to reason on possibly multiple violations of the same norm. The choice of having punctual violations here becomes of great interest. Indeed, although an obligation with deadline can be violated only once (*i.e.* at the deadline itself, when it is observed that the obligation has not been fulfilled), a maintained interdiction can be violated several times in the period[1]. Depending on the application, the perceived severity of the violation can even increase with its frequency. Once again, the ability to reason on past brings the ability to pinpoint violations, characterize them, count them and remember them.
7. *The propagation principle must be respected*: in the context of maintained interdictions, it can be defined as "If there is an interdiction on φ maintained until δ, and if δ is not true at the next time step, then the interdiction is propagated to the next time step". Eq. (28) formally characterizes this principle. It is worth noting that the fifth requirement is merely a consequence of the propagation principle, and could have been omitted.

$$For(\varphi, \delta) \wedge \neg X\delta \rightarrow XFor(\varphi, \delta) \tag{28}$$

8. *The monotony principle must be respected*: this principle now says that if there is an interdiction on φ maintained until δ, then there is an interdiction on φ maintained until

[1] In dense timeflows, it can even be violated an infinite number of times, thus resulting in a violation state limited in time. This violation state is however distinct in semantics from Dignum's violation state [13] which is intended to remember a past violation. These considerations do not apply in the non-dense timeflow chosen here.

any sooner deadline. This principle is formalized in Eq. (29).

$$For(\varphi,\delta) \wedge dated(\delta') \wedge F(\delta' \wedge F\delta) \rightarrow For(\varphi,\delta') \tag{29}$$

It is easy to build the corresponding operator and its associated violation, as defined by Eqs. (30) and (31).

$$For(\varphi,\delta) \stackrel{def}{=} \begin{cases} dated(\delta) \\ F\delta \\ (For\ \varphi)\ \mathcal{U}^-\ \delta \end{cases} \tag{30}$$

$$violFor(\varphi,\delta) \stackrel{def}{=} \begin{cases} dated(\delta) \\ F\delta \\ \varphi \\ P^-For(\varphi,\delta) \end{cases} \tag{31}$$

It can now be shown that this operator fulfills the eight requirements.

Theorem 4.2 *The operator $For(\varphi,\delta)$ and its associated violation $violFor(\varphi,\delta)$ have the following properties:*

1. *Deadlines are dated propositions;*
$$For(\varphi,\delta) \rightarrow dated(\delta)$$
2. *Deadlines are always in the strict future;*
$$For(\varphi,\delta) \rightarrow F\delta$$
3. *$\neg For(\top,\delta)$ is a theorem;*
4. *$For(\bot,\delta)$ can be derived for each properly defined deadline δ;*
5. *Maintained interdiction, even violated, are kept until the deadline;*
$$P^-For(\varphi,\delta) \wedge F\delta \rightarrow For(\varphi,\delta)$$
6. *Violations are defined only at the instants when the norm is violated;*
$$violFor(\varphi,\delta) \rightarrow \varphi$$
7. *Propagation principle is respected;*
$$For(\varphi,\delta) \wedge \neg X\delta \rightarrow XFor(\varphi,\delta)$$
8. *Monotony is respected;*
$$For(\varphi,\delta) \wedge dated(\delta') \wedge F(\delta' \wedge F\delta) \rightarrow For(\varphi,\delta')$$

Proof elements follow:

- *Properties 1 and 2:* trivial, by definition of the operator.
- *Property 3:* $\neg For\top$ is a theorem, therefore $For\top\ \mathcal{U}^-\ \delta$ is derivable only if δ is derivable. Since $F\delta$ implies $\neg\delta$ for a dated proposition, the desired contradiction is obtained.
- *Property 4:* $For\bot$ is a theorem and $F\delta$ is derivable by hypothesis, therefore $(For\bot\ \mathcal{U}^-\ \delta)$ is derivable.
- *Property 5:* trivial, per the semantics of $\mathcal{U}^-$.
- *Property 6:* trivial, per the definition of the violation.
- *Property 7:* From $(For\ \varphi)\ \mathcal{U}^-\ \delta$, $F\delta$ and $\neg X\delta$, $X((For\ \varphi)\ \mathcal{U}^-\ \delta)$ can be obtained through the semantics of $\mathcal{U}^-$. From $F\delta$ and $\neg X\delta$ one gets $XF\delta$, and $X\,dated(\delta)$ comes from the fact that $dated(\delta)$ is persistent over time, by construction.
- *Property 8:* $dated(\delta')$ and $F\delta'$ come from the hypotheses. From $(For\ \varphi)\ \mathcal{U}^-\ \delta$, $(For\ \varphi)\ \mathcal{U}^-\ \delta'$ can easily be obtained through the semantics of $\mathcal{U}^-$.

□

4.5 Synthesis

The Deontic Logic for Privacy now constitutes a formal framework dedicated to privacy-based reasoning. It embeds deontic and temporal notions in a simple and generic way and provides well-designed specialized operators in order to deal with deontico-temporal notions which are prominent in privacy-related regulations.

5 Application to privacy norms

Now that two suitable operators are available for obligations with deadlines and maintained obligations, it is possible to examine how DLP logic can be used to reason upon personal data protection regulations and to idenfity what its limits are.

5.1 Example norms

5.1.1 Information dimension

Let us start by translating in DLP logic the interpretations that have been made earlier of the fictional norm presented as an example:

> *Users must be informed at least one week in advance of the nature of any process involving their personal information.*

As it has been explained already, it can be considered from two points of view by an artificial agent: a teleologic point of view (based on an obligation to inform the user before a deadline situated one week before the process date) and a constraining point of view (using a maintained interdiction forbidding the process to be run in the week to come). The regulation will therefore be translated into (a conjunction of) two partially redundant DLP formulae, representing the two sides of the problem. Other more intuitive or more common example norms may have resulted in simpler interpretation cases.

Eq. (32) formally says that if there is an identified type (ActionType) for a given process (ProcessID), and this process uses given data (DataID), and if δ is in the future and one week before the scheduled run of the process, and the owner (User) has not been informed yet, then there is an obligation to inform her of the nature of the process (speech act represented by the *informActionType* predicate) before δ. Eq. (33) says that if the owner of these data has not been informed so far about the process, and the date δ in situated one week in the future, then there is a maintained interdiction to perform the process until δ.

$$\left.\begin{array}{l} \mathit{actionType}(\text{ProcessID}, \text{ActionType}) \\ \mathit{dated}(\delta) \\ F(\delta \wedge X^{7*24-1}\mathit{perform}(\texttt{self}, \text{ProcessID})) \\ H\neg \mathit{informActionType}(\texttt{self}, \text{User}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ \mathit{owner}(\text{ProcessID}, \text{DataID}, \text{User}) \end{array}\right\} \\ \quad \rightarrow Ob(\mathit{informActionType}(\texttt{self}, \\ \qquad \text{User}, \text{ProcessID}, \text{ActionType}), \delta) \tag{32}$$

$$\left.\begin{array}{l} \mathit{actionType}(\text{ProcessID}, \text{ActionType}) \\ H\neg \mathit{informActionType}(\texttt{self}, \text{User}, \\ \quad \text{ProcessID}, \text{ActionType}) \\ \mathit{owner}(\text{ProcessID}, \text{DataID}, \text{User}) \\ \mathit{dated}(\delta) \wedge X^{7*24}\delta \end{array}\right\} \\ \quad \rightarrow For(\mathit{perform}(\texttt{self}, \text{ProcessID}), \delta) \tag{33}$$

If the norm expressed in Eq. (32) is activated (*i.e.* if its antecedent is verified), then the agent knows that it has to inform the user before a given and well-known date δ. The propagation principle (and basically the structure of the operator, ensuring the utterance of an immediate

obligation at each step of the period) provides a direct way to make the agent feel "concerned" already by this obligation, thus triggering planning mechanisms instead of letting it postpone the interpretation and fulfilment of the norm.

On the other hand, if the norm represented by Eq. (33) is activated, then the agent sees that any occurence of ProcessID before δ would result in a violation. A privacy-aware agent willing to comply with enacted regulations would then refrain from performing this action (by generating a BDI desire about it for instance, or by postponing any undertaken plan or intention involving ProcessID). The monotony principle here makes it easier to develop reasoning means to effectively derive short-term and even immediate interdictions on the basis of a maintained interdiction.

5.1.2 User consent dimension

It is also possible to exhibit privacy-related regulations that do not mention any explicit deadline in their phrasing, but that will be translated in DLP by the means of obligations with deadlines and/or maintained obligations. Typically, the following fictional regulation is a simple and very common norm about personal data protection, attached to the "user consent" fundamental dimension:

> *A prior and written agreement from the customer is mandatory for any transaction involving her bank account number to take place.*

It does not formally contain a deadline, except that it imposes a precedence relation between the consent and the transaction. Therefore, it can be translated as an obligation to obtain consent before a given deadline, set at the date of a planned future transaction, as in Eq. (34).

$$\begin{array}{l} date(\delta) \wedge \\ F \left\{ \begin{array}{l} \delta \\ perform(\texttt{self}, \text{ProcessID}) \\ owner(\text{ProcessID}, \text{Account}, \text{Customer}) \\ dataType(\text{ProcessID}, \text{Account}, \\ \quad \texttt{Ecom.Payment.Bank.} \\ \quad\quad \texttt{PayerAccountNumber}) \end{array} \right. \\ \rightarrow Ob(consent(\text{Customer}, \texttt{self}, \\ \quad \text{ProcessID}), \delta) \end{array} \tag{34}$$

It means that if δ is a dated proposition situated in some future in which a transaction "ProcessID" is undertaken, involving some data "Account" belonging to "Customer" and of a type corresponding to a bank account number, then there is currently an obligation to obtain this customer's consent with a deadline δ. One can note here the use of the bank account datatype taken from the ECML P3P schema, proposed by the W3C for electronic commerce purposes [30].

In a similar way as previously, this norm can also be seen as an immediate interdiction to perform such a transaction in the case a consent has not been obtained yet, as expressed in Eq. (35). Since there is no delay specified, it is not a maintained interdiction but a standard one.

$$\left. \begin{array}{l} owner(\text{ProcessID}, \text{Account}, \text{Customer}) \\ dataType(\text{ProcessID}, \text{Account}, \\ \quad \texttt{Ecom.Payment.Bank.} \\ \quad \texttt{PayerAccountNumber}) \\ H \neg consent(\text{Customer}, \texttt{self}, \text{ProcessID})) \\ \rightarrow For(perform(\texttt{self}, \text{ProcessID})) \end{array} \right\} \tag{35}$$

5.1.3 Data update dimension

Let us now consider the following rule, corresponding to the "data update" regulation dimension:

> *It is mandatory to provide customers with a contact address allowing them to make update requests on their profile data.*

It corresponds to a usual legal requirement that users should usually have a means to know what kind of information is retained about them, to ask for a rectification if they are erroneous and possibly to request a deletion [23, 26, 27]. However, phrased as it is, it needs an interpretation. One can reasonably choose to understand it as an obligation to act as stated, before any collection of personal data. It then presents the same structure as the previous sentence. It will then be translated into both an obligation to provide contact information (if not done yet) before the date of a planned data collection, as in Eq. (36), and an immediate interdiction to request data if the information has not been given, as in Eq. (37).

$$\left.\begin{array}{l} date(\delta) \\ F(\delta \wedge request(\texttt{self}, \text{Customer}, \text{ProcessID}, \\ \quad \text{DatatType}, \text{DataID})) \\ H^{-} \neg informContact(\texttt{self}, \text{Customer}, \\ \quad \text{ProcessID}, \text{Contact}) \end{array}\right\} \\ \rightarrow Ob(informContact(\texttt{self}, \\ \quad \text{Customer}, \text{ProcessID}, \text{Contact}), \delta) \tag{36}$$

$$H^{-} \neg informContact(\texttt{self}, \text{Customer}, \\ \quad \text{ProcessID}, \text{Contact}) \\ \rightarrow For(request(\texttt{self}, \text{Customer}, \text{ProcessID}, \\ \quad \text{DatatType}, \text{DataID})) \tag{37}$$

In the previous formulae, *request* represents a service agent requesting personal data of a given datatype from a user agent and *informContact* represents the speech act by which the service agent provides a contact address to a user agent, in relation to an identified process ID.

5.2 Limitations

DLP logic seems to be reasonably able to represent common kinds of regulations regarding the management of personal data. However, some kinds of norms, less likely to be encountered in this context, are not yet taken into account in the present version of the model.

First, it is not natively possible to represent periodic or recurrent obligations, that is a single obligation associated with a number of deadlines. With the operators designed here, each deadline must be attached to an individual obligation. This limitation could eventually be by-passed by translating such norms into meta rules (still DLP formulae) generating finite or even infinite sets of standard obligations with deadlines in a recursive manner. This solution, which could be covered by a simpler syntactic abbreviation, is probably not the most elegant way of dealing with such notions, though.

Whereas it has been pointed out that the model by Cardoso and Oliveira [5] did not fit the semantics of obligations with deadlines chosen in this article, it contains several interesting proposals that could enrich a generic normative framework, if not a privacy-dedicated one. The first interesting point is the possibility for deadline extensions, made possible by the distinction between obligation violations (that is not fulfilling the obligation at all) and deadline violations (fulfilling the obligation too late). It seems possible to build higher levels abbreviations capturing such notions, since the underlying mechanisms are not essentially richer than the concepts manipulated here. Deadline violation is what is simply called here violation, and it is straightforward to design a predicate defining obligation violation as the absence of fulfillment even after the deadline. Yet, the notion of deadline extension, used in the case of a deadline violation, would necessary be a simulated one, attached to a brand new obligation, since the initial one would not be available anymore. Indeed, the possibility of keeping an obligation after the deadline has been rejected in the present work, but it is still possible to design customized consequences of failure of any kind, thanks to the expressiveness of DLP logic.

The notion of liveline, proposed in the same article, does not seem to be easily embedded in the obligation with deadline operator designed here. Livelines are defined as a counterpart to deadlines: if an agent has an obligation on φ with a deadline δ and a liveline δ', then it must

perform φ not later than δ and not sooner than δ'. This would need the design of a completely new operator, which the LTL modalities would most probably allow.

Finally, it seems at first sight that directed obligations (that is obligations holding an obligor and an obligee as part of their definition [22]) cannot be expressed in DLP. It is necessary to precise that the logic presented here is only a fragment of DLP, in which a few elements have been omitted for the sake of clarity. Namely, the full DLP language [17, chap. 4] indexes operator modalities with a "normative authority" (the obligor) and an agent (the obligee). It is therefore possible to enrich the reasoning based on such notions.

6 Conclusion and future work

The Deontic Logic for Privacy (DLP) has been proposed here as a language, based on temporal and deontic logic, aimed at representing and reasoning on privacy-related norms. The need for specific operators dealing with obligations with deadlines and with maintained obligations has been pointed out. Eight requirements have been defined fo each operator to bear the right intuitive meaning in the DLP formalism. Three operators, already proposed to model the notion of obligation with deadline, have been translated into DLP logic and examined. None of them could meet simultaneously all of the eight requirements. The most suited of these operators has been slightly adapted in order to satisfy the requirements, and a new operator has been designed for dealing with maintained obligations. Finally, these operators have been used in the translation of an example regulation in DLP logic.

It is interesting to point out that the translation of privacy regulations from natural language to formal expressions apparently cannot be done without a human interpretation phase, and not only because of the natural language processing problems. In the context of artificial agents, it would then be necessary to develop designing tools constraining the form of the regulations defined by a human user. With such pre-constrained regulations, a privacy-preserving framework could rely on a few well-known norm forms, which could be automatically translated in sets of DLP formulae by following the principles used in the example norms previously discussed.

The deontic and temporal features presented here have been integrated to the full version of the DLP language, which makes use of deontic modalities differentiated both by normative authority and by normative agent. The operators for obligations with deadlines and for maintained interdictions have been made the core component of an improved version of the deontic conflict detection and resolution presented in a previous communication [18]. Their propagation and (semi-) monotony principles have proven very useful for the automatic derivation of additional obligations and interdictions, thus allowing fast conflict spotting. The implementation of this conflict resolution engine (able to spot and resolve conflicts between an arbitrary number of deontico-temporal formulae) is in Prolog, making extensive use of the CLPFD finite domain constraint solver [6]. Indeed, deadlines in deontico-temporal norms can be internally represented with constraint domains instead of explicit dates or delays, thanks to the propagation and monotony principles of the operators.

Future work involves a formal evaluation of the characteristics and performance of the implementation, as well as a comparison with the formal model of DLP logic and with other possible implementations. DLP engines will then be integrated in the cognitive layers of "privacy-aware" software agents [17, chap. 3]. These agents are meant to be used as personal assistants or as automatic service providers in Internet-based (or other distributed) applications, in order to help human users to deal properly and more safely with their private information. Foreseen usage scenarios involve interface and profile management in medical contexts, ambient intelligence and e-commerce, three application domains showing prominent and different concerns for privacy.

This research has been partially supported by the Web Intelligence project of the Rhône-Alpes region ISLE cluster.

References

[1] T. Ågotnes, W. van der Hoek, J. A. Rodríguez-Aguilar, C. Sierra, and M. Wooldridge. On the logic of normative systems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, Hyderabad, India, January 2007.

[2] L. Åqvist. Combinations of tense and deontic modalities. In A. Lomuscio and D. Nute, editors, *Proceedings of the 7th International Workshop on Deontic Logic in Computer Science (DEON 2004)*, volume 3065 of *LNCS*, pages 3–28, Madeira, Portugal, May 2004. Springer.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[4] J. Brunel, J.-P. Bodeveix, and M. Filali. A state/event temporal deontic logic. In *Eighth International Workshop on Deontic Logic in Computer Science (DEON'06)*, number 4048 in LNCS, 2006.

[5] H. L. Cardoso and E. Oliveira. Directed deadline obligations in agent-based business contracts. In *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems (COIN'09)*, Budapest, Hungary, 2009.

[6] M. Carlsson, G. Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In *Programming Languages: Implementations, Logics and Programs*, 1997.

[7] B. F. Chellas. *Modal Logic, an Introduction*. Cambridge University Press, 1980.

[8] L. Cholvy and F. Cuppens. Analyzing consistency of security policies. In *Proceedings of the 18th IEEE Symposium on Research in Security and Privacy*, Oakland, CA, USA, May 1997.

[9] L. Cholvy and F. Cuppens. Reasoning about norms provided by conflicting regulations. In H. Prakken and P. McNamara, editors, *Norms, Logics and Information Systems: New Studies in Deontic Logic and Computer Science*, pages 247–264, Amsterdam, the Netherlands, 1998. IOS Press.

[10] L. Cholvy and S. Roussel. Raisonner avec une réglementation incomplète : le cas d'une politique d'échange d'informations. In *16e congrès francophone AFRIF-AFIA sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA'08)*, Amiens, France, January 2008. AFRIF-AFIA.

[11] R. Demolombe. Formalisation de l'obligation de faire avec délais. In *Troisièmes journées francophones des modèles formels de l'interaction (MFI'05)*, Caen, France, May 2005.

[12] Y. Deswarte and C. Aguilar-Melchor. Current and future privacy enhancing technologies for the internet. *Annals of Telecommunications*, 61(3-4):399–417, 2006.

[13] F. Dignum, J. Broersen, V. Dignum, and J.-J. Meyer. Meeting the deadline: Why, when and how. In M. G. Hinchey, J. L. Rash, W. Truszkowski, and C. Rouff, editors, *Third International Workshop on Formal Approaches to Agent-Based Systems (FAABS'04)*, number 3228 in LNCS, pages 30–40, Greenbelt, MD, USA, April 2004. Springer Verlag.

[14] J. F. Horty and N. D. Belnap, Jr. The deliberative stit: a study of action, omission, ability, and obligation. *Journal of Philosophical Logic*, 24:583–644, 1995.

[15] ISO/IEC. Information technology - security techniques - evaluation criteria for IT security, part 2: Security functional requirements. Technical Report 15408-2, International Organization for Standardization, 1999.

[16] R. Ortalo. Using deontic logic for security policy specification. LAAS report 96380, LAAS-CNRS, Toulouse, France, October 1996.

[17] G. Piolle. *Agents utilisateurs pour la protection des données personnelles : modélisation logique et outils informatiques*. PhD thesis, Université Joseph Fourier - Grenoble I, Grenoble, France, 6 2009.

[18] G. Piolle and Y. Demazeau. Une logique pour raisonner sur la protection des données personnelles. In *16e congrès francophone AFRIF-AFIA sur la Reconnaissance de Formes et l'Intelligence Artificielle (RFIA'08)*, Amiens, France, January 2008. AFRIF-AFIA.

[19] G. Piolle and Y. Demazeau. Representing privacy regulations with deontico-temporal operators. *Web Intelligence and Agent Systems: an International Journal (WIAS)*, 9(3):209–226, July 2011.

[20] G. Piolle, Y. Demazeau, and J. Caelen. Privacy management in user-centred multi-agent systems. In G. O'Hare, M. O'Grady, O. Dikenelli, and A. Ricci, editors, *Proceedings of the 7th Annual International Workshop on Engineering Societies in the Agents World (ESAW'06)*, volume 4457/2007 of *LNCS*, pages 354–367, Dublin, Ireland, September 2006. Springer Verlag.

[21] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57, Providence, Rhode Island, USA, 1977. IEEE Computer Society Press.

[22] Y. U. Ryu. Relativized deontic modalities for contractual obligations in formal business communication. In *Proceedings of the 30th Hawaii International Conference on System Sciences (HICCS'97)*, page 485, Washington, DC, USA, 1997. IEEE Computer Society.

[23] République française. Loi numéro 78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés. In *Journal Officiel de la République Française*, January 1978.

[24] République française. Loi numéro 2004-801 du 6 août 2004 relative à la protection des personnes physiques à l'égard des traitements de données à caractère personnel. In *Journal Officiel de la République Française*, August 2004.

[25] K. Segerberg. Some Meinong/Chisholm thesis. *Logic, Law, Morality. A festrichft in honor of Lennart Åqvist*, 51:67–77, 2003. Uppsala Philosophical Studies.

[26] The European Parliament and the Council. Directive 1995/46/EC of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. In European Union, editor, *Official Journal of the European Communities*, October 1995.

[27] The European Parliament and the Council. Directive 2002/58/EC of the european parliament and of the council of 12 july 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector. In European Union, editor, *Official Journal of the European Communities*, July 2002.

[28] A. Vasalou, J. Pitt, and G. Piolle. Proposing new ways of resolving online conflicts: an intelligent facilitation of forgiveness in CMC. In *Proceedings of the Workshop Reinventing trust, collaboration and compliance in social systems (Reinvent06) at CHI 2006*, Montreal, Quebec, April 2006.

[29] G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.

[30] World Wide Web Consortium. Using P3P for e-commerce, 1999. `http://www.w3.org/TR/P3P-for-ecommerce`.

[31] World Wide Web Consortium. Platform for Privacy Preferences specification 1.1., 2006. `http://www.w3.org/P3P/`.