Towards an Enforceable GDPR Specification

François Hublet, Alexander Kvamme, and Srđan Krstić

Departement of Computer Science, ETH Zürich
{francois.hublet,srdan.krstic}@inf.ethz.ch

Abstract. While Privacy by Design (PbD) is prescribed by modern privacy regulations such as the EU's GDPR, achieving PbD in real software systems is a notoriously difficult task. One emerging technique to realize PbD is *Runtime enforcement* (RE), in which an *enforcer*, loaded with a specification of a system's privacy requirements, observes the actions performed by the system and instructs it to perform actions that will ensure compliance with these requirements at all times.
To be able to use RE techniques for PbD, privacy regulations first need to be translated into an enforceable specification. In this paper, we report on our ongoing work in formalizing the GDPR. We first present a set of requirements and an iterative methodology for creating enforceable formal specifications of legal provisions. Then, we report on a preliminary case study in which we used our methodology to derive an enforceable specification of part of the GDPR. Our case study suggests that our methodology can be effectively used to develop accurate enforceable specifications.

Keywords: Privacy by design · Runtime enforcement · Formalization.

1 Introduction

Ensuring compliance of software systems with privacy regulations is a notoriously challenging task. With mounting evidence that current regulations such as the EU's General Data Protection Regulation (GPDR) are poorly implemented by a majority of data controllers [8], the capacity of ex-post enforcement and fines to bring about widespread compliance appears limited. This calls for a methodology that relies on principled approaches to *design* and *certify* software systems to adhere to regulations, in the spirit of Privacy by Design (PbD) [10].

Runtime enforcement[1] (RE) is one such approach. In RE, a formal specification is input to a software system, called an *enforcer*, that observes the actions performed by a System under Scrutiny (SuS). In addition to observing the SuS's actions, the enforcer can also send commands to the SuS, typically instructing the SuS to prevent or cause certain actions. By observing the SuS's actions and responding with appropriate commands, the enforcer seeks to ensure that the behavior of the SuS adheres to its specification at all times. A specification for which such an enforcer exists is called *enforceable*.

[1] Note that the term 'enforcement' has a different meaning in legal and computer science contexts. In the former, it typically refers to the actions taken by state officials to end or penalize a past or ongoing violation of the law. In the latter, '(runtime) enforcement' refers to a process of ensuring compliance with a policy at any time, *preventing* policy violations rather than compensating for them. In the following, 'enforcement' shall be understood in this latter sense.

In the context of privacy, typically regulations (e.g., the GDPR), must be enforced. Therefore, to use RE for enforcing privacy in software systems one must *formalize* privacy regulations into *enforceable specifications*. This entails *understanding* the regulations, making them *precise* in the context of specific software system actions, making them *formal* (i.e., readable by an enforcer), and, finally, *enforceable*. In this paper, we report on our ongoing work in formalizing GDPR that is both close to the letter of the law and enforceable. Our work can serve as a common ground for both computer scientists and legal experts to collaborate in achieving PbD.

This paper is organized as follows. First, we review existing work on formalizing privacy regulations, especially the GDPR (Section 2) and briefly introduce RE (Section 3). We then present four requirements and an iterative methodology for obtaining an enforceable specification of regulations in general (Section 4). Our methodology is applicable both to regulations for which no specification exists and those for which an existing *non-enforceable* (or inaccurate) specification exists. Finally, we report on a case study in which we applied our methodology to convert a series of GDPR provisions formalized in DAPRECO [29] into enforceable specifications (Section 5). In the course of our case study, we discovered a number of inaccuracies in DAPRECO that can be, at least in part, linked to Robaldo et al.'s choice of a formalism and methodology. We discuss how our methodology can be used to alleviate the risk of such inaccuracies while additionally providing out-of-the-box support for RE in real systems. In conclusion, we reflect on the limitations of our approach and open questions (Section 6).

2 Related work

While some of the first formal specifications of legal provisions [30] date back to the 1980s and there is a related active field at the interplay of logic, linguistics, computer science, and law (see, e.g., [16]), formal specifications of legal provisions amenable to formal reasoning are still relatively rare. Recently, tax codes were formalized using special-purpose programming languages [23, 20]. The complexity of tax codes lies in complex conditional structures depending on a large number of variables and conditions, requiring the use of special (typically, default [1]) logics. However, tax codes lack most of the temporal and system-specific dimensions that are intrinsic to privacy regulations.

The runtime verification community [3] has seen an increasing number of efforts to detect (or *monitor*) violations of complex policies in large-scale systems. These efforts have brought to the fore a number of challenges in terms of both policy engineering and the design of the interaction between the SuS and monitors [12]. One of the most comprehensive studies to date is Basin et al.'s monitoring of several millions of log entries from the "Internet Computer," a distributed Web3 platform, against large temporal-logic policies with over 1,000 binary operators [6]. The latter study required very careful policy engineering, as temporal logic policies had to be extracted from an informal description of protocol properties provided by engineers. To the best of our knowledge, no such large-scale study has been conducted yet with *legal* provisions.

In the last two decades, smaller portions of privacy laws were formalized, often based on ad-hoc interpretations tailored to specific application domains. In an early work, Lam et al. formalized part of the US Health Insurance Portability and Accountability Act

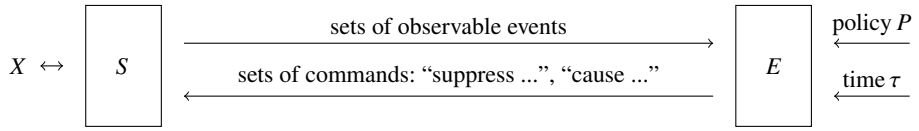

Fig. 1. System model for runtime enforcement

(HIPAA) using stratified Prolog [22]. Arfelt et al. [2] formalized data subject rights and monitored them in industrial logs. Hublet et al. [18] showed how to enforce a similar core of GDPR provisions in web applications and extended Arfelt et al.'s study to support runtime enforcement [19]. Palmirani and Governatori used LegalRuleML [26] and the PrOnto ontology [27] to model GDPR provisions [25]. Bonatti et al. formalized selected GDPR-related constraints on business processes using the Web Ontology Language OWL2 [9]. On another line of work, a large number of publications (see the survey [15]) have focused on defining GDPR-compatible *policy languages* that can be used to describe legal flows of information depending on, e.g., user consent, and help users define their own preferences. Finally, Torre et al. [31] have translated GDPR concepts into full-fledged ontologies, which, however, lack an associated specification of legal provisions.

Only a handful of previous works have attempted to formalize entire privacy regulations in a more literal way, capturing legal concepts into an appropriate ontology and converting individual paragraphs of the original legal document into logical formulae. In these respects, two series of works stand out. The first one is DeYoung et al.'s extensive formal specification of those parts of the Gramm-Leach-Bliley Act (GLBA) and HIPAA [14, 13] that defined "operational requirements" of systems. DeYoung et al. use an extension of Least Fixed Point logic (LFP) to encode these requirements into a well-documented set of LFP formulae. The second series of work is Robaldo, Bartolini et al.'s DAPRECO knowledge base [29, 28], which provides the most comprehensive formal specification of the GDPR to date. The authors provide an ontology and a set of over 900 formulae that they claim to cover most of the GDPR except articles 51–76. A small portion of this specification has been validated through interdisciplinary collaboration with legal experts [4, 5]. None of these two series of work consider enforcement.

3 Runtime enforcement

In this section, we briefly overview runtime enforcement (RE). We start from a system model that we assume a system under scrutiny (SuS) implements. This model is a simplified version of a model introduced in our previous work [19]. We then present the specification language that our enforcer supports.

System model. Figure 1 shows how an enforcer E supervises a SuS S, which interacts with an environment X that E cannot control. E must ensure that the sequence of actions executed by S complies with a given policy P. An example of such a policy is

P_1 = "any use of a data item in a system has been preceded by user consent."

E also has access to the current time τ from some reference clock.

$e(t_1, \ldots, t_k)$	Event e occurs in the log with arguments $t_1, \ldots, t_k$
NOT φ	φ is not fulfilled
φ AND ψ	Both φ and ψ are fulfilled
φ OR ψ	Either φ or ψ is fulfilled
φ IMPLIES ψ	If φ is fulfilled, then ψ is fulfilled
EXISTS x. φ	There exists some value of x such that φ is fulfilled
FORALL x. φ	For any value of x, the formula φ is fulfilled
ONCE φ	The formula φ is fulfilled at some time-point in the past or present
HISTORICALLY φ	The formula φ is fulfilled at all time-points in the past or present
EVENTUALLY φ	The formula φ is fulfilled at some time-point in the present or future
ALWAYS φ	The formula φ is fulfilled at all time-points in the present or future

Fig. 2. Syntax and semantics of selected MFOTL operators.

During its execution, S reports *events* to E. An event is $e(a_1, \ldots, a_k)$ where e is called the 'event name' and $a_1, \ldots, a_k$ are 'arguments.' An example of such an event is

$$\texttt{uses}(\text{“}\texttt{website.com}\text{”}, \text{“}\texttt{birthday}\text{”}, \text{“}\texttt{Alice}\text{”}, \text{“}\texttt{advertisement}\text{”}),$$

meaning "the application hosted at `website.com` is using user `Alice`'s birthday for advertisement purposes." In general, these events are taken from some appropriate ontology and encode specific actions of S. Events (and the corresponding actions) s that S reports to E are called *observable*. E records events in a *log* together with the current time. The log is a sequence $\sigma = ((\tau_1, D_1), \ldots, (\tau_k, D_k), \ldots)$, where $\tau_1, \ldots, \tau_k$ are timestamps and $D_1, \ldots, D_k$ are sets of events. Each pair (τ_i, D_i) is called a *time-point*.

E can then emit *commands* instructing S to cause and suppress some of its actions. An action that can be caused is represented by a *causable event*; an action that can be suppressed is represented by a *suppressable* event. Which events are causable and suppressable depends on the system's functionality and the way it implements its interface with the enforcer. For instance, if S is processing data, *data usage* can typically be made *suppressable* by ensuring that S always asks E for permission *before* processing a data item. On the other hand, *erasure of data* can typically be made *causable* provided that S provides the enforcer E with an interface to execute the erasure of specific data items (e.g., from its database). In our model [19], we allow E to send commands to S both in response to the logging of sets of events ("reactively") and on its own initiative ("proactively").

Specification. The specification needed to perform enforcement in the above model consists of three components:

1. An ontology listing the available event names and the types of their arguments and describing their high-level meaning in terms of system actions.
2. A mapping of every event name to a set of attributes among 'observable', 'causable,' and 'suppressable,' which we call *capabilities*.
3. A policy P describing the set of all logs that are deemed legal.

In the past, various logics have been used to describe policies to be enforced at runtime. In this paper, we will focus on Metric First-Order Temporal Logic (MFOTL) [11, 7], for which state-of-the-art enforcement tools exists [17, 19].

A fragment of the syntax of MFOTL and its intuitive semantics are described in Figure 2. For a formal account, we refer the reader to previous work [7, 19]. As an example, the policy P_1 can be expressed as MFOTL formula φ_1

$$\texttt{ALWAYS}\ (\ \texttt{FORALL}\ app, data, user, purpose.\ \mathsf{uses}(app, data, user, purpose)\ \texttt{IMPLIES}\ \texttt{ONCE}(\mathsf{consent}(user, app, purpose)))$$

which reads "at all times, for any app, $data$, $user$, and $purpose$, whenever the application app uses the data $data$ of user $user$ for purpose $purpose$, then, at an earlier time-point in the log, $user$ has given consent to app to use their data for $purpose$."

Properties of policies. Given an ontology and assumptions on the causability and suppressability or events, we say that a policy is

enforceable if there exists an enforcer ensuring that SuS is policy-compliant at all times;
transparently enforceable if there exists an enforcer ensuring that SuS is policy-compliant at all times *and* it causes or suppresses events *only* when not causing or not suppressing them would lead to the policy being violated.

Assuming that data usage is suppressable, the policy P_1 above is enforceable: the enforcer can simply prevent data usage. By considering an enforcer that prevents data usage *only* when it is not preceded by consent, we see that P_1 is also transparently enforceable, as usage is suppressed only when this is necessary for compliance. Our tool WhyEnf [19] can decide whether an MFOTL formula is from some (transparently) enforceable fragment, and it is able to enforce all formulae it can identify as enforceable.

4 Requirements and methodology

We describe list four key requirements for enforceable specifications of legal provisions and present our iterative methodology to develop such a specification.

Requirements. What properties should a specification of legal provisions amenable to RE have? The first requirement is straightforward; it comes in two variants:

R1.1) The policy must be enforceable.
R1.2) The policy must be transparently enforceable.

Some policies are known to be enforceable, but not transparently enforceable [19]. Hence, R1.2 might not always be achievable when R1.1 is.

Whether a policy is enforceable depends on its structure and on the causability and suppressability of the events it uses. In order for a specification to be usable for RE, we want such assumptions to be realistic. That is, we want to be able to effectively *instrument* the SuS in such a way that the enforcer can cause or suppress the actions that causable and suppressable events respectively represent. Moreover, we want SuS to accurately report all observable events to the enforcer. We call this property *instrumentability*. In practice, instrumentability can be assessed either generically for a family of systems

fulfilling certain (formal or informal) requirements, or specifically for a single SuS. The latter can be most convincingly demonstrated by actually implementing the code that reports relevant events and executes the enforcer's commands. We obtain

R2) The system must be instrumentable based on the stated capabilities, i.e.:
  - R2a) The system must report all actions that give rise to observable events.
  - R2b) The system must support causation of all actions that give rise to causable events.
  - R2c) The system must support suppression of all actions that give rise to suppressable events.

Another key requirement for a specification of legal provisions is that is should capture an acceptable interpretation of the law. This is a qualitative requirement that can only be fully assessed through an interdisciplinary effort with legal experts. How literal and strict the interpretation should be—e.g., whether the policy should strictly follow the logical structure of the law or simply state some stronger condition that guarantees legality—will depend on the specific application. In the following, we refer to this as the *faithfulness* requirement. To obtain a faithful specification, not only does the policy need to be carefully formulated, but the ontology must also be designed in such a way that the semantics of each event aligns with legal definitions and are sufficiently precise to avoid an incorrect instrumentation. We thus get

R3) The ontology and policy must faithfully capture the underlying legal provisions, i.e., all of the following conditions must hold:
  - R3a) The policy must capture an acceptable interpretation of the law,
  - R3b) Every event in the ontology must come together with a clear documentation of its semantics,
  - R3c) This semantics must be sufficiently precise so that the instrumentation of each action (logging and possibly causation/suppression of events) will be consistent with legal definitions and the expectations of legal experts.

For consistency with previous work [14, 29], we will focus on a 'literal' approach that closely follows the logical structure of existing legislation. Hence, R3a will be understood as requiring a close logical correspondence with the law.

As formal specifications of legal provisions should serve as bridges between the legal and technical communities, it is reasonable to require that these specifications be understandable by both legal and technical experts. Yet, accessibility to an audience without *any* exposure to formal reasoning is likely to be infeasible, even when using a user-friendly surface representation [4]. Fortunately, legal experts involved in the assessment of formal specifications of software systems are generally more technically experienced than their peers. They may additionally be offered some form of training that should, however, not be excessively long or difficult. Those trained experts are those we will refer to as 'legal experts' in the rest of this paper.

Ensuring the *understandability* of formal specifications is not only important for legal experts. Even technical experts' understanding of a policy may be influenced when the complexity of the policy grows. As the the representation of the same policy in various formal frameworks can be very different, the understandability of a specification

depends on the formalism used. But even within a given formalism, certain design decisions such as, e.g., ensuring that the policy is written in a reasonably concise way, can significantly improve understandability.

R4) The policy must be understandable by both technical and legal experts, i.e., all of the following conditions must hold:
  - R4a) The logical formalism must be accessible to both technical and legal audiences assuming a moderate amount of training,
  - R4b) The policy must be expressed in a clear and concise way,
  - R4c) The complexity (size, logical structure) of the policy must not grow beyond what experts can comprehend.

Only R1 can be automatically checked using a tool such as WhyEnf [19], while R2 can be demonstrated by either concrete implementation or high-level reasoning and R3–4 are qualitative. Figure 3 summarizes the parts of the specification that each requirement concerns and which tool or expert(s) can assert its fulfillment.

	Involves...			
	Ontology	Capabilities	Policy	Assessment by...
R1	✓	✓	✓	Tool (e.g. [19])
R2a	✓	✓		Technical expert
R2b	✓	✓		Technical expert
R2c	✓	✓		Technical expert
R3a	✓		✓	Legal expert
R3b	✓			Legal expert
R3c	✓			Legal and technical expert
R4a			✓	Legal and technical expert
R4b			✓	Legal and technical expert
R4c			✓	Legal and technical expert

Fig. 3. Summary of requirements

Methodology. We now present an iterative methodology to develop a specification fulfilling R1–4. The flowchart of our methodology is shown in Figure 4.

Our methodology resembles pair programming [23] in which a technical expert (TE, e.g., a programmer or a logician) collaborates with a legal expert (LE) to develop an enforceable specification of a given set of legal provisions. The TE and LE start by selecting a formalism or a series of formalisms that fulfills R4a. Using several formalisms that can be mechanically converted into each other can be meaningful when, e.g., the LE is more comfortable with a textual representation of the formula while the TE wishes a more mathematical representation [4]. If previous work has already developed a partial specification of the law, then this existing specification can be converted into a specification in the new formalism and serve as a starting point.

After a possible conversion step, the TE and LE start by jointly drafting an ontology representing the law's main concepts and checking its compatibility with requirements R3b–c (clarity, precision). Next, they draft a policy representing the law's provisions

and check its compatibility with requirements R3a and R4b–c (acceptable interpretation of the law, understandability). Then, the TE sets the capacities of each event described in the ontology to fulfill R2a–c. Finally, they use a tool such as WhyEnf [19] to check whether the policy is enforceable (i.e., R1). If this is the case, then the TE and LE have successfully derived a specification that fulfills all of the stated requirements. Otherwise, the following can be attempted to recover enforceability: (1) extend the capabilities (if possible) to observe, cause, or suppress more actions (2) modify the policy (while preserving R3a and R4b–c) to make it enforceable (3) modify the ontology and/or the policy (while preserving all requirements) to make the policy enforceable. We suggest to try these in the order of the least required changes to the specification.

5 Case study

In this section, we report on a preliminary case study in developing an enforceable GDPR specification using the methodology in Section 4. This case study was conducted as part of one of the authors' Master's thesis [21], using the existing specification from DAPRECO [29] as a starting point. While this preliminary case study did *not* involve legal experts, we claim that it already demonstrates the potential of our approach to improve over the state of the art. Namely, our case study allowed us to discover and correct a number of inaccuracies in the existing specification, and generate a specification of single GDPR provisions that is directly amenable to formal reasoning using the WhyEnf enforcement tool [19].

Formalism selection and conversion. We pick MFOTL as a formalism for specifying our policies as previous work [2, 18] has shown that MFOTL provides the necessary expressivity for encoding (parts of) existing laws, and enforcement tools for MFOTL are available [17, 19].

We first developed a series of algorithms to convert the Reified I/O Logic specification by Robaldo et al. [29] into an equivalent[2] MFOTL specification [21], and applied it to the DAPRECO knowledge base to obtain an MFOTL formula for each of 966 Reified I/O Logic formulae of DAPRECO. Conversion involved transforming I/O rules into classical implications to be enforced; identifying the temporal patterns encoded through reification; and rewriting these patterns using MFOTL operators. We also implemented a number of static checks aimed at identifying incorrect syntactic patterns, such as unused variables. The DAPRECO ontology, which to the best of our knowledge remained documented, was then extracted from the resulting MFOTL formulae.

Deriving an enforceable formal specification of Art. 7(1). In the following, we describe in detail how we derived an enforceable specification for Article 7(1) GDPR and corrected inaccuracies in an initial specification derived from DAPRECO. We chose this article as it served as an example to demonstrate DAPRECO's validation methodology [4, 5], which involved legal experts. Article 7(1) GDPR states

[2] In the course of developing this conversion algorithm, it became apparent that some of the logical formalisms used by Robaldo et al., especially reification, lacked formal semantics. A semantics thus had to be reconstructed based on textual descriptions from previous work.

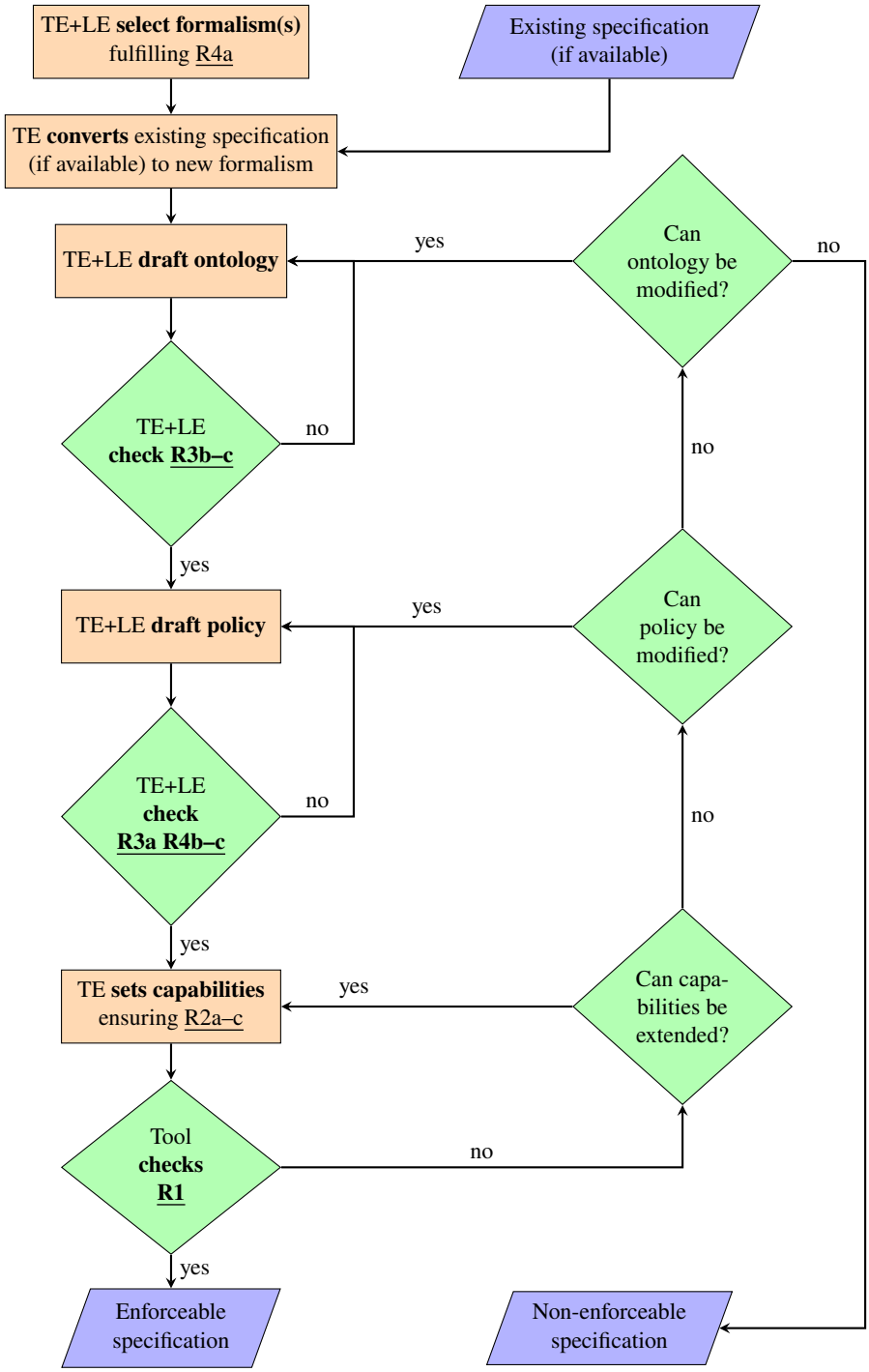

Fig. 4. Flowchart of our methodology

> Where processing is based on consent, the controller shall be able to demonstrate that the data subject has consented to processing of his or her personal data.

After an automated conversion to MFOTL, the DAPRECO version of this provision reads

$$
\begin{aligned}
\varphi^{D}_{7(1)} = \texttt{ALWAYS}\ (&\\
&\texttt{EXISTS}\ ep, eau, edp, w, z, x, epu, c.\ (\\
&\quad \mathsf{PersonalDataProcessing}(ep, x, z)\ \texttt{AND}\ \mathsf{isBasedOn}(ep, epu)\\
&\quad \texttt{AND}\ \mathsf{GiveConsent}(ehc, w, c)\ \texttt{AND}\ \mathsf{AuthorizedBy}(eau, epu, c)\\
&\quad \texttt{AND}\ \mathsf{nominates}(edp, y, x)\ \texttt{AND}\ \mathsf{PersonalData}(z, w)\ \texttt{AND}\ \mathsf{Purpose}(epu))\\
&\texttt{IMPLIES}\ (\texttt{EXISTS}\ ea, ed.\ (\mathsf{AbleTo}(ea, y, ed)\ \texttt{AND}\ \mathsf{Demonstrate}(ed, y, ehc)))).
\end{aligned}
$$

English equivalent: "*Whenever personal data of an individual is processed by a processor nominated by a controller, and the processing is based on a purpose, and the individual's consent authorizes the purpose (*sic*!), then the controller must be able to demonstrate that that consent has been given.* "

The corresponding (undocumented) ontology contains all the events appearing in the formula. First, we need to reconstruct the precise semantics of the events it contains (R3b–c). For space reasons, we do not spell out the semantics of all of these events here, but focus on GiveConsent, inBasedOn, and AuthorizedBy. From its use in the specification, the semantics of $\mathsf{GiveConsent}(ehc, w, c)$ can be reconstructed as "the individual w gives consent c; the individual's action of giving consent is given the unique identifier ehc." The meaning of $\mathsf{isBasedOn}(ep, epu)$ and $\mathsf{AuthorizedBy}(eau, epu, c)$ is much less clear: the use of the variable epu suggest the semantics "the processing task ep is *based on the purpose* epu" and "[usage for] purpose epu is authorized by consent c; this authorization relation is given the unique identifier eau." However, the law only refers to "*based on consent*," not to "based on a purpose" and "authorized by consent," and the consent action appears to be missing the identity of the controller. While Robaldo et al. may have chosen this approach for consistency with the earlier PrOnto ontology [27], which has isBasedOn, we see this part of the ontology as diverging from the GDPR's concepts. We suggest to instead use two predicates $\mathsf{isBasedOn}(ep, ehc)$ and $\mathsf{GiveConsent}(ehc, w, x, epu)$ with the semantics "data processing ep is based on consent action ehc" and "by consent action ehc, user w grands consent to x to use her data for purpose epu," and to add an event $\mathsf{hasPurpose}(ep, epu)$ meaning "the processing ep has the purpose epu." The fact that epu is always a purpose makes an additional $\mathsf{Purpose}(epu)$ event redundant in this case. After modifying the ontology, our formula is

$$
\begin{aligned}
\varphi^{2}_{7(1)} = \texttt{ALWAYS}\ (&\\
&\texttt{EXISTS}\ ep, eau, edp, w, z, x, epu, c.\ (\\
&\quad \mathsf{PersonalDataProcessing}(ep, x, z)\ \texttt{AND}\ \mathsf{isBasedOn}(ep, ehc)\\
&\quad \texttt{AND}\ \mathsf{GiveConsent}(ehc, w, x, epu)\ \texttt{AND}\ \mathsf{hasPurpose}(ep, epu)\\
&\quad \texttt{AND}\ \mathsf{nominates}(edp, y, x)\ \texttt{AND}\ \mathsf{PersonalData}(z, w))\\
&\texttt{IMPLIES}\ (\texttt{EXISTS}\ ea, ed.\ (\mathsf{AbleTo}(ea, y, ed)\ \texttt{AND}\ \mathsf{Demonstrate}(ed, y, ehc)))).
\end{aligned}
$$

English equivalent: "*Whenever personal data of an individual is processed by a processor nominated by a controller, and the processing is based on consent given by the user for the purpose of the current processing, then the controller must be able to demonstrate that that consent has been given.* "

Now, we perform the checks related to the policy. Does $\varphi^{D}_{7(1)}$ accurately reflect the letter of the law (R3a)? We claim that the temporal structure of the formula is not correct, as the specification states that an obligation to demonstrate consent only exists when the data processing (PersonalDataProcessing) and consent (GiveConsent) are *simultaneous*. This obviously goes against standard legal interpretations (see, e.g., [24, Art. 7, Rn. 6]). Hence, we must correct $\varphi^{2}_{7(1)}$ by replacing $\mathsf{GiveConsent}(ehc, w, x, epu)$ by $\texttt{ONCE}\ \mathsf{GiveConsent}(ehc, w, x, epu)$. We call the resulting formula $\varphi^{3}_{7(1)}$ Note that this problem is independent of the mismatch between GDPR concepts and the ontology that we corrected in the previous step. Is the formula clear (R4b–c)? From a mathematical point of view, one potential source of confusion is that two variables (ehc, y) are implicitly universally quantified [29]. After adding explicit quantifiers (i.e., $\texttt{ALWAYS}\ \texttt{FORALL}\ ehc, y \ldots$), we obtain

$$
\begin{aligned}
\varphi^{4}_{7(1)} = \texttt{ALWAYS}\ \texttt{FORALL}\ & ehc, y.\ (\\
& \texttt{EXISTS}\ ep, eau, edp, w, z, x, epu, c.\ (\\
& \quad \mathsf{PersonalDataProcessing}(ep, x, z)\ \texttt{AND}\ \mathsf{isBasedOn}(ep, ehc)\\
& \quad \texttt{AND}\ (\texttt{ONCE}\ \mathsf{GiveConsent}(ehc, w, x, epu))\ \texttt{AND}\ \mathsf{hasPurpose}(ep, epu)\\
& \quad \texttt{AND}\ \mathsf{nominates}(edp, y, x)\ \texttt{AND}\ \mathsf{PersonalData}(z, w))\\
& \texttt{IMPLIES}\ (\texttt{EXISTS}\ ea, ed.\ (\mathsf{AbleTo}(ea, y, ed)\ \texttt{AND}\ \mathsf{Demonstrate}(ed, y, ehc)))).
\end{aligned}
$$

English equivalent: "*Whenever personal data of an individual is processed by a processor nominated by a controller, and the processing is based on consent previously given by the user for the purpose of the current processing, then the controller must be able to demonstrate that that consent has been given.* "

We claim that $\varphi^{4}_{7(1)}$ is mathematically clear and concise, and that its complexity (a few lines) is reasonable. Making this formula understandable to non-technical experts does, however, require some accurate higher-level representation, e.g., of the form proposed by Robaldo et al. [29].

Next, we set the capabilities, assuming, as suggested in Section 3, that data processing in our SuS can be subjected to prior approval by the enforcer. In this case, PersonalDataProcessing can be made observable and suppressable. The other relevant actions can all be made observable by ensuring that the SuS can report the legal basis and purposes of processing, received consent, processor-controller delegation relationships, and its capacity to demonstrate consent to the enforcer. This satisfies R2a–c. Finally, we use WhyEnf [19] to check the enforceability of $\varphi^{4}_{7(1)}$ when PersonalDataProcessing is suppressable. The check succeeds, guaranteeing transparent enforceability (R1.2): to ensure compliance with $\varphi^{4}_{7(1)}$ at all times, it necessary and sufficient to prevent personal data processing when the system can not demonstrate that it has previously obtained consent.

Discussion. We performed a similar effort for all provisions of Articles 5, 6, 7, 11, 12, 13, and 17 covered by DAPRECO. We discovered many issues related to unclear semantics of predicates, inaccurate modeling of time, unused variables, and lack of clarity [21].

The fact that even the specification that Robaldo et al. validated with legal experts [29] proved to be incorrect after a more precise analysis shows the benefits of our methodology and its anchoring in formal methods, especially RE. The criteria of "accuracy, completeness, correctness, consistency, and conciseness" [29] against which Robaldo et al. asked their experts to evaluate the policy failed to prevent the inaccuracies we identified, likely because (1) the semantics of the ontology was never documented and evaluated together with the policy, and (2) the representation of time in Reified I/O logic was too complex to be properly used even by the experts themselves.

Our conversion to MFOTL made many modeling mistakes apparent. Hence, another take-away is that MFOTL is a promising specification language to formalize legal provisions, allowing for expressive first-order modeling with a transparent encoding of time.

6 Conclusion and open questions

In this paper, we have presented four requirements and a new methodology for developing enforceable formal specifications of privacy laws. We have then reported on a preliminary case study focusing on selected provisions of the General Data Protection Regulation (GDPR). In the course of our case study, we have identified several inaccuracies in an existing GDPR specification. Overall, our case study demonstrates the benefits of our methodology and suggests Metric First-Order Temporal Logic (MFOTL) as a promising language for formalizing laws.

In order to obtain an enforceable, state-of-the-art specification that can serve as a reference for both computer scientists and legal experts, our preliminary case study needs to be extended in two directions: by involving several technical and legal experts, and by extending the scope of the formalization effort to a larger fragment of the GDPR—ideally covering all provisions that regulate computer systems' behavior. To allow for efficient collaboration between legal and technical experts in this context, relying a more user-friendly specification language with some temporal features (rather than just MFOTL) is indispensable. We plan to develop such a language as part of our future work.

Another open question is how to *refine* general specifications of the 'literal' kind we discussed into simpler, more concrete specifications tailored to guarantee the compliance of specific systems. While a rich theory of refinement exists within formal methods, we are not aware of any previous work that would apply these techniques in a legal context.

Last but not least, our preliminary results and the inaccuracies we identified in previous work call for the development of more systematic methodology for the joint assessment of legal compliance by both legal and technical experts. In general, technical experts alone cannot assert the compliance of a system with legal requirements. But neither can legal experts if their understanding of the system's behavior is not informed by trustworthy technical experts' knowledge. As a result, only a joint assessment of compliance is ever possible. This raises fundamental theoretical and practical questions than can be interesting to both the technical and legal communities.

References

1. Antoniou, G.: A tutorial on default logics. ACM Computing Surveys (CSUR) **31**(4), 337–359 (1999)
2. Arfelt, E., Basin, D., Debois, S.: Monitoring the GDPR. In: Computer Security–ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part I 24. pp. 681–699. Springer (2019)
3. Bartocci, E., Falcone, Y.: Lectures on runtime verification. Springer (2018)
4. Bartolini, C., Lenzini, G., Santos, C.: A legal validation of a formal representation of GDPR articles. In: Proceedings of the 2nd JURIX Workshop on Technologies for Regulatory Compliance (Terecom) (2018)
5. Bartolini, C., Lenzini, G., Santos, C.: An agile approach to validate a formal representation of the GDPR. In: New Frontiers in Artificial Intelligence: JSAI-isAI 2018 Workshops, JURISIN, AI-Biz, SKL, LENLS, IDAA, Yokohama, Japan, November 12–14, 2018, Revised Selected Papers. pp. 160–176. Springer (2019)
6. Basin, D., Dietiker, D.S., Krstić, S., Pignolet, Y., Raszyk, M., Schneider, J., Ter-Gabrielyan, A.: Monitoring the Internet Computer. In: Chechik, M., Katoen, J., Leucker, M. (eds.) 25th International Symposium on Formal Methods (FM). LNCS, vol. 14000, pp. 383–402. Springer (2023)
7. Basin, D., Klaedtke, F., Müller, S., Zălinescu, E.: Monitoring metric first-order temporal properties. Journal of the ACM (JACM) **62**(2), 1–45 (2015)
8. Bollinger, D., Kubicek, K., Cotrini, C., Basin, D.: Automating cookie consent and GDPR violation detection. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 2893–2910. USENIX Association, Boston, MA (Aug 2022), https://www.usenix.org/conference/usenixsecurity22/presentation/bollinger
9. Bonatti, P.A., Kirrane, S., Petrova, I.M., Sauro, L.: Machine understandable policies and GDPR compliance checking. KI-Künstliche Intelligenz **34**, 303–315 (2020)
10. Cavoukian, A.: Privacy by design. Tech. rep., Office of the Information and Privacy Commissioner (2009)
11. Chomicki, J.: Efficient checking of temporal integrity constraints using bounded history encoding. ACM Transactions on Database Systems (TODS) **20**(2), 149–186 (1995)
12. Colombo, C., Pace, G.J.: Industrial experiences with runtime verification of financial transaction systems: Lessons learnt and standing challenges. In: Lectures on Runtime Verification – Introductory and Advanced Topics, LNCS, vol. 10457, pp. 211–232. Springer (2018)
13. DeYoung, H., Garg, D., Jia, L., Kaynar, D., Datta, A.: Experiences in the logical specification of the hipaa and glba privacy laws. In: Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society. pp. 73–82 (2010)
14. DeYoung, H., Garg, D., Kaynar, D., Datta, A.: Logical specification of the glba and hipaa privacy laws. CMU, Pittsburgh, PA, USA, Tech. Rep. CMU-CyLab-10-007 (2010)
15. Esteves, B., Rodríguez-Doncel, V.: Analysis of ontologies and policy languages to represent information flows in GDPR. Semantic Web **13**(Preprint), 1–35 (2022)
16. Gabbay, D.M., Canivez, P., Rahman, S., Thiercelin, A.: Approaches to legal rationality, vol. 20. Springer Science & Business Media (2010)
17. Hublet, F., Basin, D., Krstić, S.: Real-time policy enforcement with metric first-order temporal logic. In: European Symposium on Research in Computer Security. pp. 211–232. Springer (2022)
18. Hublet, F., Basin, D., Krstić, S.: Enforcing the GDPR. In: European Symposium on Research in Computer Security. pp. 400–422. Springer (2023)
19. Hublet, F., Lima, L., Basin, D., Krstić, S., Traytel, D.: Proactive real-time first-order enforcement. In: International Conference on Computer Aided Verification, (CAV) (2024), under revision

20. Huttner, L., Merigoux, D.: Catala: moving towards the future of legal expert systems. Artificial Intelligence and Law pp. 1–24 (2022)
21. Kvamme, A.: Bridging logics: Transforming formalizations of GDPR into enforceable mfotl specifications (2024)
22. Lam, P.E., Mitchell, J.C., Sundaram, S.: A formalization of hipaa for a medical messaging system. In: International Conference on Trust, Privacy and Security in Digital Business. pp. 73–85. Springer (2009)
23. Merigoux, D., Chataing, N., Protzenko, J.: Catala: a programming language for the law. Proceedings of the ACM on Programming Languages **5**(ICFP), 1–29 (2021)
24. Paal, B.P., Pauly, D.A., Ernst, S.: Datenschutz-Grundverordnung, Bundesdatenschutzgesetz. CH Beck München (2021)
25. Palmirani, M., Governatori, G.: Modelling legal knowledge for GDPR compliance checking. In: JURIX. vol. 313, pp. 101–110 (2018)
26. Palmirani, M., Governatori, G., Rotolo, A., Tabet, S., Boley, H., Paschke, A.: Legalruleml: Xml-based rules and norms. In: Rule-Based Modeling and Computing on the Semantic Web: 5th International Symposium, RuleML 2011–America, Ft. Lauderdale, FL, Florida, USA, November 3-5, 2011. Proceedings. pp. 298–312. Springer (2011)
27. Palmirani, M., Martoni, M., Rossi, A., Bartolini, C., Robaldo, L.: Pronto: Privacy ontology for legal reasoning. In: Electronic Government and the Information Systems Perspective: 7th International Conference, EGOVIS 2018, Regensburg, Germany, September 3–5, 2018, Proceedings 7. pp. 139–152. Springer (2018)
28. Robaldo, L., Bartolini, C., Lenzini, G.: The DAPRECO knowledge base: representing the GDPR in LegalRuleML. In: Proceedings of the Twelfth Language Resources and Evaluation Conference. pp. 5688–5697 (2020)
29. Robaldo, L., Bartolini, C., Palmirani, M., Rossi, A., Martoni, M., Lenzini, G.: Formalizing GDPR provisions in reified I/O logic: the DAPRECO knowledge base. Journal of Logic, Language and Information **29**, 401–449 (2020)
30. Sergot, M.J., Sadri, F., Kowalski, R.A., Kriwaczek, F., Hammond, P., Cory, H.T.: The british nationality act as a logic program. Communications of the ACM **29**(5), 370–386 (1986)
31. Torre, D., Alferez, M., Soltana, G., Sabetzadeh, M., Briand, L.: Modeling data protection and privacy: application and experience with GDPR. Software and Systems Modeling **20**, 2071–2087 (2021)