Monitoring the GDPR*

Emma Arfelt[1], David Basin[2], and Søren Debois[1]

[1] IT University of Copenhagen {ekoc, debois}@itu.dk
[2] ETH Zurich basin@inf.ethz.ch

Abstract The General Data Protection Regulation (GDPR) has substantially strengthened the requirements for data processing systems, requiring *audits at scale*. We show how and to what extent these audits can be automated. We contribute an analysis of which parts of the GDPR can be monitored, a formalisation of these parts in metric first-order temporal logic, and an application of the MonPoly system to automatically audit these parts. We validate our ideas on a case study using log data from industry, detecting actual violations. Altogether, we demonstrate both in theory and practice how to automate GDPR compliance checking.

Keywords: Data protection, GDPR, Compliance checking, Monitoring

1 Introduction

Problem. The EU's General Data Protection Regulation (GDPR) [24], which came into force in May 2018, is one of the most important changes and strengthening of privacy regulations in decades. The GDPR constitutes a legal data protection regime imposed on organisations processing personally identifiable information about EU citizens. The regulation is as comprehensive as it is severe: failure to comply with the technical and organisational requirements it imposes may result in fines up to the larger of 20 million Euro or 4% of the organisation's worldwide annual turnover.

The GDPR requires extremely fine-grained control over an organisation's data processing activities. For example, *every single use* of a data subject's personally identifiable data must have a *documented* legal basis. Data that is no longer necessary or conforms to that basis must promptly be deleted. Moreover, data subjects have certain rights concerning access to their data and imposing restrictions on processing activities regarding them. To be compliant, organisations must not only meet these requirements, but *document* this in a way that supports audits.

The GDPR raises numerous technical challenges for data protection researchers. Our focus in this paper is on tool-supported compliance checking: *how and to what extent can organisations automatically demonstrate compliance*, especially given that the GDPR's requirements apply to all data processing activities?

* This work supported in part by Innovation Fund Denmark project EcoKnow (7050-00034A). This paper does not constitute legal advice.

This raises sub-questions including:

1. What does the GDPR specifically require of systems?
2. What observations must we make of systems to verify compliance with these requirements?
3. To what extent can we automate compliance checking?

Approach Taken. For (1), we perform an in-depth analysis of the GDPR, identifying all articles that pose specific requirements on systems; see Tables 2, 3 and 4.

For (2), we identify among these articles those actions involving data processing, data subjects' rights, granting or revoking consent, or claiming a legal basis for processing. To automate audits, we will require that these actions are logged. We encounter two challenges here. First, GDPR relevant actions like "process data" or "revoke consent" are unlikely to directly appear as events in actual logs. We shall show that it is possible to *transform* logs so that this information is explicitly represented, enabling automated audits. Second, we shall see that automated audits cannot entirely replace human audits. For example, the GDPR requires that "information about processing is communicated to the data subject." While we can log information on the transmission of a message, we cannot in general check that the message's *contents* complies with the GDPR. Our work provides a complementary approach to auditing: Human auditors must verify by inspection and sampling that messages and documents have the proper contents. Machines can in turn be used to verify that such messages, documents, and processing activities happen when required. That is, human auditors are needed for intelligence and understanding whereas machines serve to verify compliance at scale.

For (3), we express the requirements in the articles identified in (1) as metric first-order temporal logic (MFOTL) formulae [6] over the actions identified in (2). MFOTL is a natural choice. The GDPR speaks about events, their associated data, and their temporal relationships (both qualitative and quantitative); this calls for a metric first-order temporal logic. Moreover, MFOTL is supported by the MonPoly tool, which implements a monitoring algorithm efficient enough for practical use [7,5].

As a simple example, consider GDPR Article 15(1), governing the *Right to Access*, which requires that any data subject has the right to demand from the controller access to all personal data concerning him that is processed by the controller. A simplified form of this article expressed in MFOTL might be:

$$\mathsf{ds\_access\_request}\,(dsid)\ \mathsf{IMPLIES\ EVENTUALLY}[0,30\mathrm{d}]\ \mathsf{grant\_access}(dsid)\ .$$

Here the universally quantified variable *dsid* ranges over distinct data subjects, and the interval $[0, 30\mathrm{d}]$ expresses that the controller must respond within 30 days.

MonPoly can be used *on-line* to monitor system events as they occur to find violations of this formula in real time. Alternatively, it can be used *off-line* to support a compliance audit, given the logged events. In either case, MonPoly will either declare "No violations", or "Violation of rule r found at time-point t" for each violation found. In this way, we obtain an automated audit tool.

Contributions. The answers to our research questions lead to a methodology for monitoring data protection requirements, in particular for auditing a system's compliance with the GDPR.

1. We identify GDPR clauses that can be verified by observing logged actions.
2. We encode in MFOTL key clauses of the GDPR, namely Articles 5(1c,1e), 6(1), 7(3), 13(1), 15(1), 17(1–2), 18(1–2), and 21(1).
3. We carry out a case study on an industry log and show how to extract GDPR-specific actions, and use MonPoly to find violations.

Altogether, we show that our MFOTL formalisation enables the algorithmic verification of essential parts of the GDPR. We note that the articles we verify are among the ones subject to the highest administrative fines, hence verifying compliance and non-compliance for these is particularly important.

Related work. Our paper is the first that provides running, automatic, compliance verification for the GDPR. Alternative approaches have been proposed based on design mechanisms or static analysis. In system design work, researchers have investigated augmenting existing formalisms with the concepts needed for reasoning about or enforcing the GDPR, e.g., adding relations between data and users, or relations between processing activities and consent or legal basis [1,2]. On the analysis side, [10,11] proposes variations of taint analysis to track the dispersal of personally identifiable information in GDPR-sensitive programs. Moreover, [3] proposes a mechanism to statically audit GDPR compliance that avoids directly analysing source code, extracting instead audit-relevant information from requirements specifications. A similar idea is presented in [19], which combines an ontology of GDPR concepts [20] with established methods for analysing business processes for regulatory compliance [13,12].

Outside of work specifically targeting the GDPR, several proposals have been made to use the "purpose of processing" as a factor in access control decisions [26,18,21,17,9]. In particular, the use of information-flow analysis (viz. taint analysis above) to support access control decisions was investigated in [15]. Closer to the present work, [22] investigated comparing business process models with information access logs to infer the legitimacy of processing for the purposes of access control. This idea might be used to refine the logged action representing a "legal basis" in the present paper from simply a claim that such a basis exists to support for this claim by appeal to the underlying business process (see also [3]).

Both MFOTL and MonPoly have been previously applied to privacy policies. In [4], examples of data protection rules were formulated in MFOTL. More recently, [14] investigated automatically rewriting data-flow programs to conform to privacy policies specified in MFOTL. The question arises whether the MFOTL formulae identified in the present paper can be directly used as inputs to that rewriting process. In general, the question of monitoring compliance for business processes has received considerable interest [16]; the log we consider in Section 6 is essentially the log of a business process execution.

Overview. In Section 2, we recall MFOTL's syntax and semantics and the MonPoly tool. Then, in Section 3, we analyse the GDPR and clarify which articles can neither be formalised nor monitored. We proceed to formalise in Section 4 the remainder of the GDPR in MFOTL. In Section 5, we show how to use these formulae for run-time monitoring with MonPoly. In Section 6, we apply our formalisation to an industry log and we draw conclusions in Section 7.

2 Background on MonPoly and MFOTL

We use Metric First-order Temporal Logic (MFOTL) [6] to formalise GDPR requirements, and we use the MonPoly [5,7] monitoring tool to decide whether a log conforms to a given MFOTL formula.

Metric First-order Temporal Logic (MFOTL) combines the two key properties needed to capture GDPR data protection policies: (1) the ability to relate individuals via first-order predicates, primarily data subjects, data classes, and data references, and (2) the ability to speak about events and data changing over time. Below, we briefly recall MFOTL; for a comprehensive reference, see [6].

A *signature* S is a tuple (C, R, ι), where C is a finite set of constant symbols, R is a finite set of predicate symbols disjoint from C, and the function $\iota : R \to \mathbb{N}$ associates each predicate symbol $r \in R$ with an arity $\iota(r) \in \mathbb{N}$. To illustrate, the signature for the previously presented formula regarding access defines two predicate symbols: $\mathsf{ds\_access\_request}(dsid)$ and $\mathsf{grant\_access}(dsid)$. Let $S = (C, R, \iota)$ be a signature and V a countably infinite set of variables, assuming $V \cap (C \cup R) = \emptyset$. The *syntax of formulae* over the signature S is given by the grammar in Figure 1. We present only a fragment of MFOTL, omitting equality, ordering, and the PREVIOUS operators, which we shall not need.

A *structure* $\mathcal{D}$ over the signature S comprises a domain $|\mathcal{D}| \neq \emptyset$ and interpretations $c^{\mathcal{D}} \in |\mathcal{D}|$ and $r^{\mathcal{D}} \subseteq |\mathcal{D}|^{\iota(r)}$, for each $c \in C$ and $r \in R$. A *temporal structure* over S is a pair $(\bar{\mathcal{D}}, \bar{\tau})$, where $\bar{\mathcal{D}} = (\mathcal{D}_0, \mathcal{D}_1, \dots)$ is a sequence of structures over S and $\bar{\tau} = (\tau_0, \tau_1, \dots)$ is a sequence of natural numbers, such that (1) $\bar{\tau}$ is non-decreasing and has no constant suffix ("time always eventually advances"); (2) $\bar{\mathcal{D}}$ has constant domains, that is, $|\mathcal{D}_i| = |\mathcal{D}_{i+1}|$, for all $i \geq 0$; and (3) each constant symbol $c \in C$ has a rigid interpretation, that is, $c^{\mathcal{D}_i} = c^{\mathcal{D}_{i+1}}$, for all $i \geq 0$. We denote c's interpretation by $c^{\bar{\mathcal{D}}}$. There can be successive time points with equal timestamps, and the relations $r^{\mathcal{D}_0}, r^{\mathcal{D}_1}, \dots$ in a temporal structure $(\bar{\mathcal{D}}, \bar{\tau})$ corresponding to a predicate symbol $r \in R$ may *change* over time. In contrast, the interpretation of the constant symbols $c \in C$ and the domain of the $\mathcal{D}_i$s do not change over time.

A *valuation* is a mapping $v : V \to |\bar{\mathcal{D}}|$. We abuse notation by applying a valuation v also to constant symbols $c \in C$, with $v(c) = c^{\bar{\mathcal{D}}}$. For a valuation v, a variable x, and $d \in |\bar{\mathcal{D}}|$, $v[x/d]$ is the valuation mapping x to d and leaving other variables' valuation unchanged. The *semantics* of MFOTL, $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$, is given in Figure 1, where $(\bar{\mathcal{D}}, \bar{\tau})$ is a temporal structure over the signature S, with $\bar{\mathcal{D}} = (\mathcal{D}_0, \mathcal{D}_1, \dots)$, $\bar{\tau} = (\tau_0, \tau_1, \dots)$, v a valuation, $i \in \mathbb{N}$, and ϕ a formula over S.

Meta-variables:

$t_1, t_2, \ldots$ range over $V \cup C$ r
x over R
V over values
I over intervals over $\mathbb{N}$.

Syntax:

$$\begin{aligned}\phi ::= {} & r(t_1, \ldots, t_{\iota(r)}) \\ & \mid \mathsf{NOT}\ \phi \\ & \mid \phi\ \mathsf{OR}\ \phi \\ & \mid \mathsf{EXISTS}\ x.\ \phi \\ & \mid \phi\ \mathsf{SINCE}[I]\ \phi \\ & \mid \phi\ \mathsf{UNTIL}[I]\ \phi\end{aligned}$$

Semantics:

$$\begin{array}{lll}
(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models r(t_1, \ldots, t_{\iota(r)}) & \text{iff} & \big(v(t_1), \ldots, v(t_{\iota(r)})\big) \in r^{\mathcal{D}_i} \\
(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \mathsf{NOT}\ \phi & \text{iff} & (\bar{\mathcal{D}}, \bar{\tau}, v, i) \not\models \phi \\
(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi\ \mathsf{OR}\ \phi & \text{iff} & (\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi \text{ or } (\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \psi \\
(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \mathsf{EXISTS}\ x.\ \phi & \text{iff} & (\bar{\mathcal{D}}, \bar{\tau}, v[x/d], i) \models \phi, \text{for some } d \in |\bar{\mathcal{D}}| \\
(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi\ \mathsf{SINCE}[I]\ \psi & \text{iff} & \text{for some } j \le i,\ \tau_i - \tau_j \in I, (\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \psi, \\
 & & \quad \text{and } (\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi, \text{for all } k \in [j+1, i+1) \\
(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi\ \mathsf{UNTIL}[I]\ \psi & \text{iff} & \text{for some } j \ge i,\ \tau_j - \tau_i \in I,\ (\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \psi, \\
 & & \quad \text{and } (\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi,\ \text{for all } k \in [i, j)
\end{array}$$

Figure 1: MFOTL Syntax and Semantics

Terminology and notation We use the following standard syntactic sugar:

$$\begin{aligned}\mathsf{ONCE}[I]\ \phi &:= \top\ \mathsf{SINCE}[I]\ \phi \\ \mathsf{EVENTUALLY}[I]\ \phi &:= \top\ \mathsf{UNTIL}[I]\ \ \phi \\ \mathsf{ALWAYS}[I]\ \phi &:= \mathsf{NOT}\ \mathsf{EVENTUALLY}[I]\ \mathsf{NOT}\ \phi\end{aligned}$$

We sometimes omit the interval I, understanding it to be $[0, \infty)$.

MonPoly [7] is a monitoring tool for deciding whether a log satisfies a formula in metric first-order temporal logic. Operationally, MonPoly accepts as inputs a signature, an MFOTL formula, and a log, and outputs the list of entries in the log that violate the formula [7]. The log must consist of a sequence of time-stamped system events ordered ascending by time. Technically, MonPoly accepts a formula φ with free variables $\bar{x}$ and checks $\mathsf{ALWAYS\ FORALL}\ \bar{x}.\varphi$. That is, it checks that $\mathsf{FORALL}\ \bar{x}.\varphi$ holds at every time point. As MonPoly must report the time points at which this formula is *violated*, in practice, MonPoly searches for and reports time points where the negated formula $\neg\varphi$ is satisfied.

Article(s)	Description
1–4	General provisions
6(2–4)	Member state restrictions on lawful processing
23	General member state restrictions
51–62	Supervisory authority
63–76	European data protection board
77–84	Remedies and penalties
92–99	Delegated acts and implementing acts

Figure 2: Articles unrelated to the compliance of an individual organisation.

3 Limits to GDPR monitoring

The GDPR [24] comprises 99 articles imposing specific rights and obligations on entities processing the personally identifiable information of "data subjects." In this section, we briefly categorise those articles that are not amendable for formalisation or are not suitable for automated compliance checking.

Articles unrelated to compliance As with any legal document, part of the GDPR is devoted to the legal framework surrounding the regulation: how EU member states should integrate the regulation into their local laws, the legislation's territorial scope, etc. These articles have no bearing on the question whether a particular organisation is in compliance, and as such, these are not relevant for mechanised compliance checking. We list these articles in Figure 2.

Articles unrelated to system behaviour The GDPR also imposes requirements unrelated to data processing or data subjects, regulating instead the form and functioning of the data processing organisation itself [25]. For instance, such an organisation must have a mechanism for notifying its local supervisory authority in case of a data breach, it must appoint a data protection officer, and it must be able to document that its systems comply with best IT-security practices. While these are clearly rules that an organisation can follow or break, such actions do not happen at scale.

Altogether, this class of articles, listed in Figure 3, makes no requirements on observable system actions, and so are irrelevant for compliance monitoring.

Articles requiring interpretation Many GDPR articles do not directly describe system actions, but regulate the contents of communications, e.g., Article 13(1) on information that controllers must provide, or Article 5(1d) requiring processed data to be accurate and up to date. We list the articles that do not directly relate to system actions (and not otherwise subject to auditing at scale) in Figure 4.

We shall see in Section 4.2 that even when we cannot verify that communicated contents satisfy the GDPR, we can at least monitor that communication *took place*.

Articles	Brief explanation
24–29	Organisational requirements
31	Cooperation with supervisory authority
32	Security of the system
33–34	Notification upon data breach
35–39	DPIA and DPO
40–43	Codes of conduct and certificates
44–50	Transfers to third countries
85–91	Specific processing situations

Figure 3: GDPR articles unrelated to system actions.

Art.	Desc.	Art.	Desc.
5(1) a-b,d,f; (2)	Principles of processing	14	Indirect collection
7(1–2,4)	Conditions for consent	16	Right to rectification
8	Child's consent	19	Requirement to notify
10	Processing of criminal records	20	Right to data portability
11	Processing w/o identification	22	Profiling
12	Transparency wrt. rights	30	Records of processing
13(3–4)	Information upon collection		

Figure 4: GDPR articles which do not directly relate system actions or regulate content

4 Formalisable articles

We can formalise and monitor articles where controllers, processors, or data subjects are required to take specific, observable actions in response to other specific, observable actions.

Recall that formalisation in MFOTL comprises two things: (1) a signature, specifying the actions and data we must be able to observe, and (2) a formula over that signature, specifying how those actions and data should evolve over time. We present below a signature of relevant actions, and a set of formulae over that signature formalising articles of the GDPR.

The elements of this signature are given as typed predicates. The procedure to formalise a requirement of the GDPR is as follows. First, identify both actions that trigger a requirement, e.g., *a data subject revokes his consent*, and those that are the required response, e.g., *the data controller ends processing*. Second, model these actions as predicates in an MFOTL signature. Finally, express the required causality as an MFOTL formula. In this section, we formalise GDPR Articles 5(1c), 5(1e), 6(1), 7(3), 13(1), 15(1), 17(1–2), 18(1–2), and 21(1). The corresponding signature and rules are given in Tables 5 and 6 respectively. Note that MonPoly imposes some syntactic restrictions; thus to run some of the rules with MonPoly, we must negate them by hand. These are marked with '$*$' in Table 6.

Predicate	Action
ds_deletion_request(*data, dataid, dsid*)	Data subject requests deletion
ds_access_request(*dsid*)	Data subject requests access
ds_consent(*dsid, data*)	Data subject gives consent
ds_restrict(*data, dataid, dsid*)	Data subject restricts processing of specific data
ds_repeal(*data, dataid, dsid*)	Data subject lift their restriction on specific data
ds_object(*dsid, data*)	Data subject objects to processing based on Art. 6 (1e-f)
legal_grounds(*dsid, data*)	Organisation claims legal basis
ds_revoke(*dsid, data*)	Data subject revokes consent
delete(*data, dataid, dsid*)	Controller deletes specified data
grant_access(*dsid*)	Controller grants access to specified data subject
share_with(*processorid, dataid*)	Controller shares data with a particular processor
inform(*dsid*)	Controller informs data subject about collection of data
notify_proc(*processorid, dataid*)	Controller notifies a processor of deletion
use(*data, dataid, dsid*)	Controller processes data of specified data subject
collect(*data, dataid, dsid*)	Controller collects data of specified data subject

Figure 5: MFOTL signature for the GDPR formalisation

We proceed by illustrating this analysis for the representative cases of Article 6(1) and 7(3) and we conclude by discussing outliers.

4.1 The common case: Articles 6(1) and 7(3)

Article 6 is at the GDPR's core: it defines what is required for lawful processing.

> *Processing shall be lawful only if and to the extent that at least one of the following applies:*
> *(a) the data subject has given consent to the processing of his or her personal data for one or more specific purposes;*
> *(b) processing is necessary for the performance of a contract to which the data subject is party or in order to take steps at the request of the data subject prior to entering into a contract; [...]*

To formalise this requirement, we must be able to observe both the processing of data and the establishment of legal grounds for that processing. While we cannot verify that a claim of legal grounds will hold up in court, we can, however, verify whether there exists a *claim* of legal grounds at all. A legal ground is for a specific class of data and a specific data subject. Thus we must be able to observe from our system an action legal_ground, represented as

$$\mathsf{legal\_ground}(\mathit{dsid}, \mathit{data}),$$

which is a predicate we add to our signature. The first argument to legal_ground represents a class of data (e.g. "ADDRESS" or "TELEPHONE NUMBER"), and the second is an identifier for a data subject. Note that MonPoly supports types, but for reasons of space we omitted this from Section 2 and our account here.

As the GDPR has special rules for consent-based processing (as we demonstrate later), it is convenient to single-out consent from other legal bases mentioned in Article 6(1). For this, we need the predicate

$$\mathsf{ds\_consent}(\mathit{dsid}, \mathit{data}).$$

When data is eventually used, we must know both which class of data is being processed and which data subject that data concerns. We shall see later (for erasure requirements) that we need a reference to the actual data as opposed to just its class (i.e. "+1 451 451-0000" or "DATABASE ROW 2769" as opposed to "TELEPHONE NUMBER"):

$$\mathsf{use}(\mathit{data}, \mathit{dataid}, \mathit{dsid}).$$

This predicate takes a data class, an identifier for the actual data processed, and an identifier for the data subject.

The GDPR in general conflates the *use of data* with *collecting data* into the single term *processing of data*. To formalise some articles, we will however need this distinction, so we add the following predicate to our signature:

$$\mathsf{collect}(\mathit{data}, \mathit{dataid}, \mathit{dsid}).$$

We now formalise that a data controller must have either consent from data subjects or another legal basis to process any data [[24], Article 6, sec. 1]. Otherwise the processing of data is prohibited. We formalise this requirement as our first MFOTL formula:

$$\mathsf{use}(\mathit{data},\ \mathit{dataid},\ \mathit{dsid})\ \mathsf{IMPLIES}\\ \mathsf{ONCE}\ (\mathsf{ds\_consent}(\mathit{dsid},\ \mathit{data})\ \mathsf{OR}\ \mathsf{legal\_ground}(\mathit{dsid},\ \mathit{data}))\,.$$

Recall that we consider MFOTL formulae to be implicitly forall-quantified over their free variables. Hence, the above formula states that for any class of data *data*, any concrete reference *dataid* to such data, and any data subject *dsid*, then: If at any point in time we observe processing of any data *dataid* of class *data* for *dsid*, then there must be a point in the past where we observed either consent from that *dsid* for processing *data*, or other legal grounds.

As a second, more subtle, example, consider Article 7(3), "Conditions for consent." It states that a data subject can revoke his consent at any time:

> *The data subject shall have the right to withdraw his or her consent at any time. [...]*

Absent other legal grounds, subsequent processing would then be illegal (viz. Article 6). To model this, we add to our signature a predicate representing a revocation of consent:

$$\mathsf{ds\_revoke}(\mathit{dsid},\ \mathit{data}),$$

and the formula:

$$\mathsf{use}(\mathit{data},\ \mathit{dataid},\ \mathit{dsid})\ \mathsf{IMPLIES}\ (\mathsf{ONCE}\ \mathsf{legal\_grounds}(\mathit{dsid},\ \mathit{data}))\\ \mathsf{OR}\ (\mathsf{NOT}\ \mathsf{ds\_revoke}(\mathit{dsid},\ \mathit{data})\ \mathsf{SINCE}\ \mathsf{ds\_consent}(\mathit{dsid},\ \mathit{data}))\,.$$

That is, if at some time point t we process data, then either we have legal grounds (in which case the revocation does not affect the right to process the data) or before that point t, consent was obtained and at no point between t and the given consent do we have a revocation. Note that this formula also finds violations in situations where Article 6 is violated.

4.2 Articles requiring content interpretation

As discussed in Section 3, we can monitor whether a required action is taken, and leave to a human auditor the question of whether the content complies with requirements prescribed by the GDPR. Articles 13, 15, 17(1), 18, and 21, describing the rights of the data subjects, has such conditions. These rights might frequently be exercised, and thus it is impractical to rely solely on human audits to determine if the company has responded as required. A human auditor could decide if the company's strategy for responding is compliant, whereas monitoring can help ensure, at scale, that the company responds when appropriate.

As an example, consider Article 17(1) "Right to erasure." This article defines under what circumstances a data controller or processor must delete data:

> *The data subject shall have the right to obtain from the controller the erasure of personal data concerning him or her without undue delay and the controller shall have the obligation to erase personal data without undue delay where one of the following grounds applies: [...]*

We refrain from further specifying the data subject's ground for deletion, referencing them all under the single action "deletion requested" (ds_deletion_request). We similarly omit explicitly modelling the exceptions mentioned in 17(3). It requires further human interpretation to determine the legality of a data subject's claim, and whether the controller is obligated to delete the data.

At this point, we must distinguish between classes of data and individual data items. There are two reasons: (1) the data subject may request some but not all data in a class to be erased, e.g., a data subject may request that an airline removes as an emergency contact his ex-wife but not his father. (2) To properly verify compliance, we need a formula that identifies and specifies the removal of every single data item processed for this data subject. Altogether, the action for the deletion request is

$$\mathsf{ds\_deletion\_request}(\mathit{data}, \mathit{dataid}, \mathit{dsid}).$$

Upon receiving such a request, the data controller must then respond and delete the data. This action must also be observable:

$$\mathsf{delete}(\mathit{data}, \mathit{dataid}, \mathit{dsid}).$$

The deletion in Article 17(1) is subsequently required to happen *without undue delay*, which in Recital 59(3) is limited to "at most one month." It is now straightforward to model this rule:

$$\mathsf{ds\_deletion\_request}\ (\mathit{data}, \mathit{dataid}, \mathit{dsid})$$
$$\quad \mathsf{IMPLIES\ EVENTUALLY[0,30d]}\ \mathsf{delete}(\mathit{data}, \mathit{dataid}, \mathit{dsid}).$$

That is, if at some time point t we are required to delete some particular data then within 30 days after t, we must observe this data being deleted. Moreover, it should be impossible to subsequently *process* deleted data:

$$\mathsf{use}(\mathit{data}, \mathit{dataid}, \mathit{dsid})\ \mathsf{IMPLIES}$$
$$\quad \mathsf{NOT\ ONCE}\ \mathsf{ds\_deletion\_request}(\mathit{data}, \mathit{dataid}, \mathit{dsid}).$$

4.3 Articles not monitorable

We conclude this section by considering Articles 5(1c,1e), "Data minimisation" and "Storage limitation." These articles require that:

> *Personal data shall be: [...]*
> *(c) adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed ('data minimisation'); [...]*

(e) kept [...] for no longer than is necessary for the purposes for which the personal data are processed; [...] ('storage limitation');

That is, we can never collect or store data that we will not subsequently use for a legitimate purpose (1c). Moreover, not only must we delete that data once it has outlived its purpose, with some exceptions, perpetual storage is prohibited outright (1e).

Storage limitation Recall that when we write EVENTUALLY ϕ, we implicitly intend the interval $[0, \infty)$, and thus formalising (1e) is straightforward:

$$\mathsf{collect}\,(\mathit{data}, \mathit{dataid}, \mathit{dsid})\ \mathsf{IMPLIES\ EVENTUALLY}\ \mathsf{delete}(\mathit{data},\ \mathit{dataid},\ \mathit{dsid}).$$

However, this formula is not finitely falsifiable and hence it cannot be monitored. Because it uses the unbounded EVENTUALLY modality, it is a *liveness* property, requiring something to eventually happen, without stipulating exactly when.

Data minimisation Here is an attempt to specify Article (1c):

$$\mathsf{collect}\,(\mathit{data}, \mathit{dataid}, \mathit{dsid})\ \mathsf{IMPLIES\ EVENTUALLY}\ \mathsf{use}(\mathit{data},\ \mathit{dataid},\ \mathit{dsid}).$$

By the semantics of MFOTL, this formula requires that the collected data must find use *in every run* of the system. This interpretation is likely too strong. As an example, when customers book long-haul flights, they may provide an emergency contact. However, the airline will only use this contact should an accident occur, so in the majority of cases, this data will be collected, not used, and then deleted.[3] Moreover, this requirement is also not monitorable because it is not finitely falsifiable, and requires some relaxation to be formulated precisely.

We formulate both data minimisation and storage limitation and include them in the Figure 6. However, as described above, neither are monitorable.

5 Run-time Monitoring

We now turn to the question: Does our formalisation of GDPR requirements lend itself to run-time monitoring? We show how to take logs of running systems and use a tool to *verify automatically* that these logs conform to the given formulae.

5.1 Methodology

Having established that the formulae of Figure 6, or equivalent formulations thereof, are accepted as inputs to MonPoly, we turn to the question of how to obtain a log containing the actions described in Figure 5. System logs conventionally contain information about which events happened and when they

[3] In fact, in 2017, no commercial airline passengers died from plane crashes [23], and thus presumably emergency contact data was unnecessary.

Article	MFOTL Formula
5(1)(c) Data minimisation	collect (*data*, *dataid*, *dsid*) IMPLIES EVENTUALLY use(*data*, *dataid*, *dsid*)
5(1)(e) Storage limitation	collect (*data*, *dataid*, *dsid*) IMPLIES EVENTUALLY delete(*data*, *dataid*, *dsid*)
6(1) Lawful processing	use(*data*, *dataid*, *dsid*) IMPLIES ONCE (ds_consent(*dsid*, *data*) OR legal_grounds(*dsid*, *data*))
7(3) Consent	use(*data*, *dataid*, *dsid*) IMPLIES (ONCE legal_grounds(*dsid*, *data*)) OR (NOT ds_revoke(*dsid*, *data*) SINCE ds_consent(*dsid*, *data*))
13(1) Info. on collection	collect (*data*, *dataid*, *dsid*) IMPLIES NEXT inform(*dsid*) OR ONCE inform(*dsid*)
15(1) Right to access	ds_access_request (*dsid*) IMPLIES EVENTUALLY[0,30d] grant_access(*dsid*)
17(1) Right to erasure	ds_deletion_request (*data*, *dataid*, *dsid*) IMPLIES EVENTUALLY[0,30d] delete(*data*, *dataid*, *dsid*)
17(1) Right to erasure	use(*data*, *dataid*, *dsid*) IMPLIES NOT ONCE delete(*data*, *dataid*, *dsid*)
17(2) Right to erasure∗	ds_deletion_request (*data*, *dataid*, *dsid*) AND ONCE share_with(processorid , *dataid*) AND NOT EVENTUALLY[0,30d] notify_proc(procid , *dataid*)
18(1-2) Right to restriction of processing∗	use(*data*, *dataid*, *dsid*) AND (NOT ds_repeal(*data*, *dataid*, *dsid*) SINCE ds_restrict (*data*, *dataid*, *dsid*))
21(1) Right to object	use(*data*, *dataid*, *dsid*) IMPLIES (NOT ds_object(*dsid*, *data*) SINCE legal_grounds(*dsid*, *data*))

Figure 6: MFOTL formulae expressing GDPR requirements

occurred [7]. For example, the event that a data subject asks for access or that data was shared with other processors, and the date and time this occurs.

It is not conventional (at least prior to the GDPR) to log whether an organisation has a legal ground for data processing (and *which* legal ground) or obtained consent from a data subject. However, entries in a system log often *reflect* GDPR actions such as establishing a legal basis. For instance, if a customer clicked "purchase" in a web-shop, this establishes a legal basis for using the customer's postal address for shipping. In general, we can infer GDPR actions from log entries by having domain experts apply their knowledge of the system to log entries.

Altogether, we propose the following methodology for partially verifying a system's compliance with the GDPR using run-time monitoring:

1. Identify available logs.
2. Identify the types of records in each log and relate each type to GDPR actions (Table 5). In general, this may require input from a domain expert and a systems expert, and possibly also a GDPR expert. Write a script or a program to transform automatically logs entries to GDPR actions.
3. Run MonPoly on the transformed log, using the rules of Table 6.

Obviously, this methodology depends on being able to find or infer GDPR relevant actions in the logs. We shall see in the next section how such inference is possible from an otherwise unhelpful looking real-world log.

6 Case study

We now apply the above methodology to a concrete, real-life industry log previously published in [8], which described the context of this log as follows:

The Dreyer Foundation awards grants to [...] activities [...] promoting the development of the lawyer and architect professions [...]. Roughly, an application is processed as follows. Applications are accepted in rounds. In each round, first, a caseworker pre-screens applications, weeding out obvious rejects. The remaining applications are independently reviewed by 2-4 reviewers, at least one of which must be an architect or a lawyer, depending on the type of application. Once all reviews are in, the Foundation's board decides on which applications to accept at a board meeting. Successful applications then have a running payout, until the grant period expires and an end-report is produced.

Step 1: Define the log

The log itself is from an adaptive case-management system supporting this work; it documents the processing steps taken. The log contains 12,151 events concerning 587 individual applications processed in the period December 2013–June 2015. We present an excerpt of the log in Figure 8. In the interest of presentation, we have removed and shortened the individual fields within each line of

Dreyer log entry title	GDPR action and description
`Application received`	ds_consent(*"APPL"*, *i*) When submitting his application, an applicant must also provide explicit consent for subsequent processing.
`Complete`	delete(*"APPL"*, *i*) / delete(*"ACCOUNT"*, *i*) The application has been approved and fully payed out. There are no remaining purposes for storing collected data.
`Approve`	legal_grounds(*i*, *"ACCOUNT"*) When an application is approved, we have legal grounds for storing and using the account number of the applicant.
`Retract application`	ds_deletion_request(*"APPL"*, *i*, *i*) If the applicant retracts the application, we no longer have legitimate purposes for storage or processing. We model this by requiring application data to be deleted.
`Notify (rejected)`	delete(*"APPL"*, *i*, *i*) Once we have notified the applicant that his application has been rejected, we delete the application.
`Review` (and others)	use(*"APPL"*, *i*, *i*) Along with various other actions, the application is reviewed processing the data inside it.
`Round ends` (and others)	(no action) Remaining records do not process data and thus are not relevant for monitoring.

Figure 7: Mapping of Dreyer log entries to GDPR actions. The i refers to the application instance id available in the log.

the log, re-ordered the remaining fields, and translated some log-entries to English. Note that the excerpt is non-contiguous, with actual line numbers in the log for each line given on the left. The excerpt describes a successful application, going through initial submission (78), screening (520), reviews (1483–1861), board meeting and eventual approval (2130–3779), payout (5378–5423), and finally inclusion in the end report of the round (11224–11235). Along the way, the applicant is notified about the application's state (3308, 4679).

Step 2: Transform the log

Most importantly we must extract from this log the actions listed in Figure 5 using domain knowledge of the meaning of the system actions underlying the log entries. We give the full list of Dreyer log actions, their semantics, and the corresponding GDPR actions in Figure 7. We encourage the reader to carefully consider the "Description" column, which explains exactly how domain knowledge justifies the connection between a log entry and a GDPR action.

```
78 63;20140108 0955;Appl. received  @1389171305 ds_consent("63", "APPL")
520 63;20140127 1802;Pass screening  @1390842175 use("APPL", "63", "63")
1483 63;20140313 1027;Lawyer review   @1394702823 use("APPL", "63", "63")
1505 63;20140313 1322;Review          @1394713322 use("APPL", "63", "63")
1565 63;20140316 2336;Review          @1395009396 use("APPL", "63", "63")
1861 63;20140323 2212;Arch. review    @1395609120 use("APPL", "63", "63")
2130 63;20140327 0917;Record decision @1395908246 use("APPL", "63", "63")
2135 63;20140327 0918;Board meeting
2691 63;20140409 0300;Round ends
3071 63;20140415 1450;Round approved
3308 63;20140416 1036;Notify 1        @1397637397 use("APPL", "63", "63")
3544 63;20140416 1925;Round approved
3779 63;20140416 1925;Approve         @1397669105 legal_grounds("63", "ACCOUNT")
4679 63;20140521 1141;Notify 2        @1400665291 use("APPL", "63", "63")
5378 63;20140626 1402;Payout 1        @1403784135 use("ACCOUNT", "63", "63")
5423 63;20140626 2054;Payment done    @1403808852 use("ACCOUNT", "63", "63")
11224 63;20150503 2328;Final report   @1430689888 use("ACCOUNT", "63", "63")
                                                  use("APPL", "63", "63")
11235 63;20150503 2352;Complete       @1430689922 delete("APPL", "63", "63")
                                                  delete("ACCOUNT", "63", "63")
```

Figure 8: Excerpt of Dreyer log (left) and corresponding transformed log (right).

The opening "Appl(ication) received" entry signifies the applicant filling in and submitting an electronic application form. This form includes a tick box indicating consent to subsequent processing for the purpose of considering the application; the application cannot be submitted without ticking this box. Thus, we can associate with this log entry the ds_consent action for the application data.

The Dreyer log gives us little information about exactly what data is processed. But we know that the application in its entirety is necessarily processed in reviews and decision making, and that the (subsequently supplied) account number of a successful applicant is used in payout steps. We note that by the purpose limitation, there is no legal ground to request an account number for payouts until a grant is awarded.

The data subject is not directly represented in the log; however, the log contains (first field) a number uniquely identifying the application. As each application conveniently has exactly one applicant, we conflate the application id and the data-subject. Altogether, we interpret the first line (78) in Figure 8 as the GDPR action ds_consent(*"APPL"*, *0063*), that is, consent from the data subject identified by 0063 to the processing of his application data.

Once we have established the mapping table in Figure 7, it is straightforward to automatically transform an input log into a MonPoly-compatible log of GDPR actions. We have constructed such an automatic transformation; the result of applying it to our log excerpt is also shown in Figure 8. The lines of the

original and transformed log are aligned vertically, e.g., line 5378 of the input (left) yields the transformed line `@1403808852 use(...)` (right).

Altogether, this demonstrates that we *can* extract GDPR-relevant actions from a realistic industry log. Our coverage, however, is only partial: the mapping of Dreyer log entries does not contain the actions necessary to monitor, e.g., the right to access (Article 15(1)) or erasure of previously shared data (Article 17(2)).

Step 3: Verify compliance with monitoring

We can now provide the transformed log and the formulae capturing the GDPR rules from Figure 6 as inputs to MonPoly. We discover the following violations:

- 8 violations of lawful processing (Article 6(1)). Of these, 7 arise because an account number was submitted *before* the application was approved, thus processing that data without legal grounds. The remaining violation arose because in a single instance, money was paid out even though the application in question was never recorded as approved (or rejected) in the log. Hence payment information was used without legal grounds.
- 8 violations of the Right to Erasure (Article 17(1)), part (i). These 8 were all retracted shortly after being submitted; however, they were never deleted.
- 1 violation of the Right to Erasure (Article 17(1)), part (ii). In this instance, the last payment of a successful application was acknowledged and recorded in the log only *after* the entire application was recorded as completed.

These violations range from seemingly inconsequential mistakes (*a payment being recorded late*) to a definite violation (*not having a process for deleting no-longer necessary data*). We note that by the letter of the GDPR, there were no false positives: inconsequential mistakes are still violations.

This log pre-dates the GDPR: When it was produced, the Dreyer foundation was not obligated to be GDPR compliant.

Summary

Altogether, we have demonstrated that our proposed method of using MonPoly *can* in fact find instances of GDPR non-compliance in a real-life industry log. Moreover, with the GDPR formulae (see Figure 6) already in place, the only significant work required to check the log is to map the log's contents to GDPR actions; that is, working out Table 7. This work required domain knowledge (e.g., you only need the account number once the application is approved) *in combination* with an understanding of the GDPR (e.g., you need a legal ground to store the account number). Once the mapping is established, it is trivial to write a small program to automatically produce the transformed log suitable for MonPoly. Subsequent processing by MonPoly is then automatic: the log in question is processed nearly instantaneously.

7 Conclusion

Our analysis has shown that monitoring can be used to automate compliance checking for significant parts of the GDPR. We explained why some parts of the GDPR elude monitoring and require other auditing measures or other forms of verification. We also identified and tackled challenges in extracting relevant actions for monitoring from real-world logs. Finally, we showed the value of this in a case study where we found violations ranging from apparently inconsequential to almost certainly non-compliant.

We see this work as a beginning: providing automated support for compliance checking for the GDPR and similar privacy regulations. As future work, we would like to apply our ideas to larger case studies, to both help organisations improve their handling of data and to verify their compliance. A research question here concerns how best to instrument systems with logging functionality to produce adequate logs at the right level of detail. Another question concerns distinguishing between personally identifiable information and other kinds of information, as we now only verify that the controller took the required actions. Progress here could, for example, allow us to extend our approach to monitor articles that impose requirements on content.

References

1. Thibaud Antignac, Riccardo Scandariato, and Gerardo Schneider. A Privacy-Aware Conceptual Model for Handling Personal Data. In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*, volume 9952, pages 942–957. Springer International Publishing, Cham, 2016.
2. Thibaud Antignac, Riccardo Scandariato, and Gerardo Schneider. Privacy compliance via model transformations. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 120–126. IEEE, 2018.
3. David Basin, Søren Debois, and Thomas Hildebrandt. On Purpose and by Necessity: Compliance under the GDPR. In *Proceedings of the 22nd International Conference on Financial Cryptography and Data Security (FC '18)*, Lecture Notes in Computer Science, Nieuwpoort, Curaçao, February 2018. Springer. Accepted for publication.
4. David Basin, Matus Harvan, Felix Klaedtke, and Eugen Zalinescu. Monitoring data usage in distributed systems. *IEEE Transactions on Software Engineering*, 39(10):1403–1426, 2013.
5. David Basin, Matúš Harvan, Felix Klaedtke, and Eugen Zălinescu. MONPOLY: Monitoring usage-control policies. In *Proceedings of the 2nd International Conference on Runtime Verification (RV)*, volume 7186 of *Lecture Notes in Computer Science*, pages 360–364. Springer-Verlag, 2012.
6. David Basin, Felix Klaedtke, and Samuel Müller. Monitoring security policies with metric first-order temporal logic. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, pages 23–34. ACM, 2010.
7. David Basin, Felix Klaedtke, and Eugen Zalinescu. The monpoly monitoring tool. In *RV-CuBES 2017. An International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools*, volume 3 of *Kalpa Publications in Computing*, pages 19–28. EasyChair, 2017.

8. Søren Debois and Tijs Slaats. The Analysis of a Real Life Declarative Process. In *IEEE Symposium Series on Computational Intelligence, SSCI 2015, Cape Town, South Africa, December 7-10, 2015*, pages 1374–1382. IEEE, 2015.
9. Md. Enamul Kabir, Hua Wang, and Elisa Bertino. A conditional purpose-based access control model with dynamic roles. *Expert Systems with Applications*, 38(3):1482–1489, March 2011.
10. Pietro Ferrara, Luca Olivieri, and Fausto Spoto. Tailoring Taint Analysis to GDPR. In *Privacy Technologies and Policy - 6th Annual Privacy Forum, APF 2018, Barcelona, Spain, June 13-14, 2018, Revised Selected Papers*, volume 11079 of *Lecture Notes in Computer Science*, pages 63–76. Springer, 2018.
11. Pietro Ferrara and Fausto Spoto. Static Analysis for GDPR Compliance. In *Proceedings of the Second Italian Conference on Cyber Security, Milan, Italy, February 6th - 9th, 2018*, volume 2058 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.
12. Guido Governatori. Business process compliance: An abstract normative framework. *IT–Information Technology*, 55(6):231–238, 2013.
13. Guido Governatori and Shazia Sadiq. The journey to business process compliance. In *Handbook of research on business process modeling*, pages 426–454. IGI Global, 2009.
14. Michele Guerriero, Damian Andrew Tamburri, and Elisabetta Di Nitto. Defining, Enforcing and Checking Privacy Policies in Data-intensive Applications. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '18, pages 172–182, New York, NY, USA, 2018. ACM. event-place: Gothenburg, Sweden.
15. N. V. N. Kumar and R. K. Shyamasundar. Realizing Purpose-Based Privacy Policies Succinctly via Information-Flow Labels. In *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, pages 753–760, December 2014.
16. Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil M. P. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information Systems*, 54:209–234, December 2015.
17. Amirreza Masoumzadeh and James B. D. Joshi. PuRBAC: Purpose-Aware Role-Based Access Control. In *On the Move to Meaningful Internet Systems: OTM 2008*, Lecture Notes in Computer Science, pages 1104–1121. Springer, Berlin, Heidelberg, November 2008.
18. Qun Ni, Elisa Bertino, Jorge Lobo, Carolyn Brodie, Clare-Marie Karat, John Karat, and Alberto Trombeta. Privacy-aware Role-based Access Control. *ACM Trans. Inf. Syst. Secur.*, 13(3):24:1–24:31, July 2010.
19. Monica Palmirani and Guido Governatori. Modelling Legal Knowledge for GDPR Compliance Checking. *Frontiers in Artificial Intelligence and Applications*, 313:101–110, 2018.
20. Monica Palmirani, Michele MARTONI, Arianna ROSSI, Cesare BARTOLINI, and Livio ROBALDO. Legal Ontology for Modelling GDPR Concepts and Norms. In *Legal Knowledge and Information Systems: JURIX 2018: The Thirty-first Annual Conference*, volume 313, page 91. IOS Press, 2018.
21. H. Peng, J. Gu, and X. Ye. Dynamic Purpose-Based Access Control. In *2008 IEEE International Symposium on Parallel and Distributed Processing with Applications*, pages 695–700, December 2008.

22. Milan Petković, Davide Prandi, and Nicola Zannone. Purpose Control: Did You Process the Data for the Intended Purpose? In *Secure Data Management*, Lecture Notes in Computer Science, pages 145–168. Springer, Berlin, Heidelberg, September 2011.
23. David Shepardson. 2017 safest year on record for commercial passenger air travel, 2018. `https://reut.rs/2CvBTEH`.
24. European Union. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation), 2016.
25. Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer Publishing Company, Incorporated, 1st edition, 2017.
26. N. Yang, H. Barringer, and N. Zhang. A Purpose-Based Access Control Model. In *Third International Symposium on Information Assurance and Security*, pages 143–148, August 2007.