A Compliance Mechanism for Planning in Privacy Domain Using Policies

Yousef Taheri, Gauvain Bourgne, Jean-Gabriel Ganascia

A Compliance Mechanism for Planning in Privacy Domain Using Policies

Yousef Taheri, Gauvain Bourgne, and Jean-Gabriel Ganascia

Sorbonne Universite, CNRS, LIP6, 75005 Paris, France
{yousef.taheri, gauvain.bourgne, Jean-gabriel.ganascia}@lip6.fr

Abstract. As more and more applications relying on the use and processing of personal data grow, privacy protection is becoming increasingly important. With the enforcement of the GDPR, such applications must guarantee compliance with the obligations set forth. Integrating a compliance checking mechanism with AI methods is helpful to fulfill this requirement. Toward this end, we investigate the GDPR automatic compliance checking using a planning system including personal data and an agent with actions that process data. We propose a modular framework that is capable to generate possible plans (sequence of data processing) to satisfy a given goal state, check the compliance of the plan with GDPR regulatory constraints, and provide explanation of missing obligations in case of a non-compliant. We use Answer Set Programming(ASP) and event calculus formalism to model the planning problem and make use of SPECIAL policy language as an existing work to translate GDPR requirements into ASP.

Keywords: Automatic Compliance Checking · AI planification · Compliance Mechanism · Personal Data Privacy

1 Introduction

AI applications handling personal data is being largely adopted by many companies and data processors to deliver their services to their users. Building trustworthy AI and enhancing liability in society requires tools and techniques to ensure users privacy protection. The European General Data Protection Regulation (GDPR) provides legal requirements concerning personal data processing. Organizations need to take technical measures to evaluate the compliance of personal data processing with GDPR. Compliance mechanism tools help these organizations to fulfill their need for compliance assessment.

Many of the AI products require handling personal data through an automated procedure, therefore, they need to be integrated with compliance mechanisms to operate lawfully. Most of the current works on compliance checking focus on either representing GDPR concepts as Palmirani et al. [12] or building policy pipeline for representing regulatory norms and business policies De Vos et al. [6], Bonatti et al. [4]. These tools are built for assessing the compliance of a business policy, which is used to represent characteristics of a data processing.

However, none of them has studied the integration of GDPR compliance checking in planning or an automatic data manipulation setting. Bandara et al. [2] introduces a policy specification and enforcement method in a dynamic environment. They use it for detecting conflicts among policies and do not concern with personal data and privacy protection. De Vos et al. [5] propose a methodology to support legal reasoning using institutions (systems that specify normative behavior of participants) and a corresponding computational model. Given a set of observed actions their method captures the evolution of legal model after each action in a multi-agent setting, but it does not include automatic action generation and also does not deal with privacy domain.

In this paper, we build an agent that is capable to generate a sequence of personal data manipulations that are compliant with GDPR legal constraints. The agent can change the state of the system by performing a data processing *e.g.* transfer, analyze, etc. Each state in the planning domain represents a characteristic of the personal data. The agent is capable to reach a given a target state, by performing a number of processes on personal data called a plan. We are interested in the plan's compliance with GDPR regulatory norms, including the data subject's given consent. Toward this end, we propose a modular framework with the following components.

- **Planning**, Given a goal state, an initial state and a description of the domain, this module generates all possible plans regardless of their compliance.
- **Compliance Engine**, Given a plan, this module checks for its compliance against GDPR regulatory norms, and data subject's given consent. In the case of a non-compliant plan, it explains the missing obligations.

In order to do planning while checking for compliance we need a logical formalism that deals with sequence of actions while keeping track of the world state at each step of execution. Furthermore, since expressivity is essential in the current (and future) work, such formalism should enable modeling complex narratives with multiple events. A useful candidate, is event calculus (cf. Shanahan [15]), a well-known formalism in the planning literature. Event calculus uses an explicit linear representation of time in which fluents hold and events occur. In other words, the linear time-line is used to associate states and events when a change happens in the world. Combined with non-monotonic reasoning, event calculus allows exploring different alternatives as required for planning. Therefore, it eliminates the need for a more expressive or complex formalism based on branching-time, such as situation calculus. We give a more precise description at Section 2.2.

We also need to represent GDPR regulatory norms, in order to support compliance checking. To do so, we chose to use the SPECIAL policy language which offers a unified representation of regulatory norms and data subject's given consent to enable compliance checking of business policies.

In a legal setting we deal with a potentially large body of knowledge in the related law and there are usually multiple constraints which need to be verified to assess compliance. In order to implement such a domain we need a formalism that allows on one hand to express both rules and facts, and on the

other hand it has to support non-monotonic reasoning (since we would normally encounter situations with incomplete information). Answer Set Programming (ASP) (cf. Lifschitz [10]) is a proper candidate for this purpose. A knowledge representation and reasoning paradigm with an expressive formalism and efficient solvers. ASP is largely used for common sense reasoning, abductive and deductive reasoning and is especially suitable for planning and dealing with incomplete knowledge. ASP is compatible with event calculus and is a practical choice for future developments of our current work. We use Clingo by Gebser et al. [7], as an answer set solver for ASP. Briefly speaking, it is composed of two main steps; (i)The grounder which takes as input the provided knowledge, substitute all variables with the given instances, (ii)The solver which takes the extended knowledge of the previous step as input and extract the answer sets which are basically stable models of the program.

The rest of the paper is organized as follows: Section 2 presents both a brief background on SPECIAL policy language and the version of event calculus used to model the planning domain. In Section 3 we explain our modular framework and describe how each component is constructed. In Section 4 we evaluate our framework on two simple scenarios, and discuss the results. Section 5 presents a brief discussion of the related works. Finally, in Section 6 we discuss the conclusions and mention the future works.

2 Backgrounds

2.1 The SPECIAL Policy Language

In order to support automated compliance checking, we need a representation of GDPR requirements in a machine understandable format. Such representation is required to construct the compliance engine in our modular framework (see Figure 1). There are many works trying to develop an ontology or a policy pipeline to represent GDPR requirements for compliance checking. Here we make use of the SPECIAL, a policy language based on OWL2, that offers a unified representation for consent, business policies, and regulatory obligations.

In the SPECIAL policy language, a personal data processing is formalized as a business policy. The following is an example of a business policy, each attribute describes a characteristic of the personal data handling.

```
ObjectIntersectionOf(
    ObjectSomeValuesFrom(spl:hasData svd:purchasesAndSpendingHabit)
    ObjectSomeValuesFrom(spl:hasProcessing svpr:Analyze)
    ObjectSomeValuesFrom(spl:hasPurposes vpu:Marketing)
    ObjectSomeValuesFrom(spl:hasRecipient svr:aCompany)
    ObjectSomeValuesFrom(spl:hasStorage
        ObjectIntersectionOf(
        spl:hasLocation svl:EU
        spl:hasDuration svdu:Indefinitely))
    ObjectSomeValuesFrom(sbpl:hasDuty getValidConsent)
    ObjectSomeValuesFrom(sbpl:hasDuty getAccessReqs)
    ObjectSomeValuesFrom(sbpl:hasDuty getRectifyReqs)
    ObjectSomeValuesFrom(sbpl:hasDuty getDeleteReqs)
    ObjectSomeValuesFrom(sbpl:hasLegalBasis A6-1-a-consent)
    )
```

Listing 1.1: a business policy in SPECIAL

The above attributes for this business policy are described respectively as follows, (i) The category of the personal data used in the processing is *Purchases and Spending Habit*, (ii) the processing category is *analyze*, (iii) The purpose of the processing is *Marketing*, (iv) the recipient of the processing result is *aCompany*, (v) the processing is taking place in a storage located in Europe and the duration of data storage is *Indefinite*, (vi) the duties defined for this processing, for example *getValidConsent* means that the specified software can read the data sources if consent has been given, (vii) the legal basis the processing is *A6-1-a-consent*. The values of these attributes are selected from a suitable vocabulary. SPECIAL uses the W3C's *Data Privacy Vocabularies and Controls Community Group*, (DPVCG) Pandit et al. [13]. We have made use of the same vocabulary for the terms in our framework, which will be presented in Section 3

Consent Representation A consent is represented as a usage policy in SPECIAL and expresses the characteristics of the processing for which the data subject has given his consent. Consent has the same attributes as business policy but, without legal basis and duties. For example, the following policy demonstrates data subject's consent to transfer his *Service Consumption Behavior* data with the purpose *Create Personalized Recommendations*. The recipient is *aCompany*, the storage is located in Europe, and it is valid for a duration of 365 days.

```
ObjectIntersectionOf(
    ObjectSomeValueFrom(has_purpose createPersonalizedRecommendations )
    ObjectSomeValueFrom(has_data serviceConsumptionBehavior)
    ObjectSomeValueFrom(has_processing Transfer)
    ObjectSomeValueFrom(has_recipient aCompany)
    ObjectSomeValueFrom(has_storage
        ObjectIntersectionOf(
            ObjectSomeValueFrom( has_location:EU)
            DataSomeValueFrom( has_duration DatatypeRestriction(
     xsd:integerxsd:minInclusive "365" xsd:integer))))
```

Listing 1.2: A consent in SPECIAL

A business policy is compliant with data subject's consent, if the business process is a subclass of the given consent.

Regulatory Norms In the SPECIAL policy language, the GDPR regulative norms in the form of permissions, obligations, and prohibitions are formalized as the constraints that should hold over the different attributes of a business policy. As an example Article 6-1 (lawfulness) the class `Art6_1` is formalized as follows, it means that a business process holds the obligation of this article if it has a legal basis from the provided list in Article 6-1.

```
ObjectSomeValuesFrom( hasLegalBasis
    ObjectUnionOf(
        Art6_1_a_Consent
        Art6_1_b_Contract
        Art6_1_c_LegalObligation
        Art6_1_d_VitalInterest
        Art6_1_e_PublicInterest
        Art6_1_f_LegitimateInterest))
```

Obligations are formalized as classes and can be combined by operations `ObjectUnionOf` or `ObjectIntersectOf` to form the GDPR obligations at the top level partially. For example, the obligations of Chapter 2 of GDPR (Principles) is

modeled is SPECIAL as a union of the obligations of Article 6 (Lawful processing), Article 9 (Sensitive Data), and Article 10 (Criminal Data).

```
ObjectUnionOf(
    Art6_LawfulProcessing
    Art9_SensitiveData
    Art10_CriminalData)
```

The above expression means the processing is lawful if either the obligations of Article 6 (Lawful Processing), or Article 9 (Sensitive Data), or Article 10 (Criminal Data), are satisfied.

2.2 Event Calculus

Event Calculus (EC) is a logic based formal language for representing events and their effects. It has first been introduced by Kowalski and Sergot [8]. Various versions of the Event Calculus has been used in the literature. In this work, we use a specific version taken from Berreby et al. [3]. It relies on formal representation of events and states on a discrete set of time points. The dynamic state of the world is represented by a set of properties called fluent that hold or not at any time point. Transition between the states is made by events that occur in time T. These events are characterized by preconditions that must hold for the event to occur and effects that describe how they affect states at $T+1$. A fluent holds at T if it was initiated by an event occurrence at $T-1$. A fluent which is true at T continues to hold until the occurrence of an event which terminates it. An event in our framework represents an action which is described in Section 3. The axioms of the event calculus is presented below.

```
negative(neg(F)) :- effect(E,neg(F)).
initiates(E,F,T) :- effect(E,F), occurs(E,T), not negative(F).
terminates(E,F,T) :- effect(E,neg(F)), occurs(E,T),time(T).
clipped(F,T) :- terminates(E,F,T).

holds(F,0) :- initially(F).
holds(F,T) :- initiates(E,F,T-1), time(T).
holds(F,T) :- holds(F,T-1), not clipped(F,T-1), time(T).

:- occurs(E,T), prec(F,E), not holds(F,T), act(E), time(T).
0 {occurs(E, T)} 1 :- act(E), time(T),T<maxtime.
:- occurs(E1,T), occurs(E2,T) ,E1!=E2.
```

Listing 1.3: Event calculus axioms

The choice rule `0 {performs(E, T)} 1 :- act(E), time(T),T<maxtime.` is used to exempt the `perform/2` predicate from minimization in ASP, this is the generator part that we use to solve a planning problem. The rule `:- performs(E1,T), performs(E2,T) ,E1!=E2.` means that events can not occur at the same time.

Note that ASP programs are represented as a finite set of rules where the head and body of the rules are composed of atoms which are equivalent to classical FOL(under the form of clauses with universally quantified variables). Variables in ASP are strings that begin with a capital lette, and they are all universally quantified.

3 Modular Framework

We consider an agent that handles personal data processing. We are concerned about the compliance of the agents' actions with GDPR. In order to model both requirements of the planning domain and compliance checking, we built a modular framework with 2 components. The first one is the planning module that contains the specification of storage, and personal data in the system and the agents' actions. Each action describes a transformation on data or changes its storage. Given an initial state and a goal state, this module generates all the possible plans which satisfy the goal. By plan, we refer to a sequence of processing on personal data. The second module (Compliance engine) checks if each plan is compliant with both regulatory norms and data subject's given consent, and can provide explanation of missing obligations in the case of non-compliance. Figure 1 illustrates the structure of the framework.

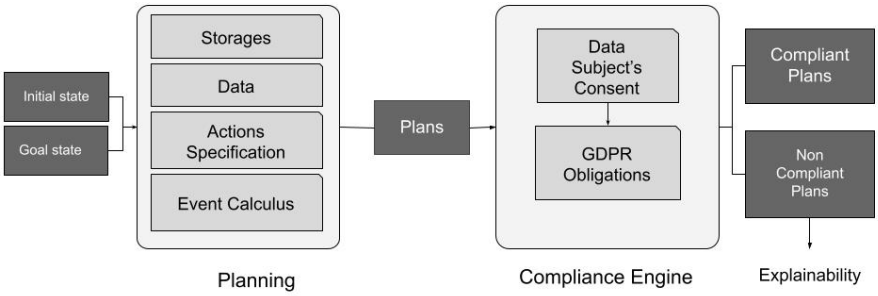

Fig. 1: Modular structure

3.1 A Use Case Model

We describe our framework by implementing it on a use case model. An international company operates in multiple European countries and United States as well. The company has several sectors for providing services to customers. Each sector owns a server for storing personal data. The servers are connected through an internal network and can transmit data among each other. One of the servers is a computing server in which the company analyze customers data for various purposes. The company has also a partner as a data processor which delivers analytic services to the company. Figure 2 illustrates the connection between servers of the company and its partner processor, as well as their location.

In order to provide services, the company needs to analyze customers data and use its result. When a sector requests the outcome of a particular analysis, a sequence of processing should be performed to provide the result to it's corresponding server. We implement our framework in this scenario to design an agent

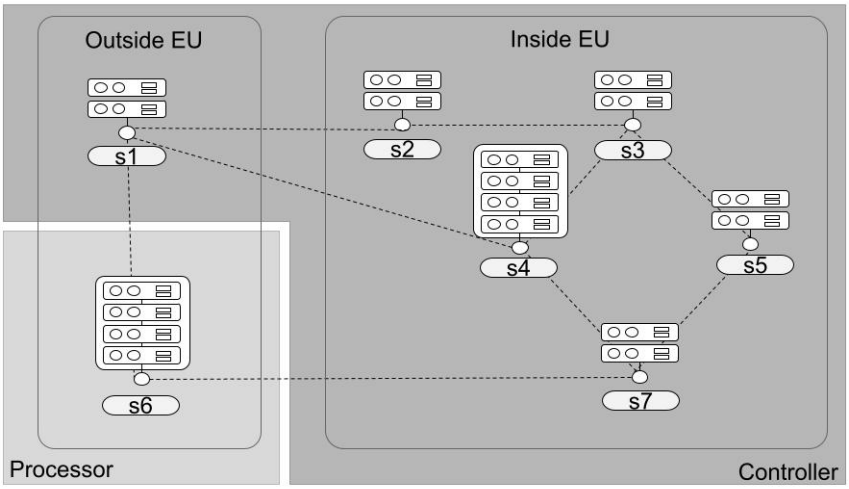

Fig. 2: Connection among servers

to automatically generate a sequence of data processing to provide the requested output on a data, and check for the compliance of generated sequences.

3.2 Planning

The key challenge in the design of the planning domain is formalizing actions, states and domain knowledge in a way that enables for both planning and compliance checking. We assign GDPR related attributes to domain objects to use it for compliance checking. We describe the main parts of the planning domain.

Storage A storage in our framework basically represents anything capable of storing data and processing it. *e.g.* servers, cloud space etc. We use the `storage` to represent a server in our planning domain. We also use `connected/2` predicate to represent that there is a connection between 2 storage. For example, below ASP formalization means that `s1`, `s2`, and `s3` are storage and there is a connection between `s1` and `s2`, and `s1` and `s3`, meaning that they can transmit data.

```
storage(s1).
storage(s2).
storage(s3).
connected(s1,s2).
connected(s1,s3).
```

The knowledge about other servers, and the connection between them, has been formalized the same way. In order to assess the compliance of these actions, we need the information about the action itself as well as the other complementary information that are concerned with the GDPR. For example, the below code, shows how we represent the controller of the storage `s6` and `s7` and the location of `s1` and `s2`.

```
has(s6, controller, aProcessor).
has(s7, controller, aCompany).
has(s1, location, us).
has(s2, location, eu).
```

`s4` and `s6` are computing servers capable of analyzing data for various purposes. The knowledge about supported purposes on each one can be represented in the following way.

```
has(s4, analysisPurpose ,marketing).
has(s4, analysisPurpose ,personalisedAdvertising).
has(s6, analysisPurpose ,optimisationForController).
```

Data A personal data is represented by a `resource`. Each resource belongs to a data subject and has a category. Consider two resources `d1` and `d2` where they have categories *Purchases and Spending Habit* and *Service Consumption Behavior* this is represented in the following way in ASP.

```
resource(d1).
resource(d2).
has(d1, dataCategory, purchasesAndSpendingHabit).
has(d2, dataCategory, serviceConsumptionBehavior).
```

When performing actions on resources, they either move to another storage or transform into a new data. We need to represent these data manipulations in our domain. We define the predicate *data* which represents any personal data either a resource or the output of analysis process on this resource with a certain purpose.

```
data(D):- resource(D).
data( analysisOutput(D,P) ):-  resource(D), purpose(P).
```

If the attributes are static we represent them as facts and if they are dynamic we represent them by fluents. Attributes like storage of the data or the content of a storage are effected by actions.

Actions Specification The agent action in our domain represents a data processing. It supports transfer and analysis processing of personal data with several purposes. A transfer action in our domain is characterized by the data, its current location, the destination and purpose of the transfer. In order to perform analysis action we require the data, and the storage where the processing is taking place and the purpose of the analysis as well. We formalize the knowledge about agents actions as follows:

```
act(transfer(D,A,B,P)):- data(D), storage(A),storage(B), purpose(P),
    connected(A,B), A!=B.
act(analyse(D,A,P)):- data(D), storage(A), purpose(P) , has(A,
    analysisPurpose ,P).
```

Each action should be specified by its preconditions and effects. A transfer action changes the storage of the data, or equivalently, it modifies the content of the origin and destination storage. This is captured by the fluent `hasData(A,D)` predicate, which represents that storage A has data D.The analysis action transforms the personal data into a new data, which is the output of this analysis. Below, we represent the effects and preconditions of these actions.

```
prec( hasData(A,D), transfer(D,A,B,P)):- act(transfer(D,A,B,P)).
effect(transfer(D,A,B,P), hasData(B,D)):- act(transfer(D,A,B,P)).
effect(transfer(D,A,B,P), neg(hasData(A,D))):- act(transfer(D,A,B,P)).

prec( hasData(A,D), analyze(D,A,P)):- act(analyze(D,A,P)).
effect(analyze(D,A,P), hasData( A, analysisOutput(D,P) )):- act(analyze(D,A,P)).
effect(analyze(D,A,P), neg( hasData(A,D) )):- act(analyse(D,A,P)).
```

As an example, we describe the effects and preconditions of analyze action. `prec( hasData(A,D), analyze(D,A,P))` means that in order to perform the action, `analyze (D,A,P)` the fluent `hasData(S,D)` should hold, meaning that the corresponding data should be present on the corresponding storage. When the action is performed, the data transforms into the output and is represented by `analysisOutput(D,P)` . The last line `effect(analyze(D,A,P), neg( hasData(A,D) ))` means that after the action is performed, the output data would be replaced by the input data. The predicate`neg( hasData(A,D) ))` indicates the negative effect of the action, which is the input data in no longer on the corresponding storage (see event calculus axioms in listing 1.3).

3.3 Compliance Engine

This module contains required elements for compliance checking against regulative norms and data subject's consent. For this purpose, it should be fed with legal specifications and formalization of the given consent. Legal specification contains organizational measures, the legal basis of the processing, and the duties defined for processing. The compliance engine has 3 main parts, the first part assigns legal information to actions based on the legal specifications. It then checks for the compliance against regulatory obligations in the next one. In the last part can check the compliance of these actions with the data subject's given consent. Each part is described below.

Actions as Business Policy An action should be associated with the legal information similar to the attributes of a business policy in SPECIAL (see the example of a business policy in listing 1.1). The legal information associated with an action should match with the description of a business policy to be compatible with the underlying policy language. The below example shows how a transfer action is associated with the legal information using the format `has( Action, GDPR_attribute, Value)`. For example, the knowledge about legal basis at line 6 or appropriate safeguards for a personal data transfer at line 7 and 8. Notice that only a fragment of the associated attributes are shown below, you can find the complete list in the code repository[1].

```
has(transfer(D,A,B,P), dataCategory, X) :- act(transfer(D,A,B,P)), has(D,
    dataCategory, X).
has(transfer(D,A,B,P), storage, B):- act(transfer(D,A,B,P)).
has(transfer(D,A,B,P), purpose, P):- act(transfer(D,A,B,P)).
has(transfer(D,A,B,P), recipient, X):- act(transfer(D,A,B,P)), has(B,
    controller, X).
system_legal_basis(art6_1_a_Consent).
```

[1] `https://gitlab.lip6.fr/taheri/planning-compliance-mechanism-policies.git`.

```
has(transfer(D,A,B,P), legalBasis, X):- act(transfer(D,A,B,P)),
    system_legal_basis(X).
transfer_safeguard(s4, s1, art46_2_e_ApprovedCodeOfConduct).
has(transfer(D,A,B,P), measures, X):- act(transfer(D,A,B,P)),
    transfer_safeguard(A,B, X).
```

GDPR Regulatory Obligations We define necessary predicates and axioms to produce a straightforward translation of the GDPR regulatory obligations encoded in SPECIAL policy language and support for explainability in the case of non-compliance. As an example, the obligation at bottom level, Article 6-1(lawful processing) presented in 2.1, has been translated into ASP using the predicate `fulfills/2`. This rule means that an action O fulfills the obligations of article 6-1, if it has a legal basis as defined in the list. Note that in, ASP `pred(a;b)` is equivalent to `pred(a)` and `pred(b)`.

```
art6_1_LegalBasis( art6_1_a_Consent;
                   art6_1_b_Contract;
                   art6_1_c_LegalObligation;
                   art6_1_d_VitalInterest;
                   art6_1_e_PublicInterest;
                   art6_1_f_LegitimateInterest).
fulfills(O, art6_1_LegalBasis):- has(O, legalBasis, X), art6_1_LegalBasis(X),
     act(O).
```

Each regulation is named after its reference in the GDPR text, part of the current supported obligations in this module are presented as by the predicate `regulation`. Where `gdpr_Requirements` represents the obligations at the top level.

```
regulation(art6_1_LegalBasis;
    art6_lawfulProcessing;
    art12_22_SubjectRights;
    chap3_RightsOfDataSubjects;
    chap2_LawfulProcessing;
    art9_sensitiveData;
    gdpr_Requirements).
```

In the SPECIAL policy language, the obligation at the bottom level are nested to form the top level obligation. The regulations are combined using the operators `ObjectUnionOf` and `ObjectIntersectOf`. We capture the same semantic by the predicates `inUniounOf` and `inIntersectOf`. For example, the listing 2.1 is translated as follows

```
inUnionOf(art6_LawfulProcessing ,chap2_LawfulProcessing ).
inUnionOf(art9_SensitiveData, chap2_LawfulProcessing ).
inUnionOf(art10_CriminalData, chap2_LawfulProcessing ).
```

The predicate `inUnionOf/2` is defined as below. It means that if an action P fulfills the obligation set of $R2$, and $R2$ is in union set of $R1$ then, it also fulfills the obligation of $R1$.

```
fulfills(P,R1):- fulfills(P,R2), inUnionOf(R2,R1), act(P).
```

An action is compliant if it fulfills all obligations of the fraction of the GDPR at the top level. A plan contains several actions, and it is possible that only a certain operation violates the compliance of the plan. In this case we are interested to know which missing obligation caused the non-compliance, in order to do so we use the predicate `missing/2` in the following ASP rule, it indicates that the obligations of a certain article are missed.

```
missing1(P,R,R):- not fulfills(P,R),regulation(R), act(P), occurs(P,_).
missing1(P,R1,R2):- not fulfills(P,R2),  upperClass(R3,R2),missing1(P,R1,R3),
      regulation(R1), regulation(R2).
missing(P,R):- missing1(P,R,gdpr_Requirements), not auxiliaryRegulation(R).
```

Data Subject's Consent Suppose that when collecting personal data, the user has given explicit consent for transferring his *Purchases and Spending Habit* data for the purpose of *marketing*. Based on this consent, the data can only be disclosed to *aCompnay*, and it should be stored only in Europe. We translate this consent using the same format as listing 1.2. Note that SPECIAL also supports time intervals for the validity of the consent, but we do not support it here. In our scenario, the data subject has also given his consent to analyze processing with same attributes.

```
has(c2, dataCategory, serviceConsumptionBehavior).
has(c2, processing, transfer).
has(c2, purpose, marketing).
has(c2, recipient, aCompany).
has(c2, storageLocation, eu).
```

In our modeling, a processing is compliant with the given consent if it has the same attributes as the action. We check for the compliance of an action with the given consent using the following set of rules. It basically states that valid consent of an operation is satisfied if there is a coherent consent for it; and an action is coherent with a consent if there is no difference between the attributes of the consent and the operation. We capture it by the predicate `validConsentSatisfied` that is true when there is coherent consent for it.

```
non_coherent(P,C):- has(P,A, Z1) , has(C, A, Z2) ,Z1!=Z2, act(P), consent(C).
validConsentSatisfied(P):- not non_coherent(P,C), act(P), consent(C).
```

4 Evaluation

Once we have modeled our domain knowledge, the planning module can be used to generate plans by providing an initial state and a goal state. A plan is generated to deliver the result of processing of personal data to the server asking for it. Consider that data $d1$ is initially stored in the server $s1$. We represent this initial state by `initially( hasData(s1,d1))`.

There is a request from server $s4$ for the results of the analysis on data $d1$ with the purpose *marketing*. We represent this request by `requestAnalysis(s4, d7 , marketing)`. This request is then translated into a goal for the system that the output of this analysis should be stored on the storage asking for it.

```
holds(goal,T):- holds( hasData(A, analysisOutput(D,P)), T ), requestAnalysis(
    A,D,P).
:- not holds(goal, maxtime).
```

After providing the initial state and a goal state, all the possible plans are generated to satisfy the given request. Plans are included in Table 1. Note that all these sequence of actions are generated regardless of their compliance. Each

Plan	Time Step	Actions
1	1	transfer(d1,s2,s3,marketing)
	2	transfer(d1,s3,s4,marketing)
	3	analyse(d1,s4,marketing)
	4	transfer(analyseOut(d1,marketing),s4,s7,marketing)
	5	transfer(analyseOut(d1,marketing),s7,s5,marketing)
2	1	transfer(d1,s2,s1,marketing)
	2	transfer(d1,s1,s4,marketing)
	3	analyse(d1,s4,marketing)
	4	transfer(analyseOut(d1,marketing),s4,s7,marketing)
	5	transfer(analyseOut(d1,marketing),s7,s5,marketing)
3	1	transfer(d1, s2, s3, marketing)
	2	transfer(d1, s3, s4, marketing)
	3	analyse(d1, s4, marketing)
	4	transfer(analyseOut(d1, marketing), s4, s3, marketing)
	5	transfer(analyseOut(d7, marketing), s3, s5, marketing)
4	1	transfer(d1,s2,s1,marketing)
	2	transfer(d1,s1,s4,marketing)
	3	analyse(d1,s4,marketing)
	4	transfer(analyseOut(d1,marketing),s4,s3,marketing)
	5	transfer(analyseOut(d1,marketing),s3,s5,marketing)

Table 1: Automatic generated plans.

plan is a set of actions presented by the predicate `perform/2` which indicates the action and the time step in which it can be performed.

Having the plans generated by the previous module, the compliance engine can distinguish the compliant plans with the non-compliant ones and also provide a simple explanation for non-compliance by referring to the missing obligations. A plan is compliant if all the actions in that plan are compliant. Below we show the compliance checking result in two scenarios; Compliance checking with (i) data subject's given consent and (ii) GDPR regulatory norms. In both cases the initial state and the goal state are the same and the same plans are generated by the planning module (shown in Table 1), therefore the compliance engine assess the compliance of the identical plans but with different legal restrictions.

Consent compliance checking In this scenario the customer has given a customized set of consent for various data processing. In particular we suppose that the data subject has given her consent only for internal transfers in EU so the compliance engine distinguish non-compliant plans if they are compatible with data subject's given consent. Table 2 presents the compliance of each plan as well as the explanation of the missing obligations.

Plan	Compliance	Explanation
1 and 3	Yes	-
2 and 4	No	missing(transfer(d1,s2,s1,marketing), art12_22_SubjectRights,) missing(transfer(d1,s2,s1,marketing), chap3_RightsOfDataSubjects) missing(transfer(d1,s2,s1,marketing), exceptions_as_per_Art23) missing(transfer(d1,s2,s1,marketing), chap9_Derogations) missing(transfer(d1,s2,s1,marketing), gdpr_Requirements)

Table 2: Automatic compliance checking of plans (Consent).

All the plans are generated automatically by the personal data managing agent. The process of compliance checking is also done automatically in the second module. Plan 1 and 3 are compliant since they fulfill all the obligation set forth of GDPR as well as the compliance with consent. Plan 2 and 4 are both non-compliant because of the same reason. The action `transfer(d1,s1,s3,marketing)` lacks the obligation of `art12_22_SubjectRights` since the transfer action does not match with the provided consent. When the obligations of a regulation are missed, it also causes that the obligations of the super class regulations to fail. In this case the action `transfer(d1,s2,s1,marketing)` misses the obligations of `chap3_RightsOfDataSubjects` , and `gdpr_Requirements` as they are the top classes of regulations in the policy formalization. Two other regulations have been reported as missed obligations, `exceptions_as_per_Art23` and `chap9_Derogations`, this is because if the obligations of these regulations is fulfilled, it causes the transfer action to comply with GDPR.

Compliance checking against GDPR regulatory norms In this scenario we suppose that all the necessary consent is provided, so consent is no more a restricting constraint. We check the compliance of plans against GDPR regulatory norms, in particular obligations of GDPR Chapter 5 (Transfers of personal data to third countries or international organisations).

According to SPECIAL policy language[2] a transfer to a third country is only possible if it is not among the unauthorized transfers by Union law Article 48 (Transfers or disclosures not authorised by Union law) and is equipped with measures to assure a secure data transfer; these measures could be one of the appropriate safeguards as in Article 46 (Transfers subject to appropriate safeguards). Again we aim at checking the compliance of plans shown in Table 1, with the assumption that all the necessary consent is provided but no safety measures exist among the servers outside the EU (third countries) and the servers located in EU. The resulting compliance report is indicated in Table 3

[2] link to the documentations: `https://specialprivacy.ercim.eu/platform/pilots-policies-and-the-formalization-of-the-gdpr`.

Plan	Compliance	Explanation
1 and 3	Yes	-
2 and 4	No	missing(transfer(d1,s2,s1,marketing),chap5_DataTransferToThirdCountry) missing(transfer(d1,s2,s1,marketing),adequateLevelOfProtection_as_per_Art45) missing(transfer(d1,s2,s1,marketing),appropriateSafeguards_as_per_Art46) missing(transfer(d1,s2,s1,marketing),art49_Derogations) missing(transfer(d1,s2,s1,marketing),chap9_Derogations) missing(transfer(d1,s2,s1,marketing), gdpr_Requirements)

Table 3: Automatic compliance checking of plans (obligations of GDPR chapter 5)

As shown in Table 3 plan 2 and 4 are not compliant, since the action `transfer(d1,s2,s1,marketing)` miss the required obligations of GDPR Chapter 5. The principal missed elements are `adequateLevelOfProtection_as_per_Art45`, `appropriateSafeguards_as_per_Art46` or `art49_Derogations`. The transfer action could be compliant if certain regulations among missed ones are satisfied.

5 Related works

Since the adoption of the GDPR, several tools and techniques have been introduced to facilitate the compliance assessment for data controllers and processors. Some of these methods are in the form of a questionnaire which evaluates the compliance of data processor and controllers *e.g.* Microsoft Trust Center, Agarwal et al. [1], but these methods do not support automated compliance checking.

Others focus on building an ontological concept of GDPR, *e.g.* PrOnto a privacy ontology Palmirani et al. [12] which relays on LegalRuleML Palmirani et al. [11] for legal reasoning and compliance checking of business processes. These methods can be used to make a repository of rules based on regulative and constitutive norms (cf. Robaldo et al. [14]). However, they provide a machine-readable representation of GDPR norms suitable for legal reasoning. One of our aims as future work is to integrate use a comprehensive ontology of GDPR norms with a legal reasoning engine.

Another body of works develop policy languages to enable the compliance checking of business processes De Vos et al. [6], Bonatti et al. [4]. The latter is based on the W3C Data Privacy Vocabulary and Controls Community Group (DPVCG) Pandit et al. [13] that is a vocabulary towards interoperability in the context of data privacy. De Vos et al. [6] introduce an ODRL policy pipeline to represent GDPR requirements and business policies. They then translate these representations in answer set programming and use it to check for the compliance of a business policy with GDPR regulatory obligations.

A number of works focus on integrating policies in a dynamic environment. Bandara et al. [2] present a method for transforming both, policy and system behavior specifications into a formal notation that is based on Event Calculus.

However, they use it for detecting conflicts in the system. In the privacy domain, Le Métayer and Rauzy [9] propose a formal framework to specify the notion of control over personal data and to reason about it, but it does not support compliance mechanism with any data protection regulation.

The methodology by De Vos et al. [5] uses a domain-specific language called *InstAL* based on ASP which is fairly similar to event calculus. They use *InstAL* to model events and time-varying properties in the system but not to generate sequences of actions to satisfy a given goal. Instead, they provide the program with the performed actions and are mainly interested in the evolution of the governing norms when an action takes place by an agent. However, in our current work norms are assumed constant and we use them to distinguish compliant and non-compliant plans. Adding support for time-varying norms is considered as a future extension to the current work.

6 Conclusion and future work

The goal in the current paper was two-fold (i) designing an AI agent that generates sequence of data processing or plans in order to satisfy a given goal, (ii) compliance checking of these plans with GDPR. In order to fulfill these goals, we presented a framework for planning in the privacy domain and compliance checking with GDPR. We made use of event calculus to formalize agent actions on personal data, and time-varying properties of the system. In order to formalize GDPR obligation set and data subjects consent, we chose SPECIAL policy language. We described how the knowledge of the planning domain and legal knowledge can be represented and combined together in our framework. We presented two scenarios in Section 4 and showed how our framework can be used to achieve the mentioned goals.

Our future goal is to design an AI agent with real-time legal and ethical supervisors concerning personal data protection. This work is highly dependent on the legal ontology or the policy language that we use to represent GDPR requirements. In the current paper we use SPECIAL as the underlying policy language, which is a simple machine-readable policy that does not support deontic operations.

Our ongoing work includes using a more comprehensive legal ontology like, PrOnto Palmirani et al. [12] or other policy languages like ODRL based one by De Vos et al. [6], and cover more GDPR articles in the design of the system. The current work assess the compliance in a static manner, i.e., it does not consider the evolution of norms in real-time as in De Vos et al. [5], adding support for more complex legal models is on other aspect of this work which we are trying to improve.

Other future works include developing our framework to implement more complex data processing scenarios cable to handle specific ethical issues concerning data protection an adding support for deontic operators and handle conflicts in real-time.

References

[1] S. Agarwal, S. Steyskal, F. Antunovic, and S. Kirrane. Legislative compliance assessment: framework, model and gdpr instantiation. In *Annual Privacy Forum*, pages 131–149. Springer, 2018.

[2] A. K. Bandara, E. C. Lupu, and A. Russo. Using event calculus to formalise policy specification and analysis. In *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pages 26–39. IEEE, 2003.

[3] F. Berreby, G. Bourgne, and J.-G. Ganascia. A declarative modular framework for representing and applying ethical principles. In *16th Conference on Autonomous Agents and MultiAgent Systems*, 2017.

[4] P. A. Bonatti, S. Kirrane, I. M. Petrova, and L. Sauro. Machine understandable policies and gdpr compliance checking. *KI-Künstliche Intelligenz*, 34 (3):303–315, 2020.

[5] M. De Vos, J. Padget, and K. Satoh. Legal modelling and reasoning using institutions. In *JSAI International Symposium on Artificial Intelligence*, pages 129–140. Springer, 2010.

[6] M. De Vos, S. Kirrane, J. Padget, and K. Satoh. Odrl policy modelling and compliance checking. In *International Joint Conference on Rules and Reasoning*, pages 36–51. Springer, 2019.

[7] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. Clingo= asp+ control: Preliminary report. *arXiv preprint arXiv:1405.3694*, 2014.

[8] R. Kowalski and M. Sergot. A logic-based calculus of events. In *Foundations of knowledge base management*, pages 23–55. Springer, 1989.

[9] D. Le Métayer and P. Rauzy. Capacity: an abstract model of control over personal data. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pages 64–75, 2018.

[10] V. Lifschitz. *Answer set programming*. Springer Heidelberg, 2019.

[11] M. Palmirani, G. Governatori, A. Rotolo, S. Tabet, H. Boley, and A. Paschke. Legalruleml: Xml-based rules and norms. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 298–312. Springer, 2011.

[12] M. Palmirani, M. Martoni, A. Rossi, C. Bartolini, and L. Robaldo. Legal ontology for modelling gdpr concepts and norms. In *Legal Knowledge and Information Systems*, pages 91–100. IOS Press, 2018.

[13] H. J. Pandit, A. Polleres, B. Bos, R. Brennan, B. Bruegger, F. J. Ekaputra, J. D. Fernández, R. G. Hamed, E. Kiesling, M. Lizar, et al. Creating a vocabulary for data privacy. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 714–730. Springer, 2019.

[14] L. Robaldo, C. Bartolini, M. Palmirani, A. Rossi, M. Martoni, and G. Lenzini. Formalizing gdpr provisions in reified i/o logic: the dapreco knowledge base. *Journal of Logic, Language and Information*, 29(4):401–449, 2020.

[15] M. Shanahan. The event calculus explained. In *Artificial intelligence today*, pages 409–430. Springer, 1999.