Multi-agent Online Planning Architecture for Real-time Compliance

Hisashi Hayashi, Theodoros Mitsikas, Yousef Sojasi Taheri, Kanae Tsushima, Ralph Schäfermeier, Gauvain Bourgne, Jean-Gabriel Ganascia, Adrian Paschke, Ken Satoh

Multi-agent Online Planning Architecture for Real-time Compliance

Hisashi Hayashi[1,*,†], Theodoros Mitsikas[2,3,*,†], Yousef Taheri[4,*,†], Kanae Tsushima[5,*,†], Ralph Schäfermeier[6], Gauvain Bourgne[4], Jean-Gabriel Ganascia[4], Adrian Paschke[3,7] and Ken Satoh[5,8,*]

[1]*Advanced Institute of Industrial Technology, Tokyo, Japan*
[2]*National Technical University of Athens, Zografou, Greece*
[3]*Institut für Angewandte Informatik, Leipzig, Germany*
[4]*Sorbonne University, Paris, France*
[5]*National Institute of Informatics, Tokyo, Japan*
[6]*Leipzig University, Leipzig, Germany*
[7]*Freie Universität Berlin and Fraunhofer FOKUS, Berlin, Germany*
[8]*SOKENDAI, Tokyo, Japan*

Abstract

Due to the recent rapid introduction of AI technologies into society, we face new risks related to AI. Therefore, it is very important to let AI be compliant with legal and ethical norms to reduce such risks. In this paper, we propose a method of compliance mechanism of AI agent planning in a multi-agent setting. We formalize legal norms as hard constraints which must be satisfied and ethical norms as soft constraints which should be satisfied as much as possible. Moreover, our mechanism can handle real-time exogenous environmental change by a replanning mechanism. We believe that our architecture is an important step towards "trustworthy AI".

Keywords

real-time compliance, online HTN planning, legal compliance, legal checking, ethical compliance, ethical checking, Prova, multi-agent system

RuleML+RR'23: 17th International Rule Challenge and 7th Doctoral Consortium, September 18–20, 2023, Oslo, Norway
*Corresponding authors.
†These authors contributed equally.
hayashi-hisashi@aiit.ac.jp (H. Hayashi); mitsikas@central.ntua.gr (T. Mitsikas); yousef.taheri@lip6.fr (Y. Taheri); k_tsushima@nii.ac.jp (K. Tsushima); ralph.schafermeier@gmail.com (R. Schäfermeier); gauvain.bourgne@lip6.fr (G. Bourgne); jean-gabriel.ganascia@lip6.fr (J. Ganascia); adrian.paschke@fokus.fraunhofer.de (A. Paschke); ksatoh@nii.ac.jp (K. Satoh)
https://researchmap.jp/hayashi-hisashi (H. Hayashi); https://lip6.fr/Yousef.Taheri (Y. Taheri); https://researchmap.jp/tsushima (K. Tsushima); https://www.data-analytics-center.org/team-0886abf92d9d2018 (R. Schäfermeier); https://lip6.fr/Gauvain.Bourgne (G. Bourgne); https://lip6.fr/Jean-Gabriel.Ganascia (J. Ganascia); https://www.mi.fu-berlin.de/inf/groups/ag-csw/Members/members/paschke.html (A. Paschke); http://research.nii.ac.jp/~ksatoh/ (K. Satoh)
0000-0003-2134-8420 (H. Hayashi); 0000-0002-7570-3603 (T. Mitsikas); 0000-0002-4349-6726 (R. Schäfermeier); 0000-0003-3156-9040 (A. Paschke)

1. Introduction

With the rapid evolution and spread of AI technologies in society, we can receive many benefits from AI. However, these same technologies also introduce new risks and negative consequences for individuals or society, that threaten legal and ethical principles. Thus, we need to ensure AI to be compliant with these principles and this is a central concern that has become prominent both in public opinion and policy maker's agenda. In EU, there has been a proposal of "AI ACT" [1] to ban AI systems that have an unacceptably high risk to create a clear threat to society, livelihoods, and rights of people and strongly regulate AI systems that have a high risk to be used in critical infrastructures and systems influencing human rights. Other AI systems are not regulated by this ACT but AI systems in general should be trustworthy [2], which means they should be lawful (respecting all applicable laws and regulations), ethical (adhering to ethical principles and values), and robust (both technically and considering their social environment). Therefore, technical solutions are needed to achieve this goal, and it is strongly believed/we strongly believe that mechanisms addressing these issues must/should/have to be embedded at the core of AI agent architectures.

As far as we know, there has been no research combining legal and ethical compliance in AI systems. However, there are various researches for each legal compliance and ethical compliance. In legal compliance, there are various researches such as modal (deontic) logics [3, 4], natural language processing [5] and logic programming [6]. However, the compliance check of these researches is off-line; in other words, given a specification of an AI agent or an execution trace of an AI agent, the compliance mechanism checks inconsistency between norms and a specification of an AI agent. Computational ethics is a field concerned with computational models of ethical principles. Various models of ethical decision processes have been proposed depending on the ethical principles being modeled and the expressivity of the representation language. Recent examples include [7, 8, 9].

In this paper, we propose an overall architecture that combines legal and ethical compliance mechanisms to ensure the above conditions of trustworthy AI in the context of AI agent planning in multi-agent systems. As a use case, we consider the management of personal data. In the domain of the use case, concerns about privacy and protection of personal data have received a lot of attention not only from a philosophical and ethical side (see for instance the aforementioned European guidelines) but also from a legal perspective, with European initiatives such as the GDPR which put companies under close scrutiny of their policy on the subject.

In our framework, we regard legal rules as "hard constraints" which AI must follow and ethical rules as "soft constraints" which play a role to choose the best action sequences among possible action sequences and we have a real-time compliance mechanism to check compliance even if there is a change of environment during the execution of a plan and we can modify the plan to be compliant with the new environment.

The remainder of this paper is organized as follows. In Section 2, we explain the overview of the proposed architecture. In Section 3 we give a use case used for an explanation of our architecture. In Sections 4, 5, 6 and 7, we propose main components for AI compliance mechanism; planning agent, legal checker, ethical checker, and action executor. Section 8 concludes the paper.

2. Architecture overview

Figure 1 shows the overall architecture around the planning agent. The planning agent conducts planning, replanning, and execution. The legal checker evaluates a plan based on legal norms and judges whether the plan is legal or illegal. The ethical checker evaluates and compares multiple plans based on ethical norms and selects the most ethical plan. The action executor executes the action and reports the result according to the request from the planning agent.

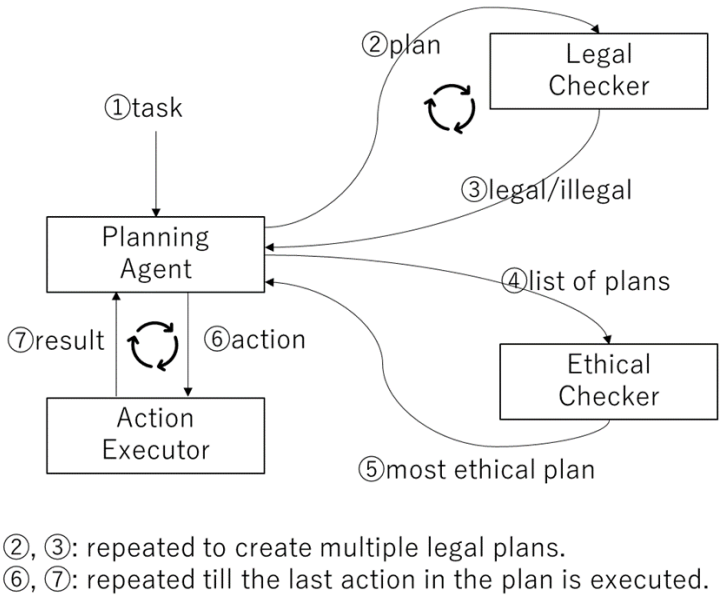

Figure 1: Planning and execution

Given a task (Step 1), the planning agent makes the least-cost plan and sends it to the legal checker (Step 2). The legal checker judges whether the plan is legal or illegal and reports the result to the planning agent (Step 3). Then, the planning agent constructs the second least-cost plan and sends it to the legal checker (Step 2 of the second loop) for legal checking. This process (repetition of Steps 2-3) continues till the predefined number of legal plans is obtained or no more plan exists. After constructing multiple low-cost legal plans, the planning agent sends the plans to the ethical checker (Step 4). The ethical checker selects the most ethical plan and reports it to the planning agent (Step 5). At this point, the planning agent commits to the selected plan, which is legal and most ethical, and executes each action in the plan one by one calling the action executor (Step 6). The action executor executes the action specified by the planning agent and reports the result to the planning agent (Step 7). The planning agent updates the belief and plans based on the execution result of the action. When the action executor reports new observations to the planning agent, the current plan might become invalid or less attractive in terms of cost. In this case, replanning is triggered. When replanning, the planning agent also calls the legal checker and ethical checker in the same way as in initial planning (Steps 2-5).

The planning agent keeps and modifies all the alternative plans including the plans which were not checked by legal and ethical checkers according to the most recent information that is reported from the action executor. When the situation changes in the middle of action execution, the current plan might become invalid. The planning agent can notice this because it checks the "protected links" that record the fluents that has to be true between two actions in the plan. In this case, the planning agent replans using legal and ethical checkers in the same way as initial planning.

In our use case, we consider the international data transfer scenario, where the planning agent has to execute such actions as data transfer between different servers in different countries. In this use case, the action executor is a multi-agent system where the data is transferred from one server agent to another. The message passing between the server agents plays an important role in coordinating this multi-agent system. We will later explain this in Section 7.

In our previous research [10], legal norms of data transfer were expressed in the precondition of tasks, and ethical norms were expressed as costs of tasks. However, in this research, we clearly separated the planning agent, the legal checker, and the ethical checker as different modules because each module requires a specialized knowledge base which should be developed by different specialists.

3. Use case

An employment platform company provides job recommendation services for users. The company transmits personal data through its distributed servers to generate recommendations and deliver the result to users. On one hand, certain legal obligations from data protection regulations, in particular the European GDPR, apply to processing and transferring personal data. On the other hand, such use of personal data raises some ethical concerns about the privacy of users, the safety of personal data transfers, bias in the recommendations output, etc., that need to be respected. The company is using an automated process to handle data among its servers and wishes to ensure that legal obligations are met and ethical criteria are respected as much as possible. As a use case, we model a simplified version of this process for a single user on the company's distributed servers. We show how our planning architecture with legal and ethical checking can be applied in this case. See Figure 2.

Nodes There are four nodes that represent parts of the company's distributed servers. Node 2 is outside the EU, and the other nodes are inside the EU. The region of a node indicates above all the legislative zones, as processing and transferring personal data to other regions may pose legal and ethical issues and therefore needs to be checked. The safety levels of the nodes are also used for ethical verification.

Processing Two types of data processing are available for recommendation: process 1 (p1) and process 2 (p2). These data processors exist at the processing node.

Personal data The company uses personal data to generate job recommendations for users. The data to be used are essential for legal and ethical evaluations in our architecture. Two data

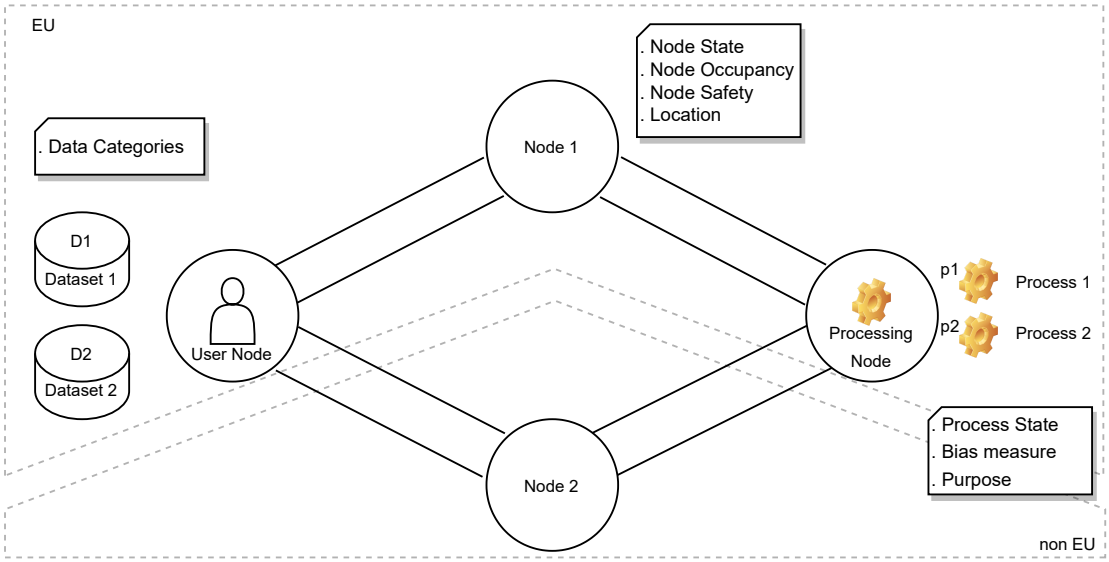

Figure 2: Use case: job recommendation platform

sets (D1 and D2) are available at the user node for the purpose of recommendation.

4. Planning agent

In this section, the architecture of the planning agent is explained in more detail. Figure 3 shows the system architecture of the planning agent. The planning agent conducts planning, replanning, and execution. Plans are made by the online HTN planner. In addition, the planning agent uses two external modules: the legal checker and the ethical checker. The legal checker checks whether a plan is legal or illegal. The ethical checker compares multiple legal plans and selects the most ethical plan.

The agent controller of the planning agent integrates the online HTN planner, the legal checker, and the ethical checker to produce a legal and ethical plan and requests the plan executor to execute the plan.

The online HTN planner uses an online forward-chaining total-order HTN planning algorithm of Dynagent [11]. Like SHOP [12], which is a standard HTN planner, it makes plans by task decomposition where a plan is a sequence of tasks (abstract tasks and actions). A plan is executable when each task is an action (= a primitive task). In this algorithm, given a task, plans (= sequences of tasks) are constructed by repeatedly decomposing an abstract task in a plan into a sequence of subtasks in the order of execution till a plan becomes executable.

It uses the cost information of each task and conducts the best-first search, which finds the least-cost plan. The information which is used for planning is called belief. Belief includes facts, preconditions of tasks, effects of actions, costs of tasks, and task-decomposition rules (called methods in SHOP).

Unlike SHOP, the algorithm of Dynagent is an online planning algorithm. It keeps and

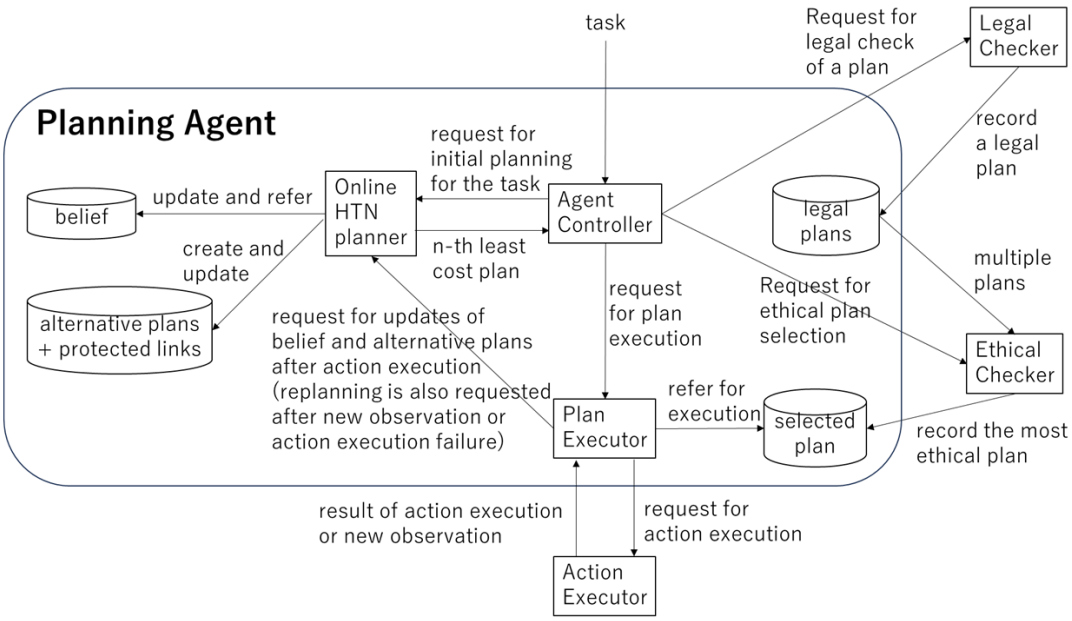

Figure 3: Planning agent

incrementally modifies the alternative plans. It also maintains such supplementary information as preconditions of tasks and effects of actions in association with each plan. The preconditions of a task must be satisfied just before the execution of the task. A fluent (= a predicate whose truth value changes) becomes true or false after an action execution, which is called an effect of the action. When a precondition of an action depends on an effect of another preceding action, this information is recorded as a protected link to check the validity of the plan. When the situation changes in the middle of plan execution, if a protected link is broken, the plan becomes invalid. Even if a precondition of a task in a plan was not satisfied in initial planning, the planner keeps this plan because the plan might become valid in future when the precondition is satisfied. In summary, this online algorithm deletes invalid plans and adds new valid plans adapting to the changing world. The plan is changed when the current plan becomes invalid or the current plan becomes less attractive in terms of costs.

Based on the current plan, the plan executor requests the external action executor to execute the next action in the plan. The action executor reports the result of the action execution or new observation to the plan executor, and the plan executor requests the online HTN planner to update the belief and the alternative plans.

Each time an action is successfully executed, the belief is updated based on the effects of the action, the action is removed from the head of each plan, the alternative plans which become invalid are removed, and the new valid plans are added to the alternative plans. If the action execution is not successful, the current plan is not executable. In this case, all the plans whose head is the same action are not executable, and they are removed from the alternative plans. When the situation changes unexpectedly, the action executor might report new observation to the action executor. In this case, the belief is updated and plans are checked according to

the new belief using the protected links recorded in the plans, which might lead to replanning. After replanning, the new plans are checked by the legal checker and the ethical checker as in initial planning, the execution of the newly selected plan is resumed soon afterward.

4.1. Planning for the use case

The objective of this task is to deliver job recommendations to the user by using their personal data which is stored in their assigned node. The planning agent generates all possible plans. These plans will be later evaluated by the legal and ethical checkers to select the legal and the most ethical plan. The belief of the planning agent can be constructed in the same way as in [10], which is omitted in this paper because of limited space. However, note that legal and ethical checkers are constructed as separate modules in this paper, which is different from our previous study [10].

Initial planning The planning agent decomposes the objective task to generate plans or sequences of actions. There are four choices to be made in the planning process, and each one has two possible options according to our use case. i) The data set (*D1* or *D2*), ii) the intermediary node for transmitting data to the process node (*node1* or *node2*), iii) the processing to be applied to the chosen data set (*p1* or *p2*), iv) the intermediary node for delivering the process output (*node1* or *node2*). This will lead to a total of 16 possible plans. These choices of plans will be checked by the legal checker and only one plan will be selected by the ethical checker.

An example of the plan (plan 1) for recommendation is as follows:

```
1- transfer D1 from the user node to node1
2- transfer D1 from node1 to the processing node
3- run the process p1 using D1 at the processing node
4- transfer the process output from the processing node to node1
5- transfer the process output from node1 to the user node
```

Online replanning Suppose that plan 1 was selected for execution, and actions 1, 2, and 3 in plan 1 have been executed. We have the process output at the processing node at this point. Now suppose that node1 becomes tentatively unavailable. In that case, the planning agent replans and produces plans that avoid node 1. Afterward, legal and ethical checks will be conducted in the same way as initial planning. An example of the plan for recommendation after replanning is as follows:

```
1- transfer the process output from the processing node to node2
2- transfer the process output from node2 to the user node
```

In HTN planning, a task is decomposed into subtasks. An example of a task-decomposition rule is shown below. This rule means that the task (`move`) of multi-hop data transfer from `NodeOrig` to `NodeDest` is decomposed into two primitive transfer actions (`transfer`) via NodeMiddle.

```
htn(move(Dataset, NodeOrig, NodeMiddle, NodeDest, Purpose) ,[],[
    transfer(Dataset, NodeOrig, NodeMiddle, Purpose),
    transfer(Dataset, NodeMiddle, NodeDest, Purpose)]).
```

The preconditions and effects of an action are expressed by a rule as usual. The following rule means that by executing the action `transfer`, the location of `Dataset` changes from `NodeOrig` to `NodeDest` if `Dataset` is at `NodeOrig`, `NodeDest` is directly connected from `NodeOrig` by a link, and `NodeDest` is active.

```
action(transfer(Dataset, NodeOrig, NodeDest, _),[
   dataSetAt(Dataset, NodeOrig),
   connected(NodeOrig, NodeDest),
   nodeState(NodeDest, active)
],[
   initiates(dataSetAt(Dataset, NodeDest)),
   terminates(dataSetAt(Dataset, NodeOrig))]).
```

The planning agent uses many other rules in this use case. However, we do not show them because of limited space.

5. Legal checker

The role of the legal checker is to check whether a given plan by the planning agent is legal. Because a plan is a list of actions, the legal checker checks all the actions and only decides that the plan is legal if all the actions are legal. In this section, we will look at how the legal checker checks in the context of GDPR cases.

First, the legal checker obtains actions from the plan received from the planning agent. For example, consider the following plan, which transfers D1 through node 1 and processes it for recommendation: [transfer(user_node, node1, recommendation, D1), transfer(node1, processing_node, recommendation, D1), processing(p1, D1)].

The execution flow is shown in Figure 4. First, (1) the legal checker database is accessed to check whether the data in the action is personal data. If it is not personal data, it is not covered by the GDPR and is therefore legal. Next, (2) checks whether the current action is within the scope of the GDPR. If it is not GDPR-applicable, then the result is legal, as no GDPR decision is required[1]. If it is GDPR-applicable, (3) check whether the action being performed is a transfer. This is because the next step depends on this. If the action is a transfer, (4) checks whether the country or region after the transfer meets the conditions of the GDPR and there is consent from the data subject for the relevant purposes. If the conditions are satisfied, the result is legal. Otherwise, it is illegal. In (3), if the action is not a transfer, check (5) whether there is consent.

Let us see what happens to the previous plan. First, check the first action: transfer(user_node, node1, recommendation, D1). Suppose we start the evaluation with the contents of the legal database in Table 1. The first table contains data set information: the name of the data, the permissions obtained, whether it contains personal data, and whether the GDPR applies. The second table contains process information: the name of the process, the purpose of the process,

[1]This is because we do not consider other laws now. If we treat other laws, we need to modify this part.

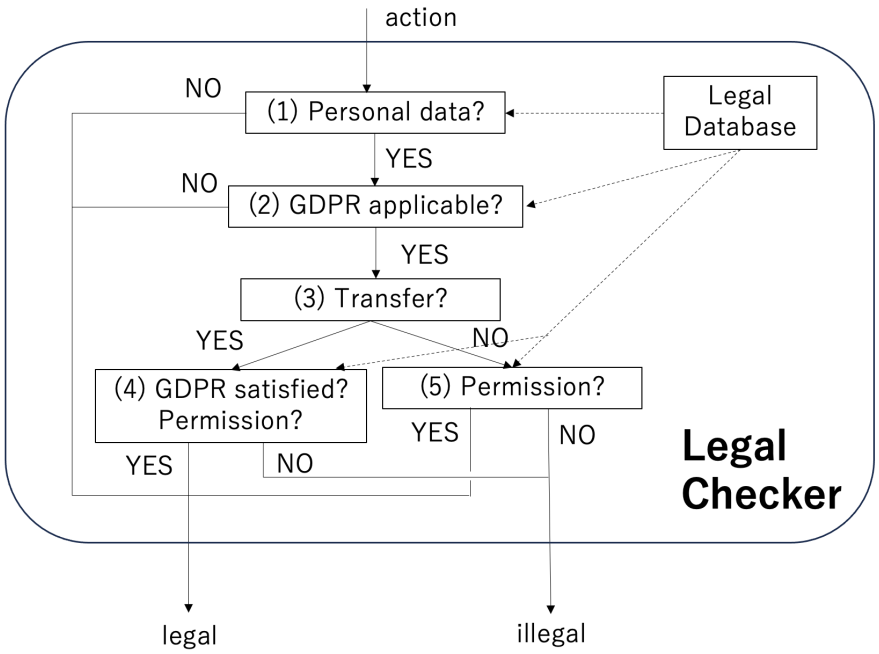

Figure 4: Legal checker for GDPR

dataset name	*permission*	*personal?*	*GDPR data?*
$D1$	$recommendation(=R), transfer\ for\ R$	YES	YES
$D2$	$analysis(=A), transfer\ for\ A$	YES	YES

process name	*purpose*	*location*
$p1$	$recommendation$	$in\ EU$
$p2$	$analysis$	$in\ EU$

Table 1
Legal Databases for data set and process information

and location information. Although we omit the table of the location information of nodes, it is also stored like the second table.

- In (1) go to YES because D1 is stored in the database as personal data.
- In (2), go to YES because the data is being transferred from user_node, which is in the EU. And D1 is data that should be protected by the GDPR, as can be seen from the legal database.
- In (3), go to YES because this action is a transfer.
- In (4), go to YES because the data is transferred to node 1 which is in the EU and D1 has permission to transfer for the recommendation.

- The final result of this action is legal.

The following is part of the Proleg [13] code that evaluates the above example[2].

```
legal(transfer(_data, _from_country, _to_country, _purpose))
  <= personal(_data),
     gdpr_applicable(_from_country),
     with_consent_of_the_transfer_purpose(_data, _purpose).
```

As the first action, the second and third actions are also legal. Since all the actions are legal, the plan that includes these three actions is also legal. If we consider the rewriting of D1 to D2 in the previous plan, all actions are illegal and then the rewritten plan is illegal. This is because there is no permission to transfer for recommendation and process recommendation in relation to D2.

The legal database has initial information that may change during the evaluation. Therefore, the legal checker receives information from the planner and updates it in real-time. To implement this system, we use Proleg [13], a language that extends Prolog with exceptions to make it easier to deal with laws.

5.1. Legal checking of plans in the use case

In the use case example we suppose that the data set $D2$ contains some categories of data which the user has not authorized to be used for the purpose of the recommendation. In other words, legal obligations for processing certain data categories are not fulfilled. The legal checker in this case filters out all the plans which involve processing $D2$ and gives as output a list of legal plans. Table 2 shows these plans indexed from 1 to 8.

Plan	Dataset	Middle Node1	Process	Middle Node2
Plan 1	D1	node 1	p1	node 1
Plan 2	D1	node 1	p1	node 2
Plan 3	D1	node 2	p1	node 1
Plan 4	D1	node 2	p1	node 2
Plan 5	D1	node 1	p2	node 1
Plan 6	D1	node 1	p2	node 2
Plan 7	D1	node 2	p2	node 1
Plan 8	D1	node 2	p2	node 2

Table 2
Legal plans after illegal plans have been filtered out

[2]In this code, we do not use any feature of Proleg, so we can read this as Prolog code.

6. Ethical checker

Here, we describe the ethical check component. Ethical constraints are usually used as a permissibility condition, as in the legal checker component that integrates the hard norms as a precondition of actions. However, in our architecture, ethical criteria are used to order plans and not as a permissibility condition, meaning that ethical conditions should be satisfied as much as possible. For this purpose, we use an ethical ordering model based on several criteria (ethical norms or guidelines). Evaluations are modeled as preference relations on an ordinal scale, and the best plan is selected based on its compliance with ethical criteria. We call the ethical criteria *soft norms* that is a prescriptive rule or guideline to respect ethical norms and can be viewed as different sources or components of utility. Unlike hard norms, soft norms do not make an evaluation of right or wrong, but rather a comparison of options by checking which one is better or respects the underlying ethical norm more. See Figure 5.

Briefly speaking, the ethical check module takes as input a set of plans, then the plans are ordered according to several criteria and aggregated to obtain a final order. The first-order plan in the final aggregation is selected as the best. Certain properties are anticipated in the final order; for example, transitivity is essential to avoid counterintuitive results. That is why we use voting rules with desired properties that are well studied in social choice theory [14].

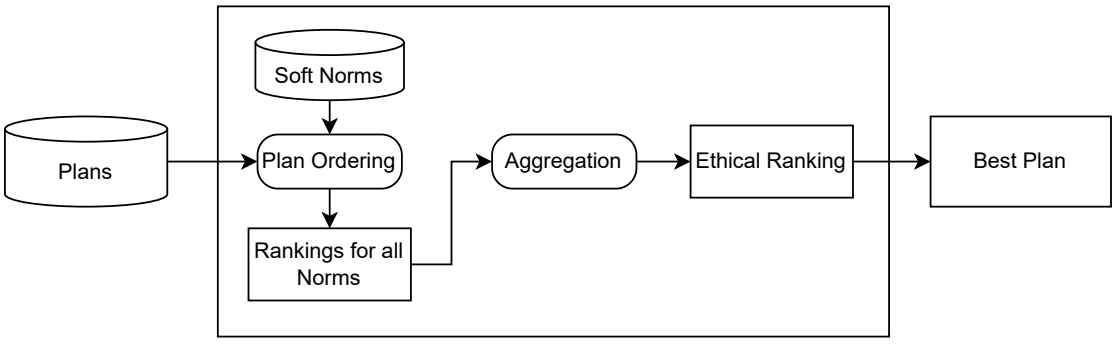

Figure 5: Ethical checker

Formally, ethical verification has consisted of $\mathbb{P}$, a set of input plans, $\boldsymbol{\mathcal{N}}$, a set of soft norms or ethical criteria, $\rho : \boldsymbol{\mathcal{N}} \longmapsto 2^{\mathbb{P}\times\mathbb{P}}$ that assigns a transitive and complete preference $\succcurlyeq_n := \rho(n)$ over the plans for each criterion $n \in \boldsymbol{\mathcal{N}}$, which can be viewed as an order or vote. $\Psi : (2^{\mathbb{P}\times\mathbb{P}})^{|\boldsymbol{\mathcal{N}}|} \longmapsto 2^{\mathbb{P}\times\mathbb{P}}$, is an aggregation function, and the final ethical evaluation of the plans is $\succcurlyeq_e = \Psi(\succcurlyeq_1, \ldots, \succcurlyeq_{|\boldsymbol{\mathcal{N}}|})$. The aggregation process is described below.

Suppose that there are $k \leq |\boldsymbol{\mathcal{N}}|$ classes of soft norms such that $\boldsymbol{\mathcal{N}} = \cup_{i\in\{1,\ldots,k\}}\mathcal{N}_i$. Classes are necessary to model a superiority relation between a group of norms, which will be described later. A vote is assigned to each class $\mathcal{N}_i$, which is the aggregation of the votes within that class and is represented by $\succcurlyeq_{\mathcal{N}_i}$.

We use Copeland's rule is our case to aggregate the votes in each class. The Copeland rule chooses the alternative(s) that win the most pairwise majority comparisons. We can use a weighted version in order to show the importance of the criterion; however, without losing

generality, we suppose that all criterion have equal importance, i.e. weights within a class. For each $a, b \in \mathbb{P}$, let r^i_{ab} be defined as follows.

$$r^i_{ab} = \begin{cases} 1 & |\{n \in \mathcal{N}_i | a \succcurlyeq_n b \wedge \neg b \succcurlyeq_n a\}| > |\{n \in \mathcal{N}_i | b \succcurlyeq_n a \wedge \neg a \succcurlyeq_n b\}| \\ \frac{1}{2} & |\{n \in \mathcal{N}_i | a \succcurlyeq_n b \wedge \neg b \succcurlyeq_n a\}| = |\{n \in \mathcal{N}_i | b \succcurlyeq_n a \wedge \neg a \succcurlyeq_n b\}| \\ 0 & |\{n \in \mathcal{N}_i | a \succcurlyeq_n b \wedge \neg b \succcurlyeq_n a\}| < |\{n \in \mathcal{N}_i | b \succcurlyeq_n a \wedge \neg a \succcurlyeq_n b\}| \end{cases} \quad (1)$$

An alternative wins in a pairwise comparison if its overall score is higher than the other.

$$a \succcurlyeq_{\mathcal{N}_i} b \Leftrightarrow \sum_{x \in \mathbb{P}} r^i_{a,x} \geq \sum_{x \in \mathbb{P}} r^i_{b,x} \quad (2)$$

We suppose that there is a superiority relation between classes that is given by a total strict order $\succ^s \subseteq 2^{\mathcal{N}} \times 2^{\mathcal{N}}$. The final preference in this case is obtained by lexicographically comparing class votes, which is formulated as follows.

$$a \succcurlyeq_e b \iff \exists\, i \in \{1, \ldots, k\} \wedge a \succcurlyeq_{\mathcal{N}_i} b \wedge (\forall\, j \in \{1, \ldots, k\} \wedge \mathcal{N}_j \succ^s \mathcal{N}_i \Rightarrow a \succcurlyeq_{\mathcal{N}_j} b) \quad (3)$$

Note that there are general properties that are desired in a voting rule, like Pareto's efficiency, monotonicity, etc. An important property in our case is the Condorcet principle, which means that a voting rule should select the alternative that beats every other alternative in pairwise comparisons as the winner[15]. The Copeland's rule used in this method is an example of such rules; other rules such as max-min, etc. that satisfy the Condorcet principle can be used interchangeably. The lack of this property may cause a cyclic preference and lead to a loss of transitivity.

6.1. Ethical checking of plans in the use case

The ethical checker uses domain knowledge to order plans and select the best one. It takes as input a list of legal plans as in Table 2, and order them according to the defined soft norms. The orders can be seen as the votes of each norm on the input plans. The soft norms used in our use case and the ethical norms or principles to which they adhere are indicated below.

- **Data minimization (N1)**: Suggests using fewer personal data categories to respect the privacy of the user.
- **Sensitive data (N2)**: Proposes using less sensitive data to respect the privacy and safety of the user.
- **Transfer regions (N3)**: It suggests avoiding transfers outside the legislative zone to protect personal data and the user's safety.
- **Node safety (N4)**: Proposes to avoid less secure storage of personal data to respect the safety of the user.
- **Transfer efficiency (N5)**: Suggests using less busy nodes to increase the efficiency of data transfers and help increase the beneficence of the service.
- **Algorithmic bias (N6)**: Suggests using processing that is not or less biased toward any group in order to respect fairness in providing the service.

Plan	N1	N2	N3	N4	N5	N6	Aggregated
Plan 1	1	1	1	1	3	1	1
Plan 2	1	1	2	2	2	1	2
Plan 3	1	1	2	2	2	1	2
Plan 4	1	1	3	3	1	1	4
Plan 5	1	1	1	1	3	2	3
Plan 6	1	1	2	2	2	2	5
Plan 7	1	1	2	2	2	2	5
Plan 8	1	1	3	3	1	2	6

Table 3
Ranking of plans by each norm

Note that the mentioned rules are some examples of soft norms that are applicable to our use case. For simplicity, we assume that there is only one class of norms and that every norm within that class is of equal importance. The input plans are then ordered on the basis of how much they satisfy the given soft norms. The order of each norm on the plans and their aggregation are shown by their corresponding ranks in Table 3. In this use case, *Plan 1* has the highest order and will be selected as the best plan to execute.

7. Action executor

In this section, we describe the action executor component. As shown in Figure 6, its role is to execute tasks provided by the planning agent and provide feedback to the planning agent upon each task completion.

The action executor is implemented in the rule language Prova [16, 17], leveraging its reactive agent programming capabilities and its support for reaction rule based workflows. Thus, each actor (nodes, users) of the use case is represented as an agent, while all actors are utilizing a common rulebase. In Prova, all agents are instances of rulebases, allowing agents with similar functionality to use the same rulebase. During execution, each rulebase instance can independently be modified through positive or negative updates (assertions and retractions, respectively). In addition, there is a coordinating Prova agent (using a separate rulebase) that communicates with the planning agent and relays messages to the other agents.

The implementation utilizes Prova's features such as message exchange, reactive agent messaging with assertions and retractions, slots, and guards. Message exchanging is performed via the message passing primitives, namely the predicates `sendMsg/5`, `rcvMsg/5`, as well as their variants `sendMsgSync/5`, `rcvMult/5`.

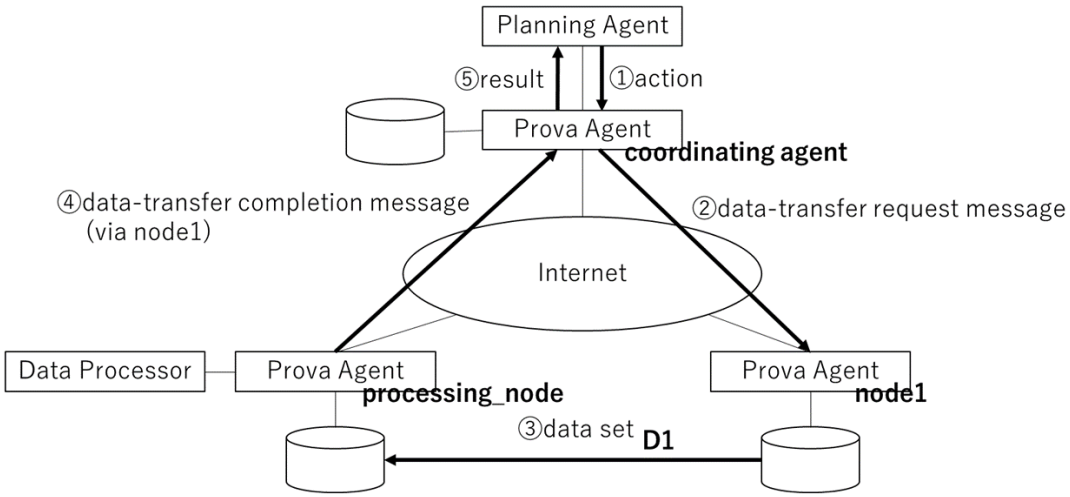

Figure 6: Data transfer action execution via Prova

7.1. Action execution in the use case

As shown in Figure 6, to execute the action `transfer(node1, processing_node, D1)`, the following steps are performed:

1. The coordinating Prova agent gets the action execution command of `transfer(node1, processing_node, D1)` from the planning agent.
2. The coordinating Prova agent relays the message to `node1`.
3. `node1` sends a message to `processing_node` containing a request to copy the data set `D1`, and the calculated checksum of `D1` for verification purposes.
4. `processing_node` receives the message, calculates the checksum of the received data, and if the two checksums match, it proceeds with accepting the message and asserting the data set `D1` in its database. Finally, it sends a data-transfer completion message to the coordinating Prova agent via `node1`.
5. The coordinating Prova agent sends the execution result to the planning agent.

For example, step 4 is performed by the following Prova code, which features a Prova guard for the checksum calculation and comparison:

```
node() :-
    rcvMult(X,P,F,Perf,{operation->copy, data->D, checkSum->CHCK2})
      ↪[CHCK = D.hashCode(),equal(CHCK,CHCK2)],
    assert(db(data,D)),
    sendMsgSync(X,P,F,inform,{operation->confirm, checkSum->CHCK}).
```

8. Conclusions

In this paper, we propose an overall architecture of AI compliance in the context of AI agent planning in multi-agent systems.

1. The planning agent computes plans which are feasible according to its belief in the current environment and sends them to the legal checker.
2. The legal checker filters illegal plans according to legal norms.
3. The ethical checker chooses the best plan according to ethical norms.
4. The planning agent requests the action executor to execute each action in the plan.
5. The action executor executes the action and reports the result to the planning agent. It also reports new observations in the environment.
6. The action executor is implemented as a multiagent system for data transfer.
7. The planning agent updates its belief and plans based on the result of action execution or new observation. If necessary, it also replans calling the legal and ethical checkers.

We believe that our real-time compliance mechanism manipulating environmental change contributes to realizing "trustworthy AI" better than embedding compliance mechanism into AI at the design phase since we cannot predict all the incompliant plans according to the Frame Problem.

As future challenges, we will consider the following:

- We would like to analyze computational complexity of real-time compliance mechanism which makes a new compliant plan according to a change of environments.
- In this work, planning, legal check, and ethical check were implemented in separate modules. This makes it easier to develop an independent database for each module. However, if the planner considers legal and ethical norms to some extent, the planner can avoid repeatedly producing clearly illegal or unethical plans before calling legal/ethical checkers, which leads to efficiency. In future work, we will test to what extent the planner should consider the legal and ethical norms in cooperation with the legal/ethical checkers.
- It would also be possible to extend the legal checker to include the data protection legislation of each country, to provide a more detailed check of the legislation, not just the GDPR.
- We would like to consider computationally efficient ways to aggregate rankings or votes for the ethical checker and to design an ontology to collect the principles and prescriptions in AI ethics guidelines which can cover a broad range of ethical norms.
- Another possible future work direction is to aim for a fully decentralized system, where the components will communicate by exchanging queries and facts. To achieve this, we can utilize a Prolog $\leftrightarrow$ RuleML translator such as BiMetaTrans [18].

Acknowledgments

This work has been partially funded by the Agence Nationale de la Recherche (ANR, French Research Agency) project RECOMP (ANR-20-IADJ-0004), Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) project RECOMP (DFG – GZ: PA 1820/5-1), JST AIP

Trilateral AI Research Grant No. JPMJCR20G4, JST Mirai Program Grant No. JPMJMI20B3, and JSPS KAKENHI Grant No. 22H00543 and 21K12144.

References

[1] E. Commission, Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206 (2018).

[2] the High-Level Expert Group on AI, Ethics guidelines for trustworthy AI, https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai (2019).

[3] G. Governatori, et al., Designing for compliance: Norms and goals, in: Proc. of RuleML2010, 2011, p. 282–297.

[4] M. B. van Riemsdijk, et al., Agent reasoning for norm compliance: a semantic approach, in: Proc. of AAMAS 2013, 2013, pp. 499–506.

[5] G. Contissa, et al., Claudette meets GDPR: Automating the evaluation of privacy policies using artificial intelligence, https://ssrn.com/abstract=3208596 (2018).

[6] F. Chesani, et al., Compliance in business processes with incomplete information and time constraints: a general framework based on abductive reasoning, Fundamenta Informaticae 161 (2018) 75–111.

[7] S. B. Naveen Sundar Govindarajulu, On automating the doctrine of double effect, in: Proc. of IJCAI 2017, 2107, pp. 4722–4730.

[8] A. Saptawijaya, L. M. Pereira, Logic programming for modeling morality, Logic Journal of the IGPL 24 (2016) 510–525.

[9] F. Lindner, R. Mattmueller, B. Nebel, Moral permissibility of action plans, in: Proc. of AAAI 2019, 2019, pp. 7635–7642.

[10] H. Hayashi, K. Satoh, Online HTN planning for data transfer and utilization considering legal and ethical norms: Case study, in: A. P. Rocha, L. Steels, H. J. van den Herik (Eds.), Proceedings of the 15th International Conference on Agents and Artificial Intelligence, ICAART 2023, Volume 1, Lisbon, Portugal, February 22-24, 2023, SCITEPRESS, 2023, pp. 154–164.

[11] H. Hayashi, S. Tokura, T. Hasegawa, F. Ozaki, Dynagent: An incremental forward-chaining htn planning agent in dynamic domains, in: M. Baldoni, U. Endriss, A. Omicini, P. Torroni (Eds.), Declarative Agent Languages and Technologies III, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 171–187.

[12] D. Nau, Y. Cao, A. Lotem, H. Munoz-Avila, Shop: Simple hierarchical ordered planner, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, p. 968–973.

[13] K. Satoh, et. al., Proleg: An implementation of the presupposed ultimate fact theory of japanese civil code by prolog technology, in: New Frontiers in Artificial Intelligence, Springer Berlin Heidelberg, 2011, pp. 153–164.

[14] H. Nurmi, Voting systems for social choice, in: Handbook of Group Decision and Negotiation, Springer, 2010, pp. 167–182.

[15] F. Brandt, V. Conitzer, U. Endriss, Computational social choice, Multiagent systems 2 (2012) 213–284.

[16] A. Kozlenkov, R. Penaloza, V. Nigam, L. Royer, G. Dawelbait, M. Schroeder, Prova: Rule-Based Java Scripting for Distributed Web Applications: A Case Study in Bioinformatics, in: T. Grust, H. Höpfner, A. Illarramendi, S. Jablonski, M. Mesiti, S. Müller, P.-L. Patranjan, K.-U. Sattler, M. Spiliopoulou, J. Wijsen (Eds.), Current Trends in Database Technology – EDBT 2006, Springer, Berlin, Heidelberg, 2006, pp. 899–908.

[17] A. Kozlenkov, Prova Rule Language version 3.0 User's Guide, 2010. URL: https://github.com/prova/prova/tree/master/doc.

[18] M. Thom, H. Boley, T. Mitsikas, Invertible bidirectional metalogical translation between Prolog and RuleML for knowledge representation and querying, in: V. Gutiérrez-Basulto, T. Kliegr, A. Soylu, M. Giese, D. Roman (Eds.), Rules and Reasoning, Springer International Publishing, Cham, 2020, pp. 112–128.