Toward smooth integration of an online HTN planning agent with legal and ethical checkers

Hisashi Hayashi, Yousef Taheri, Kanae Tsushima, Gauvain Bourgne, Jean-Gabriel Ganascia, Ken Satoh

Toward smooth integration of an online HTN planning agent with legal and ethical checkers

Hisashi Hayashi [a,*], **Yousef Taheri** [b,**], **Kanae Tsushima** [c,***], **Gauvain Bourgne** [b,****], **Jean-Gabriel Ganascia** [b,*****] **and Ken Satoh** [c,******]

[a]Advanced Institute of Industrial Technology, Tokyo, Japan
[b]Sorbonne University, Paris, France
[c]Center of Juris-Informatics, Research Organization of Information and Systems, Tokyo, Japan

Abstract. Owing to legal and ethical issues such as privacy, safety, and bias, it is crucial to adhere to the laws and respect the ethical guidelines of different countries when transferring or utilizing datasets via the Internet. Therefore, it is necessary to meticulously plan data transfer and utilization in compliance with local laws and ethical guidelines. Given the variability in legal and ethical norms across countries and the specialized knowledge required, we assume that legal and ethical checkers are implemented as independent modules that can be installed on different servers. In this study, we demonstrate how to integrate a planning agent, which utilizes an online HTN (hierarchical task network) planner, with legal and ethical checkers. We also introduce, evaluate, and compare three interaction modes between these modules, assessing the number of interactions and computation times using scenarios involving international data transfer and utilization.

1 Introduction

As data are transferred via the Internet to be used globally for numerous services, legal and ethical issues concerning privacy, security, and other factors have become central concerns. Numerous laws and ethical guidelines have been established to regulate data transfer and usage. A well-known set of data-protection regulations is the European General Data Protection Regulations (GDPR) [7]. Owing to the complexity of laws and ethical guidelines, research has focused on automated compliance checks for data transfer norms. In particular, the policy representation of the GDPR has been studied extensively [1, 4, 14, 22].

Planning the transfer and utilization of datasets is crucial because of the multistep nature of these processes. Moreover, compliance with laws and ethical guidelines is essential when constructing data transfer and utilization plans. Some studies focused on automated planning that considers ethical and legal norms [3, 9, 10, 19]. In particular, the studies [9, 10] utilized a general-purpose online HTN (hierarchical task network) planner for data transfer planning, adapting it to changing situations in which rules describing legal and ethical norms were included in the planning agent database.

Generally, owing to the complexity of legal and ethical norms, specialized expertise is required to conduct automated compliance checks across different countries. Thus, in this study, we proposed the use of a general-purpose online planner and independently developed the norm checkers. In particular, we developed a new architecture that integrates a planning agent with legal and ethical checkers implemented as separate modules. Each of the modules sharing the same interface can be implemented differently in the proposed architecture. However, when these modules are installed on separate servers, it is crucial to ensure that they use the same up-to-date information. This would ensure high efficiency through frequent interactions between these modules.

The contributions of this study are as follows: First, we propose a new architecture that integrates an online planning agent with a legal and an ethical checker. Next, we demonstrate efficiency improvement by changing the database locations and introducing the concept of fluent subscription. Finally, the efficiency gains in terms of the number of interactions between modules and their computation times are illustrated through simulations involving multiple scenarios of planning and replanning for data transfer and utilization.

The remainder of this paper is organized as follows. Section 2 presents a new architecture that integrates the three modules discussed earlier. Section 3 introduces the three interaction modes between the planning agent and the legal and ethical checkers. Section 4 explains the experimental procedure. The results thus obtained are presented and discussed in Section 5. Finally, Section 6 concludes this paper.

2 Overall architecture

This section introduces the overall architecture surrounding the planning agent as shown in Figure. 1. It includes a legal checker, an ethical checker, and an action executor. The planning agent features an online HTN planner that generates plans based on its beliefs and modifies them according to the changing states during plan execution. The agent sends action execution instructions to the action executor, and updates its beliefs and plans based on reports from the action executor. A legal checker evaluates each action in a plan based on legal norms to determine whether it is legal. The ethics checker selects the most ethical plan by comparing multiple plans based on various ethical norms. The action executor performs actions and reports

* Corresponding Author. Email: hayashi-hisashi@aiit.ac.jp
** Email: yousef.taheri@lip6.fr
*** Email: k_tsushima@nii.ac.jp
**** Email: gauvain.bourgne@lip6.fr
***** Email: jean-gabriel.ganascia@lip6.fr
****** Email: ksatoh@nii.ac.jp

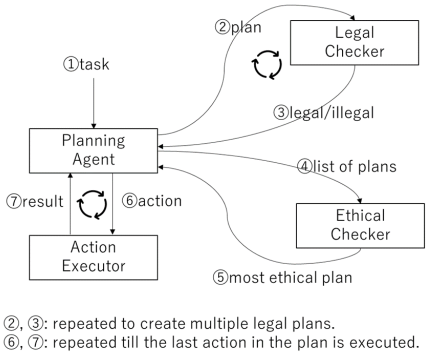

Figure 1. The flow of planning and execution in the proposed architecture.

the results to the planning agent. Sometimes, the executor recognizes unexpected changes in the world, such as changes in the activeness, safety level, or occupancy level of servers, and reports them to the planning agent.

Given task (①), the planning agent creates a least-costly plan using best-first search and sends it to the legal checker (②). The legal checker determines whether the plan is legal and reports the results to the planning agent (③). The planning agent then constructs the second least-costly plan and sends it to a legal checker (② in the second loop) for legal verification. This process is repeated (②–③) until a predefined number of legal plans are obtained or no more possible plans exist. the planning agent sends these low-cost legal plans to the ethical checker (④) and requests that it select the most ethical plan.

The ethics checker then selects the most ethical plan from the given legal plan and reports it back to the planning agent (⑤). At this point, the planning agent commits to the plan selected by the ethical checker. This plan is legal and the most ethical. It sequentially executes each action in the plan using the action executor (⑥). Following the action execution request from the planning agent, the action executor attempts to execute a specified action and/or conduct observations. The results are then reported to the planning agent (⑦), which updates its beliefs and plans based on action execution result and/or observations. When the current plan may become invalid or less cost-efficient, the action executor reports new observations to the planning agent, triggering replanning. Similar to initial planning, the planning agent calls on legal and ethical checkers during replanning (②-⑤).

2.1 Planning agent

The planning agent creates plans using a planner based on the online forward-chaining total-order HTN planning algorithm of Dynagent [11]. Similar to SHOP [13], a standard (offline) HTN planner, it creates plans through task decomposition using a best-first search to find the least-costly plan. The information used for planning is called belief and includes facts, task preconditions, action effects, task costs, and task decomposition rules (called methods in SHOP).

Because of the expressiveness of the planning domain heuristics, SHOP-like total-order HTN planners continue being utilized and studied to improve computational efficiency [2, 12, 18]. Another modern online forward-chaining HTN-like planner conducted a Monte Carlo tree search [15, 16] to find a plan in a large search space.

The planning agent also monitors and controls plan execution, incrementally modifying alternative plans during execution. A state change may affect certain task preconditions in the plans. Therefore, the planning agent checks the preconditions, deletes invalid plans, and adds new valid plans to adapt to a changing world. Moreover, the plan is adjusted if it becomes invalid or less cost-efficient.

The planning agent uses the action executor to perform actions in the current plan. Each time an action is successfully executed, the belief is updated based on the action's effects. The planning agent removes, first, the executed action from the head of each plan, and second, invalid alternative plans. It then adds new valid plans. If an action execution fails, the current plan becomes nonexecutable, and all plans with this failed action at their heads are removed from the alternatives.

As mentioned earlier, the planning agent also relies on legal and ethical checkers to filter out illegal plans and select the most ethical legal plan, respectively.

2.2 Legal checker

Various studies have been conducted on legal compliance using modal (deontic) logic [8, 21], natural language processing [6], and logic programming [5]. In addition, some languages have been introduced to represent legal rules, such as Proleg [17], which extends Prolog with exceptions to handle laws better. In this study, we used the logic programming language Prolog in the legal checker for the following reasons: First, it allows the logic of legal norms with exceptions to be expressed as "negation as failure." Second, because we implemented other parts of the system using Prolog, using the same language for the legal checker helps ensure a seamless implementation. However, each module can be implemented in any programming Language in theory.

The legal checker verifies whether the plan suggested by the planning agent is legal. Because a plan consists of a list of actions, the legal checker evaluates each action and deems the plan legal only if all actions are legal. In this study, the legal checker checks whether the given actions are legal according to GDPR based on the given database information. The database contains information about the permissions granted by of the data owners, countries in the EU, and nodes in the EU, and so on. For example, if a data owner does not grant permission to transfer the data outside the EU, the legal checker determines that it is illegal if the given plan uses the data and a route that goes outside the EU.

2.3 Ethical checker

The ethical checker is responsible for evaluating and selecting the best plan among the valid ones. Its evaluation mechanism was first introduced in [20] and is primarily an ordering process based on a model with multiple criteria. The ordering process considers different criteria which can stem from either moral or optimization considerations. Moral criteria refer to a certain harm or risk that can affect the individuals involved, while optimization criteria are necessary for system efficiency. Hence, they can be seen as either neutral or moral criteria that aim to promote good instead of preventing harm. For example, in the use case model described in Section 4, personal data are transferred through different nodes to be processed for a certain purpose. In this case, two criteria (among others) are used in order to select the best path: node safety and node occupancy. Transferring data through a safer node reduces the risk of a breach and subsequent harm to the subject. Data transfer through a less busy node is faster, which increases the overall system efficiency. Therefore, although

node occupancy is not directly related to risk, it affects system performance and user satisfaction.

The input plans are evaluated on each criterion using an ordinal scale : each criterion orders the plans according to its underlying standard. An ordinal scale helps avoid inconsistencies and improves expressivity of ethical evaluations. After ordering plans according to multiple criteria, they are aggregated to obtain a single order and the best plan is identified. We consider two types of aggregation behavior. First, an order may be (universally) superior to another, in which case, the aggregated order is similar to the superior one, and the inferior order is only considered when the two alternatives have an equal order. Second, when there is a type of reconciliation or trade-off between two (or more) orders instead of superiority, they are seen as votes and aggregated by a voting rule. Note that voting rules from computational social choice theory can be used in this case. Furthermore, the orders can be weighted to represent their importance during the aggregation. Finally, all orders are aggregated by specifying the superiority and trade-off relationships between their corresponding criteria. This specification serves as the ethical setting for the ethical checker, which is built on a relativist view, meaning that it does not judge which input plans are morally right or wrong; instead, it selects the best plan by identifying the one that is best aligned with the given ethical setting.

3 Interaction modes between modules

In this section, we introduce three interaction modes between the planning agent and the legal and ethical checkers.

As discussed in Section 2, the planning agent interacts with legal and ethical checkers during planning and replanning. We assumed that the planning agent, legal checker, and ethical checker are implemented as separate modules that can be installed on different servers. This assumption is natural, given that ethical and legal norms vary between countries. To achieve higher efficiency, we must reduce the number of interactions between these modules and decrease the computation time. Additionally, it is essential to ensure that the most recent information is reflected in the plans.

We introduced the following three interaction modes. 1: *default* mode, 2: *subscription* mode, and 3: *all-subscription* mode. The interaction modes are compared in Section 5 through experiments that evaluate the number of interactions between modules and the required computation time. The three interaction modes are described in the following subsections.

3.1 Default mode

The default mode is the simplest interaction design and serves as the baseline mode. Figure 2 shows interactions in the default mode. In this mode, the common knowledge of fluents describing the changing world is recorded in the planning agent database as a belief. Legal and ethical checkers query the planning agent regarding the truth value of a fluent whenever they need to evaluate a plan for legal or ethical checks, respectively.

Each time an action is executed or the truth value of a fluent is updated, the planning agent replans and updates multiple plans, the legal checker verifies the legality of each updated plan, and the ethical checker selects the most ethical plan from these updated legal plans.

This default interaction mode ensures that the most recent information is used for planning, replanning, and legal and ethical checks.

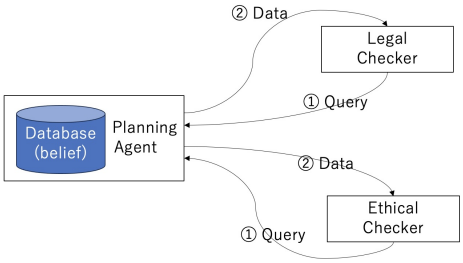

Figure 2. Default mode.

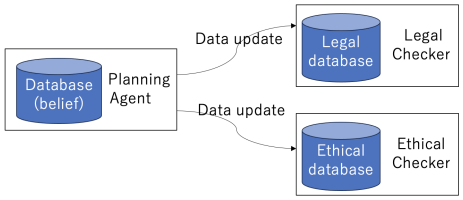

Figure 3. Subscription mode.

However, this is inefficient because the planning agent sometimes requests legal and ethical checks, even when unnecessary. In addition, legal and ethical checkers frequently query the planning agent for the truth value of a fluent, thereby increasing the number of interactions.

3.2 Subscription mode

The subscription mode was designed to improve interaction efficiency. Although the default model is simple and relatively easy to implement, it is inefficient for two reasons: First, the legal and ethical checkers frequently query the planning agent to check the truth value of a fluent, which is among the planning agent's beliefs. This significantly increases the number of interactions between these modules. Second, the planning agent sends requests to the legal and ethical checkers each time an action is executed, increasing unnecessary legal and ethical checks, number of interactions, and the required computation time. When an action in a plan is executed successfully, if the action execution does not change the truth values of the fluents that affect legal and ethical norms, it is unnecessary to modify the current plan and refer to legal and ethical checkers.

In the subscription mode, to address the first reason, the legal and ethical checkers declare the fluents that affect their norm checks as subscribed fluents. Figure 3 shows the interactions in subscription mode. The legal (or ethical) checker maintains a separate database of the subscribed fluents. Initially, the planning agent, legal checker, and ethical checker record the same truth values for each subscribed fluent in their databases.

To address the second problem, in the subscription mode, the planning agent omits legal and ethical checks when an action is successfully executed, provided that the action execution does not change the truth values of fluents that affect legal or ethical norms. However, if the action execution changes these truth values, the planning agent requests the legal checker to refilter the illegal plans and the ethical checker to select the most ethical legal plan.

If the truth value of a fluent is updated through observation, the validity of the plans may be affected. In such cases, the subscription mode is similar to the default mode, i.e., the planning agent replans and creates multiple plans, the legal checker verifies the legality of

each of these plans, and the ethical checker selects the most ethical legal plan.

3.3 All-subscription mode

The *all-subscription* mode is a special example of the subscription mode. In this mode, all fluents are subscribed by their legal and ethical standards. In this case, it is unnecessary to declare the subscribed fluents.

4 Use case model

In order to show the characteristics and efficiency of our proposed approach, we apply it in a data transfer and processing situation. A similar use case model has been used in [19] and [9, 10] as a demonstration of legal / ethical compliance of data manipulations. The model mainly includes multiple nodes that are used to transfer or process data and are connected as illustrated in Figure 4. Each node represents a section of a corporation that is located at a different location, which may be within the EU or outside the EU. Node 4 (marked as a square) is the central node that serves as a cloud server to process data for different purposes. Other nodes (marked as circles) are used to store and transfer data. In this use case, users' personal data are stored in circle nodes. Different sections may ask to apply a processing on data and receive the output of the processing at their corresponding node.

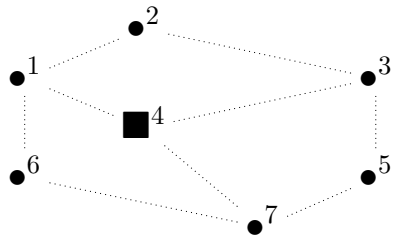

Figure 4. Nodes and connections in the network

In order to perform a task, the system locates the data, transfers them to the processing node, and applies a process with the corresponding purpose. After processing personal data, the system delivers the output to the requested node. The planner in our architecture generates possible plans to satisfy the given task, i.e. the possible paths to transfer data and process them in the network. Each possibility represents different behaviors of the system. According to this architecture, these behaviors are verified by the legal checker for any infringement of the (modeled) regulations. The legal checker rules out the illegal plans, and the remaining plans are ordered by the ethical checker based on their alignment with the ethical specification (cf. Section 2.3).

There is additional information on this use case that enables testing our architecture in different scenarios. Table 1 shows the information on the nodes. The *region* is the location of each node. Since our focus is particularly on GDPR, the regions are categorized as *EU* and *NonEU*. The region of the node is used in the legal verification process. Transferring personal data outside the legislative zone may have ethical implications for data subjects; it is also used in the ethical verification process. The *safety level* corresponds to the safety protocols supported by each node that can be high, medium, or low. Transferring data through more secure nodes is necessary to avoid any possible breach that harms user privacy. Thus, it is important in ethical checking process. The *occupancy level* indicates whether or not a node is busy. It is used to minimize data management time and improve the overall efficiency of the system.

Table 1. The attributes of each node

Node	Region	Safety Level	Occupancy Level
1	Non EU	medium	normal
2	EU	medium	normal
3	EU	medium	busy
4	EU	high	busy
5	EU	high	normal
6	Non EU	low	busy
7	Non EU	high	normal

Table 2. The information of available processing

Processing	Location	Purpose	Bias Level	Required Categories
p1	node 4	recom	2	[c1,c2,c3,c4]
p2	node 4	recom	1	[c2,c3,c5]
p3	node 4	recom	3	[c1,c3,c6,c7,c8]

Table 3. The information on personal data

Data	Category	Storage Location	Owner
du11	c1	node 1	u1
du12	c2	node 1	u1
...	...	...	...
du27	c1	node 2	u2
du28	c2	node 2	u2

Table 2 shows the processing available to apply on personal data. It includes information on the *location* of processing that is node 4 and the *purpose* that is *recommendation* for all processing in this case. The *bias level* shows the extent to which processing can be biased with respect to a certain group. We show this simply by positive integers. Each processing requires certain *categories* of data which are indicated by a list and the category name, e.g. c1, c2, etc.

Last but not least, Table 3 shows information on personal data. This includes their corresponding *category*, the node on which the data are *stored*, the data subject who is the *owner* of the personal data, and permission from the user to take the data out of the EU. Note that the permission may be changed by the data owner during execution.

We demonstrate the functionality of our architecture by testing it in some scenarios in the following section.

4.1 Scenario basecase

Scenario basecase is the baseline scenario. In particular, the situation remains unchanged and the job given to the planning agent is as follows: load the necessary data and process recommendations and deliver the results to node 7. As shown in the map, several routes can be considered. First, the planning agent creates several plans using different data and/or different routes. The legal checker performs the following checks on those plans: node 7 is outside the EU, and some data are prohibited from being taken out of the EU; thus, the plans containing prohibited data are rejected. The ethical checker chooses the best plan from the legal plans. When we ran our prototype, the chosen plan used the following route: node 1 $\rightarrow$ node 4 (recommendation process) $\rightarrow$ node 7.

4.2 Scenario precondition-replan

This scenario aims to show how the system reacts to physical changes in the operating environment, that is, changes in the use case of connected networks, which is explained in the previous section. The objective is to process the personal data of user $u2$ for recommendation purposes. The data are initially stored in a database at node 2 and the processing output is requested at the same node. The initially selected plan is to transfer the data to node 4 via node 1, apply processing $p2$, and send the output to node 2 via node 1. As shown in Table 1, nodes 1 and 3 have the same values for every attribute, except for the occupancy level, where node1 is less busy than node 3; therefore, node 1 is selected in the initial plan. During execution, when the data are loaded from the database, the system realizes that node 1 is suddenly deactivated. The planner replans and selects node 3 as an intermediate to both send data to node 4, where the processing is applied, and transfer it back to node 2. This new plan is executed step by step, and just after the processing in node 4, the system recognizes that node 1 has been reactivated. This new change is considered by replanning from the current stat and node 1 is chosen again as the intermediate node for sending the output back to node 2.

4.3 Scenario ethical-replan

Scenario ethical-replan illustrates how the system reacts to changes that affect the ordering of plans by the ethical checker. In this scenario, the task is to use $u1$'s personal data to create recommendations and deliver results at node 5. $u1$'s data is stored at node 1. To perform the task, the planner transfers personal data from node 1 to node 4 to run the selected process and chooses an intermediary node between nodes 3 and 7 to deliver the result to node 5. Because the safety level of node 7 is higher, the ethical checker initially selects a plan that transfers data through this node. However, just after processing the data at node 4, the system realizes that, owing to some external incidents, the safety level of node 7 has changed to *low*. A re-evaluation is then initiated by the system, and the ethical checker selects the path that passes through node 3 because it is now safer. In this scenario, the physical constraints are fixed; however, the properties that affect the ordering of the ethical checker, and consequently, the selected plan, are changed. The re-evaluation process demonstrates the functionality of our proposed architecture and the ethical checker component in similar situations.

4.4 Scenario legal-replan

In Scenario legal-replan, the planner discovers that a user has rewritten the permission information in the database during execution. The legal checker checks the legality again and finds that the chosen plan is currently not allowed. Thus, the planner re-creates different plans. Specifically, the initial plan selected the dataset [du21,du23,du26,du27,du28] and the route to obtain data from nodes 2 to 7 via the EU to achieve the goal. However, during execution, the permission information for du28 was rewritten to prohibit taking the data out of the EU, which illegalized moving the data through this route. Therefore, the planner uses another dataset [du22,du23,du25] to achieve this goal.

5 Experiments and discussions

Tables 4 and 5 show the executed results. All executions were performed using SWI-Prolog (threaded, 64 bits, version 9.0.4) on a computer: Mac Book Air running MacOS 14.4.1, Apple M2, 8 cores, and 24GB memory. All the runs used the same maximum number of plans, 16. This implies that the planner can create a maximum of 16 plans. The database information presented in Section 4 was almost the same; however, some parameters were modified to represent each scenario. Note that each module can be implemented in any programming language and installed on different servers as long as they can communicate with one another, for example, via remote procedure calls.

Table 4. Executed results: CPU time in seconds.

	default	all-subscription	subscription
basecase	1.349094	1.343983	0.465795
precondition-replan	2.80622	2.747913	1.578951
ethical-replan	2.714904	2.704922	1.313285
legal-replan	3.407112	3.370376	0.809458

Table 5. Executed results: the number of interactions.

	default	all-subscription	subscription
basecase	16916	84	25
precondition-replan	50357	140	51
ethical-replan	32760	121	43
legal-replan	41069	157	47

In our current implementation, we used SWI-Prolog to run three modules on a single computer. Therefore, the communication cost between the modules is minimal. However, these modules could be distributed across servers, increasing the communication cost between the modules. In this experiment, the communication cost was evaluated by counting the number of interactions.

Comparing the default and all-subscription modes, the computation times were almost equal but the number of interactions in the all-subscription mode was significantly lower.

In the default mode, the legal and ethical checkers are called whenever an action is executed. The all-subscription mode functions similarly because an action execution normally changes the truth values of some fluents, which are subscribed to by both checkers.

Furthermore, in the default mode, the legal and ethical checkers have to request the planning agent for the truth value of a fluent. Whereas, in the all-subscription mode, these checkers consult their own databases and need not consult the planning agent. This significantly reduces the number of interactions.

Considering the communication time required for each interaction, the impact of all-subscription mode is huge. Note that, although the communication times for interaction are not included in Table 4, it is possible to estimate them by multiplying the number of interactions and the approximated unit communication time.

In the subscription mode, the number of unnecessary legal and ethical checks are reduced. Compared with the all-subscription mode, both the number of interactions and the computation times is lower. This shows the considerable impact of the subscription mode on the system efficiency.

In any case, the subscription mode was the most efficient in terms of number of interactions and computation time.

6 Conclusion

This paper demonstrates the implementation of a planning agent that smoothly integrates an online planner, a legal checker, and an ethical checker. Moreover, we compared and evaluated three interaction modes and found that the fluent subscription technique works well and significantly reduces the number of interactions and computation time, which are vital for the smooth and efficient integration of

these modules. In future, we plan to improve our integration method for real-time computation of legal and ethical planning.

Acknowledgements

This work was partially funded by JST AIP Trilateral AI Research Grant No. JPMJCR20G4; JST Mirai Program, Grant No. JPMJMI23B1; and JSPS KAKENHI, Grant No. 22H00543 and 21K12144; and Agence Nationale de la Recherche (ANR, French Research Agency) project RECOMP (ANR-20-IADJ-0004).

References

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

[2] G. Behnke, D. Höller, and S. Biundo. totSAT — totally-ordered hierarchical planning through SAT. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 6110–6118, 2018.

[3] F. Berreby, G. Bourgne, and J.-G. Ganascia. Event-based and scenario-based causality for computational ethics. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 147–155, 2018.

[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] F. Chesani et al. Compliance in business processes with incomplete information and time constraints: a general framework based on abductive reasoning. *Fundamenta Informaticae*, 161(1-2):75–111, 2018.

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

[7] European Commission. Regulation (EU) 2016/679 of the European Parliament and of the Council, 2016. URL http://data.europa.eu/eli/reg/2016/679/oj.

[8] G. Governatori et al. Designing for compliance: Norms and goals. In *International Joint Conference on Rules and Reasoning*, page 282–297, 2011.

[9] H. Hayashi and K. Satoh. Towards legally and ethically correct online htn planning for data transfer. In *International Conference on Agents and Artificial Intelligence*, volume 1, pages 154–164, 2023.

[10] H. Hayashi and K. Satoh. Online HTN planning for data transfer and utilization considering legal and ethical norms: Case study. In *International Workshop on Non-Monotonic Reasoning, Federated Logic Conference*, volume 3197 of *CEUR Workshop Proceedings*, pages 4–15, 2023.

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

[12] M. C. Magnaguagno, F. Meneguzzi, and L. Silva. HyperTensioN: A three-stage compiler for planning. In *International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network Planning Track*, pages 5–8, 2021.

[13] D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila. SHOP: Simple hierarchical ordered planner. In *International Joint Conference on Artificial Intelligence*, volume 2, page 968–973, 1999.

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

[15] S. Patra, M. Ghallab, D. Nau, and P. Traverso. Acting and planning using operational models. In *AAAI Conference on Artificial Intelligence*, pages 7691–7698, 2019.

[16] S. Patra, J. Mason, A. Kumar, M. Ghallab, P. Traverso, and D. Nau. Integrating acting, planning, and learning in hierarchical operational models. In *International Conference on Automated Planning and Scheduling*, pages 478–487, 2020.

[17] K. Satoh and et. al. Proleg: An implementation of the presupposed ultimate fact theory of Japanese civil code by prolog technology. In *New Frontiers in Artificial Intelligence*, pages 153–164. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25655-4.

[18] D. Schreiber. Lilotane: A lifted sat-based approach to hierarchical planning. *Journal of Artificial Intelligence Research*, 70:1117–1181, 2021.

[19] Y. Taheri, G. Bourgne, and J.-G. Ganascia. A compliance mechanism for planning in privacy domain using policies. In K. Yada, Y. Takama, K. Mineshima, and K. Satoh, editors, *New Frontiers in Artificial Intelligence*, pages 77–92, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-36190-6.

[20] Y. Taheri, G. Bourgne, and J.-G. Ganascia. Modelling integration of responsible AI values for ethical decision making. In *Workshop on Computational Machine Ethics, International Conference on Principles of Knowledge Representation and Reasoning*, 2023.

[21] M. B. van Riemsdijk et al. Agent reasoning for norm compliance: a semantic approach. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 499–506, 2013.

[22] M. D. 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, 2019.