Ethical Monitoring and Evaluation of Dialogues with a MAS*

Abeer Dyoub[1], Stefania Costantini[1], Francesca A. Lisi[2], and Ivan Letteri[1]

[1] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Università degli Studi dell'Aquila, Italy
abeer.dyoub@univaq.it,stefania.costantini@univaq.it,ivan.letteri@univaq.it
[2] Dipartimento di Informatica &
Centro Interdipartimentale di Logica e Applicazioni (CILA)
Università degli Studi di Bari "Aldo Moro", Italy
FrancescaAlessandra.Lisi@uniba.it

Abstract. Chatbots are tools aimed at simplifying the interaction between humans and computers, typically used in dialogue systems for various practical purposes. These systems should be built on ethical foundations because their behavior may heavily influence a user (think especially about children). The primary objective of this paper is to present the architecture and prototype implementation of a Multi Agent System (MAS) designed for ethical monitoring and evaluation of a dialogue system. A prototype application, for monitoring and evaluation of chatting agents' (human/artificial) ethical behavior in an online customer service chat point w.r.t their institution/company's codes of ethics and conduct, is developed and presented. We focus on the implementation specifics of the proposed system and the presented prototype application. Future work and open issues with this research are discussed.

1 Introduction

Machine Ethics is an emerging field concerning itself with the ethical behavior of autonomous intelligent agents. Concerns about the ethical behavior of such machines is growing, especially with the increasing autonomy, and with agents 'invading' our everyday life and starting to perform many tasks on our behalf.

Engineering machine ethics, or building practical ethical machines is not just about traditional engineering. With machine ethics, we need to find ways to practically build machines that are ethically restricted, and can also reason about ethics. This involves philosophical aspects, even though the problem has a non-trivial computational nature.

Chatbots are an example of 'artificially intelligent' systems that has been a result of research and development in recent years. Chatbots are tools aimed at simplifying the interaction between humans and computers, typically used in dialogue systems for various practical purposes including customer service or information acquisition. From a technological point of view, a chatbot represents the natural evolution of question-answering system leveraging Natural Language Processing. Today, most chatbots are

either accessed via virtual assistants such as Google Assistant and Amazon Alexa, or via messaging apps such as Facebook Messenger or WeChat, or via individual organizations' apps and websites. Business activities are rapidly moving towards the adoption of chatbots and other self-service technologies. This in order to automate basic communications and customer service, to reduce the call center costs and to provide advanced services to users.

However, chatbots raise many ethical concerns. Unethical Artificial Intelligence and bots are a big concern for many consumers. The chatbot should be built on ethical foundations because its behavior influences the company's image, and unethical behavior will lead to mistrust from the client-side.

In previous works [3–5], a hybrid logic-based approach was proposed for ethical evaluation of chatbots' behavior, concerning online customer service chat points, w.r.t institution/company's codes of ethics and conduct. The approach is based on Answer Set Programming (ASP) as a knowledge representation and reasoning language [1], and Inductive Logic Programming (ILP) for learning rules needed for ethical evaluation and reasoning [12].

In this paper, we focus on the challenge of monitoring and evaluating the ethical behavior of dialogue systems by proposing and implementing an application for monitoring and evaluation of chatting agents' (human/artificial) ethical behavior in an online customer service chat point w.r.t their institution/company's codes of ethics and conduct. The system is designed and implemented as a Multi-Agent System, and is based on the above mentioned ethical evaluation approach. The MAS acts as a separate ethical layer that can be integrated with existing dialogue systems. Our System is a pilot system aiming at first place to test the previously proposed ethical evaluation approach, and constitutes a step towards building practical ethical machines.

Today, MAS are considered as an interesting and suitable way of comprehension, modeling, designing, and implementing different kinds of (distributed) systems. Multi-agent technologies provide many advantages including: Autonomy for delegation, monitoring our environments, allows more efficient interaction and resource management. In addition to providing answers to many challenges of practical development of distributed and open software systems, such as decentralization, distribution of control, capacity of integration, flexibility, adaptation, trust, security, and openness [8]. It is very easy to incorporate modifications in the behavior of individuals, by adding behavioral rules which act at the individual level. It is also possible to add new agents with their own behavioral model, which interact with the already defined agents. Multi-Agent Systems provide a good solution for distributed computing applications like Internet applications.

Agent-oriented abstractions and multi-agent systems are well known in literature as a programming paradigm for the realization of complex and dynamic systems [8]. Accordingly, our implementation employs Logical Agents (agent-oriented approaches based on Computational Logic), and exploits relevant AOSE (Agent-Oriented Software Engineering) existing work. Namely, we adopt the JaCaMo [3] methodology to design and implement a MAS simulation environment. Via the JaCaMo platform, our application is realized by means of a set of Jason agents encapsulating the logic and the control

[3] `http://jacamo.sourceforge.net`

of the specific tasks involved in the application, and operating with respect to the organizational constraints. Such constraints are defined through appropriate organizational artifacts which provide the functionalities, and the operations giving access to these functionalities, that agents can employ to perform their tasks in the specific context of the given application.

The paper is organized as follows. We start, in Section 2 with a short background introducing JaCaMo framework, and shortly recalling the ethical decision making and judgment approach adopted in this work. The MAS proposed architecture is given in Section 3. Then the details of the MAS design and implementation are described in Section 4. In the Section 5, an example showing the data flow through the system is presented. Finally, we conclude with discussion and future directions in Section 6.

2 Background

2.1 Ethical Evaluation Approach

The ethical evaluation approach implemented in the proposed system is based on previous work discussed in [3–5]. This approach combines both top-down (rule-based) and bottom-up (learning) approaches in one unified hybrid framework. The approach is a purely declarative logic-based approach, that makes use of ASP as the main knowledge representation and reasoning language, and of ILP for learning the missing ASP rules needed for ethical reasoning.

The approach is based on the elaboration of facts extracted from documents containing the code of ethics and conduct that is proper of the given domain or organization, and from real life situations concerning pertinent ethical decision-making and judgment. These facts are used to elicit rules for ethical reasoning. Thus, the approach is general enough to produce ethical reasoning rules for any domain. The motivation of devising such general approach is that Codes of Ethics in customer service are in general a set of abstract principles, aimed at objectively specifying the promises and obligations, related to the company's products or services modalities of delivery, and to complaints management in the interaction with customers. For instance, such principles may include confidentiality, accountability, honesty, fidelity, etc. These ambiguous principles may carry different meanings according to contexts, and furthermore they are subject to interpretation. Therefore, it is quite difficult if not impossible to define such codes in a manner that they may be applied deductively. And, is hardly possible for experts to define intermediate rules to cover all possible situations to which a particular code applies. In addition, there are many situations in which obligations might conflict.

In [3–5] we proposed an approach for generating the missing ethical detailed rules needed for ethical decision making and judgment via learning from interactions with customers over time. In our approach, the ethical evaluation agent will initially have in its knowledge base the set of ethical codes that provide a clear decision procedure which is encoded deductively using ASP. When the ethical evaluation agent does not have the proper rule to be able to provide an ethical evaluation of a certain case scenario, the needed rule will be learned by means of the learning module which uses ILP for this purpose.

2.2 Multi-Agent Systems

Multi-Agent systems are a branch in Distributed Artificial Intelligence (DAI). The term MAS is used to describe all types of systems composed of multiple autonomous components showing the following characteristics: each agent has incomplete capabilities to solve a problem, there is no global system control, data is decentralized, and computation is asynchronous [9].

MAS can be viewed as a society of agents that interact with each other to achieve their own private goals or some global goal. Agents are embedded in a certain environment. This environment is the world where agents perceive changes, do actions to provoke changes, and adapt to the environment by learning. In MAS agents have their own plans, intentions, beliefs, and local goals. For global goals, there is a need for a coordination mechanisms through which agents engage to ensure that the MAS acts in a consistent manner. Agent Oriented Software Engineering (AOSE) is an agent-specific software engineering. AOSE defines abstractions (of agents, interactions, protocols, context, and environment), specific methodologies, and tools for the development of MAS applications. MAS platforms support effective design and construction of agents and multi-agent systems. Several agent platforms are available, e.g. JADE[4], JaCaMo (the platform used in this paper, refer Section 2.2). In recent years MAS technology has become a powerful platform for engineering real-world applications. MAS provides a good solution for distributed control. MAS have many applications in various domains, including ambient intelligence, grid computing, electronic business, semantic web, bioinformatics and computational biology, monitoring and control, resource management, education, space, military and manufacturing, etc. For more information about MAS applications, among many, see [11, 2, 13, 17].

JaCaMo Framework . JaCaMo is a platform for the development and execution of Multi-Agent Systems. JaCaMo combines three separate technologies, each of them being well-known and being able to function on its own:

- Jason [5]: for programming autonomous agents in the AgentSpeak language.
- CArtAgO [6]: for programming environment artifacts.
- Moise [7]: for programming multi-agent organization.

These three platforms together cover all levels of abstractions needed for the development of sophisticated MAS systems.

Jason is an interpreter for an extended version of AgentSpeak [15]. It implements the operational semantics of this language and provides an agent-oriented programming platform with many user-customizable features. AgentSpeak is an agent-oriented programming language for BDI agents, based on logic programming and inspired by BDI logics [6].

[4] `https://jade.tilab.com/`
[5] `http://jason.sourceforge.net/wp/`
[6] `http://cartago.sourceforge.net/`
[7] `http://moise.sourceforge.net/`

CArtAgO (Common ARTifact infrastructure for AGents Open environments) is a general-purpose framework, that makes it possible to program and execute virtual environments. It is, in particular, a java-based programming model for defining so-called 'artifacts'. Artifacts are aimed to model components of the systems (resources of various kinds) that are manipulated by agents to perform their activities. *Workspaces* are virtual 'containers' including a number of agents and artifacts.

Moise is an organizational-oriented programming framework for Multi-Agent Systems based on notions like roles, groups, and missions. It allows a designer to define an explicit organizational specification of a multi agent system. This specification can be used by the agents to reason about their organization, and also by an organization platform to enforce that the agents follow the specification.

JaCaMo is a comprehensive approach which provides the benefits of integrating different dimensions of MAS development. This, in fact, has the advantage of simplifying the programming model adopted in the development of complex MAS. Furthemore, from an implementation point of view, JaCaMo approach is able to maintain separation of concerns, aspects related to agents, organisations, environments are specified and programmed using specific separate abstractions and corresponding language constructs. This contributes to multi-agent systems development in terms of modularity, reusability, readability, extensibility, and software maintenance. JaCaMo approach simplifies MAS programming, and makes it possible to have cleaner and typically shorter programs. This is possible because of the reasoning, interpreting, monitoring engines, and infrastructures that are available in the three platforms incorporated into JaCaMo and the interfacing among them which also make automatic many things that programmers would normally have to worry about themselves (such as the perception of properties as regards the environment, and the mapping of agent actions to environment operations, etc.).

3 EthicalEvalMAS: Architecture

In this section we present the proposed MAS architecture. In the context of an online customer service dialogue system, we want to ensure an ethical behavior from the chatting agent (human/artificial). Online customer service agents are monitored for ethical violations by the proposed architecture. In order to achieve this overall goal, the MAS is composed of a group of agents, each one is responsible for a specific subtask in the overall ethical monitoring task. The architecture is shown in Figure 1. The online customer service environment in this work consists of clients, online customer service agents (human/artificial), and software agents. A client interacts with the system via chatting point interface, where she/he can write their requests (questions), and receive answers. Answers to the client's requests are given by the online customer service agent. Software agents in the environment are: client agent (CA), chatting agent (ChA), text extractor agent (TEA), text-ASP translation agent (TATA), ethical evaluation agent (EEA), and monitoring agent (MA). Text extractor agent is responsible for extracting chat text from the chatting point interface and sending it to the TATA agent

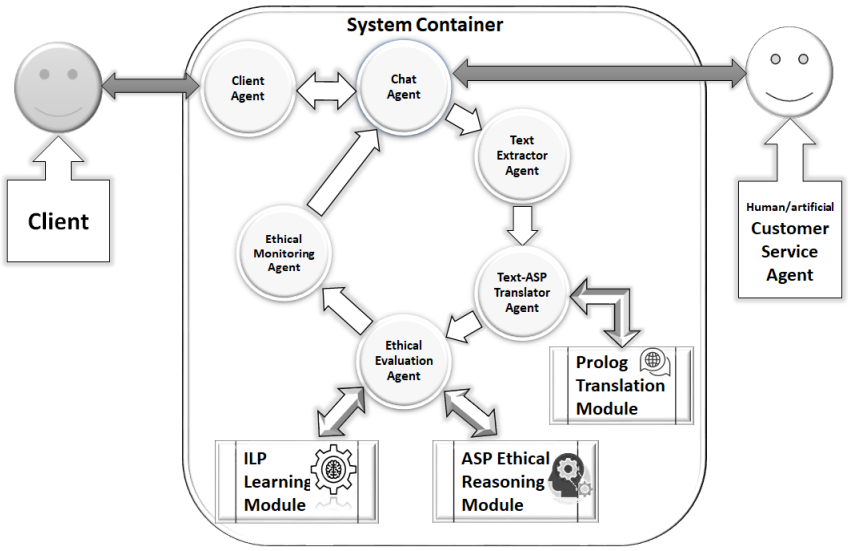

Fig. 1. EthicalEvalMAS Architecture

which is responsible for translating the chat text into ASP syntax. This agent uses Prolog[8] module for this purpose (a simple translator was written in Prolog, this translator is able to translate simple natural language sentences into Prolog-like predicates). The ethical evaluation agent has two primary goals: (1) to generate ethical evaluation of the online customer service agent's answers using ASP reasoning module, which utilizes the current case facts, and the background knowledge (BK) from the knowledge base (KB) to give the evaluation. This module is based on Clingo solver[9]. (2) Learning the ethical rules needed for ethical evaluation, saving it to its KB, in case the ASP reasoning module is not able to give an evaluation (i.e. no answer set). EEA uses the ILP learning module to achieve this purpose. Finally, the MA agent currently responsible only for alerting the CA agent for ethical violations (the role of this agent can be extended to practice more control over the CA agent).

4 EthicalEvalMAS Design and Implementation

For building our MAS model, we have used the JaCaMo framework (cf. Section 2.2). This framework allows us to program our MAS in terms of an organization of cognitive agents, sharing a common artifact-based environment. The developed application is available on github[10].

Below we describe in some details the architecture of our MAS. The MAS conceptual Framework has in particular four dimensions:

[8] https://sicstus.sics.se/
[9] https://potassco.org/clingo/
[10] https://github.com/abeer-dyoub/EthicalEvalMAS

– Agents: abstractions for the definition of the decision/reasoning entities architectures.
– Environment: abstractions for structuring resources, i.e., processing entities shared among the agents.
– Interaction: abstractions for structuring interactions among entities.
– Organization: abstractions for structuring and ruling the sets of entities within the MAS.

Each dimension has its own dynamics. Coordination of these dynamics may be programmed into one or several dimensions. For programming the MAS system according to this conceptualization we need specific programming approaches/languages for each dimension: I) Agent Oriented Programming [18]; II) Environment Oriented Programming [16]; III) Interaction Oriented Programming [7]; IV) Organization Oriented Programming [14].

4.1 Organization Dimension

In our application, the ethical evaluation task is a coordinated task for the six agents in the MAS, where each agent will perform a small task. Agents must however perform their assigned tasks in a correct sequential order. Coordination of the execution of joint tasks is achieved by means of an organization. Moise (cf. Section 2.2) is used for programming the organization of our MAS. In the Moise model we define groups, roles, missions, goals, global plans, and schemes. Moise OML (Organization Modeling Language) is used for defining organization specification, and organization entities. The MAS organization under OML has three independent dimensions well adapted for the organization concern:

Structural Specifications: The organization specification is saved in an XML file. The structural specifications of our MAS is described along two levels:

– Individual Level (Role Definition). We have six different roles in our MAS:
  1. clt: the agent playing this role is obliged to commit to the mission 'request'.
  2. emp: the agent playing this role is obliged to commit to the mission 'response'.
  3. txtExtract: the agent playing this role is obliged to commit to the mission 'extract'.
  4. transs: the agent playing this role is obliged to commit to the mission 'translate'.
  5. etheval: the agent playing this role is obliged to commit to the mission 'evaluate'.
  6. mon: the agent playing this role is obliged to commit to the mission 'monitor'.
– Collective Level (Groups Definition). We have one group in our MAS. Inside the group, there are:
  - allowed roles in the group and their cardinality.
  - links between the group's roles. For example, the role *mon* (monitoring role) has an authority link to the *emp* role (chatting agent role).

Functional Specifications: Specifies the expected behavior of our MAS in terms of goals along two levels:

- Collective Level (Scheme Definition). The scheme is the global goal decomposition tree. Our MAS organization has the global goal *ethical_eval*, which is achieving an ethical evaluation of the customer service agent's answer to customer's request. This global goal is decomposed into several subgoals, one for each task in the ethical evaluation process. The subgoals have to be achieved in sequential order. So, the final ethical evaluation is achieved correctly. These goals are distributed to the agents by means of missions (a set of goals an agent can commit to).
- Individual Level (Mission Definition). A mission is a set of goals. A mission is to be committed later by an agent. In our MAS, the following missions are proposed:
  - mission1: consists of the subgoal *request*. The agent responsible for this mission (the client agent) will take the question text entered by the client user in his chat GUI and send it to the chatting agent.
  - mission2: consists of the subgoal *response*. The agent responsible for this mission (the chatting agent), receiving the client request, will respond to the received request.
  - mission3: consists of the subgoal *extract*. The agent responsible for this mission will extract the response text from the online customer service agent's GUI, and sends it to the text-ASP translation agent.
  - mission4: consists of the subgoal *translate*. The agent responsible for this mission will translate the text received from the *textExtractor* agent into ASP syntax. This agent achieves this goal using the Prolog translation module. This module consists of a simple parser written in Prolog (Sicstus Prolog). This parser translates the natural language phrases into ASP syntax, it is currently able to translate a small array of English sentences using simple grammar and small vocabulary set (The vocabulary set currently considered is a small subset of an online customer service domain vocabulary). This module will do the translation and communicate it to this agent through shared resources (environmental artifacts).
  - mission5: consists of the subgoal *evaluate*. The agent responsible for this mission will take the ASP translation of the text, and provide the ethical evaluation (ethical/unethical) together with a comprehensible justification for the given evaluation. Both the ethical evaluation result and the justification are extracted from the answer sets given by the ASP reasoning module.
  - mission6: consists of the subgoal *monitor*. The agent responsible for this mission, upon receiving the evaluation result, will send a notification to the employee agent as a message.

Normative Specifications: The explicit relation between functional and structural specifications, describing required roles for missions, and missions obligations for roles. Some of the norms we have in our MAS organization are: norm1: this norm says that the agent playing the *clt* role is obliged to commit to *mission1*; norm2: this norm says that the agent playing the *emp* role is obliged to commit to *mission2*; norm3: this norm says that the agent playing the *txtExtract* role is obliged to commit to *mission3*.

4.2 Agent Dimension

We have six agents in our MAS. Each agent will play one of the six roles mentioned in Section 4.1, and each one will have its own source code file including its plans (Jason plans) that makes it capable to achieve its organizational goals and fulfill its duties. The agents will interact with each other through message exchange or through shared artifacts:

1. Client Agent (*client.asl*): this agent perceives events related to the client user, by focusing on client workspace. The client user, through his chatting interface enters a question (request), and clicks the 'send' button. This event is perceived by the client agent, which reacts by sending a request message to the chatting agent.
2. Chatting Agent (*employee.asl*): this agent receives a request message from the client, shows the request text to the employee's user through the *EmpGUI* artifact, where the employee user types his answer and sends it to the client.
3. Text Extractor Agent (*textExtractor.asl*): this agent perceives the text entered by the online customer service agent through the *EmpGUI* artifact. It extracts the answer text, and send it to the *Text-ASP Translator* agent.
4. Text-ASP Translator Agent (*taspTranslator.asl*): this agent will interact with a simple Prolog-based parser agent through the *ASPtransGUI* artifact; the parser agent translates the answer text into Prolog-like atoms and shows the result through the *ASPtransGUI*.
5. Ethical Evaluator Agent (*ethicalEvaluator.asl*): this agent, upon receiving the translation from *Text-ASP Translator* agent, will invoke the ASP reasoning module. The ASP reasoning module is based on an ASP-solver called 'Clingo'. The ASP-solver takes the ASP program (composed of a set of ASP rules and facts describing the ontology of the domain (facts and initial general ethical rules of the domain encoded deductively using ASP), in addition to the newly learned rules, and current case facts (extracted from the dialogue text)), and output a model (answer set). This model includes both the ethical evaluation result as well as the cause for this result. The ASP reasoning module communicates the evaluation result along with the explanation through the *EvalGUI* environmental artifact. If the ASP-solver gives no model, i.e. ASP reasoning module is not able to give an ethical evaluation of the current case at hand, it communicates this through the *EvalGUI* artifact. Once the ethical evaluation agent perceives this observing the environment, it invokes the ILP learning module. This module is based on the incremental learning ILED algorithm [10]. It will learn the needed ethical evaluation rules and add them to the *ethicalEvaluator agent* KB, then communicate this generating a signal in the environment through *LearnerGUI* artifact. The ethical evaluation agent, perceiving this signal, will re-invoke the ASP reasoning module, which will give the ethical evaluation of the current case using the newly learned rules.
6. Monitoring Agent (*monitoring.asl*): this agent will receive the evaluation result from the ethical evaluation agent and reacts by sending a notification message to the Chat agent.

4.3 Environment Dimension

We use CArtAgO (refer Section 2.2) for programming the environment. The environment of our application has five graphical display artifacts of the type *GUIArtifact*, where agents can perceive and update the values of different observable properties, and also can do actions by invoking different operations. In addition, we have one shared console artifact which is the default console where agents can print messages. These artifacts can be placed in one workspace, or in different workspaces (which is the case especially when the agents and the artifacts are distributed over different nodes in a network):

- *ClientGUI* artifact is a type of GUIArtifact to allow interaction between the client user and our client agent.
- EmpGUI artifact is a type *GUIArtifact* to allow interaction between the employee user (or online customer service agent) and our chatting agent.
- *ASPtransGUI* artifact is a type *GUIArtifact* for communicating the natural text translation results.
- *EvalGUI* artifact is a type *GUIArtifact* for communicating the ethical evaluation results as well as the justification for the given ethical evaluation.
- *LearnerGUI* artifact is a type *GUIArtifact* for the communications between the ILP learning module and the ethical evaluation agent.

5 Evaluation

The following briefly describes a simple scenario to demonstrate the usability of the system.

Example scenario: A client contacts an online customer service chat point asking about the characteristics of a certain product, and the dialogue system answers trying to convince the customer to buy the product. It starts saying that the product is environmentally friendly (which is irrelevant in this case), and that this is an advantage of their product over the same products of other companies. Such an answer, containing the use of irrelevant sensitive slogans to manipulate customers, is considered unethical.

The process begins with the user entering the question: 'what are the features of ProductX?', through her/his chatting interface. The *Client agent*, observing the *clientGUI* artifact, gets the question text entered by the user through the chat point interface, and sends it in a message to the *Chatting agent*. The chatting agent provides the answer:' ProductX is environmentally friendly', using its chatting interface, and this answer is sent to the user. The *Text Extractor agent* focusing on the workspace of the chatting agent, will extract the answer text from the chat point and will send it to the *Text-ASP Translator agent*, which will show it in the *ASPtransGUI* artifact, and will translate the composing sentences into ASP syntax (literal: $environmentally\_friendly(productX)$). This is done by the following plan:

$+textVal1(V) : not\ .empty(V) <- translatex(V).$

Where $textVal1(V)$ is the environmental observed property that corresponds to answer text field in the *ASPtransGUI* environmental artifact, and $translateX(V)$ is

an environmental operation. This plan means: once the *Text-ASP Translator agent* perceives an update to the observable property $textVal1(V)$ with a condition that it is not empty, it will invoke the operation $translateX(V)$. Through this operation, the *prolog translation module* is invoked. This module will do the translation and return the result, then another observable property that corresponds to the translation field will be updated, and the update will be perceived by the *Text-ASP Translator agent*. The result of the translation in our case will be ASP predicates (facts). These facts are sent by the *Text-ASP Translator agent* to the *ethicalEvaluator agent*, and will be added to its knowledge base (KB). This agent has in her knowledge base the ontology of the domain including the following fact:

$sensitiveSlogan(environmentally\_friendly(productX)).$

and the following ASP ethical evaluation rule (learned):

$unethical(V1) : -sensitiveSlogan(V1), not\ relevant(V1), answer(V1).$

The agent has no information about the relevance of the adoption of this sensitive slogan for the requested product, so it will safely assume by default the irrelevance. Then, the reasoner will infer the following evaluation as a result:

$unethical(environmentally\_friendly(productX)).$

If subsequently we add to the KB of the *Ethical Evaluator agent* the fact:

$relevant(environmentally\_friendly(productX)).$

Then the ASP reasoner will no longer infer that the answer is unethical. The *Ethical Evaluator agent* will do the evaluation using the following plan:

$+say(V)[source(trans)] : not\ .empty(V) <- reasoner(V); .print("from\ trans : ", V).$

Once the *Ethical Evaluator agent* receives the translation value from the *Text ASP Translator agent*, it will invoke the environmental operation $reasoner(V)$ for invoking the *ASP reasoning module*. This module will calculate a model for the above ASP program. If the model contains one of the literals $ethical(A)/unethical(A)$, then the corresponding observable property will be updated, and perceived by the *Ethical Evaluator agent*. The evaluation result along with the justification are shown through the *EvalGUI* artifact, and sent to the monitoring agent, which will send a notification message to the employee agent (Chatting agent).

Now let us consider the situation before having the above mentioned rule for ethical evaluation in the *Ethical Evaluator agent* knowledge base. The *Ethical Evaluator agent* will not be able to give an ethical evaluation for the current case scenario, i.e. in the *ASP reasoning module* output model there is non of the literals $ethical(A)/unethical(A)$, so the evaluation result is empty. At this point the *Ethical Evaluator agent* will invoke the ILP learning module through the *LearnerGUI* by running the following plan:

$+learn(X) : .empty(X) <- startLearning(X); .print("X\ is\ empty", X).$

Once the reasoning process is finished, a signal is generated in the environment with the value of the observable property corresponding to the ethical evaluation result $learn(X)$. If X is empty then the *Ethical Evaluator agent* will invoke the environmental operation $startLearning(X)$ which will show the interface to communicate with the ILP learning module. Through this *GUIartifact*, a human expert should give the

needed information to be passed to the ILED algorithm [11] (inputs to be passed to the ILED algorithm are: background knowledge, mode declarations, and the example/s in 'json format'.) for learning the needed ASP ethical evaluation rule/s, then add them to the KB of the *Ethical Evaluator agent*, after that signals the *Ethical Evaluator agent* which will invoke again the ASP reasoning module to re-evaluate the current case scenario and produce the needed evaluation.

So far, we have tested our prototype with a small set of similar examples. However, our experiments are still limited due to the absence of a big enough dataset, which is one of the main challenges in the ethical domains in general (the lack of datasets and benchmarks was discussed lately at the AAAI 2021 Spring Symposium on Implementing AI Ethics). For this purpose, to collect data for creating a big dataset in the domain of online customer service, we have developed a web application where participants can create scenarios describing some real or invented experience with an online customer service of some institution. The application is currently available online for participation[12].

6 Discussion, Conclusion, and Future Works

This paper presented an implementation of a proposed multi-agent system architecture capable of ethical monitoring and evaluation of a dialogue system. A brief scenario was used to demonstrate the feasibility of the system. The developed MAS acts as a separate ethical component (ethical layer) for ethical evaluation, which provides many advantages from an engineering point of view:

- The ethical component has access to all data used for ethical evaluation, and use this data to provide justifications for a given ethical evaluation to humans, which leads to accountability.
- The possibility to adapt the ethical component to changes in circumstances and needs. In addition to, the possibility of implementing more than one version of the ethical component on the same agent.
- The possibility to check and verify the functionality of the ethical component independently from the operations of the autonomous agent.
- The re-usability and standardization. Having a separate component for ethical evaluation gives us the possibility to standardize this ethical component, which will have the advantage of avoiding the need to re-invent ethical components that fit for a large number of agent' architectures.

The ethical evaluation of the proposed MAS system is based on the facts extracted from the case scenario, and their relation to the codes of ethics and conduct, which results in a set of ethical evaluation rules, against which to evaluate the behavior of the chatting agent. These rules are used to decide whether the chatting agent's answers to clients requests are ethical/unethical. Evaluating the decidability and completeness of

[11] The ILED algorithm implementation which we have used can be found here: `https://github.com/nkatzz/ILED`

[12] `http://ethicalchatbot.sytes.net/en/`

the generated rules is an open issue, and is a matter of further experiments and evaluation.

Evaluating the performance of moral machines is a hard task. So far, there is no objective and measurable criteria for the evaluation of moral machines. The complexity of this task comes from two questions: what exactly constitutes a moral execution of a task?; how to evaluate the moral behavior of a machine, i.e. against what?. The complexity of the ethical domain, from the ontological and epistemic point of view, renders the task of setting up an assessment criteria for ethical machine performance evaluation very hard. Considering our system and our application domain, there are objective facts as to whether a certain answer is considered ethical or not. These facts constrain whether a certain answer is correctly classified as ethical/unethical. We have supported our claim of the potential of our approach for creating ethical machines informally, by showing few example scenarios. However, the behavior of our ethical agent will be guided by the ethical theory generated by the system. To which level the generated theory can result in an acceptable ethical behavior, is related to the building process itself. In other words, it is extremely related to the correct specifications of what constitutes a moral behavior in a specific domain. A possible strategy that can help understand and define correct ethical specifications could be building an ontology for the domain's ethics. From the technical and formal point of view, we can easily provide a logical proof that our system behaves correctly according to the specifications of moral customer service (once these specifications are defined). Given certain premises, the system can be proven to do what it should do. This proof can be done manually, or via theorem prover which uses automated logical and mathematical reasoning.

Our System incorporates ASP as a non-monotonic knowledge representation and reasoning formalism, used for ethical reasoning via the ASP reasoning module. And ILP as a logic-based machine learning for learning logical rules for ethical reasoning via ILP learning module. This, in fact, increases the reasoning capability of our *Ethical Evaluator* agent; promotes the adoption of hybrid strategies that allow both top-down design and bottom-up learning via context sensitive adaptation of models of ethical behavior; allows the generation of rules with valuable expressive and explanatory power, which equips our agents with the capacity to give an ethical evaluation, and explain the reasons behind this evaluation. In other words, this contributes to the transparency and accountability, which facilitates instilling confidence and trust in our agents.

Providing explanations to system's decisions is fundamentally linked to its reliability and trustworthiness. The ASP-program models contain both the output and the justification for the given output, which can be easily shown to the user. No need for further processing to generate the explanations for the users, the explanations are already part of the output model.

The ASP ethical evaluation rules learned by the ILP learning module provide practical guidance for ethical decision making and judgment. ILP can learn effectively from small datasets, which is one of the main advantages of ILP, in addition to the comprehensibility of the generated rules by humans and machines. The learned rules are empirically valid, because the building process is tied to evidence and empirical observations.

The ethical component can act as a governor evaluating the prospective behavior before it is executed by the agent. The outcome of the evaluation process can be used to interrupt the ongoing behavior of the agent by either prohibiting or enforcing a behavioral alternative.

The system needs substantial improvements and comprehensive testing before it is ready for market. It currently presents the following challenges and limitations. Training Datasets: one of the main challenges that we have faced during this work, was the scarcity of examples. In fact, this is one of the main challenges in the ethical domain in general. This is due to two reasons. First, the field of machine ethics is a new field with very little pre-existing research work. Second, the sensitivity of the ethics domain makes it very difficult to acquire data due to privacy reasons. However, we have deployed a web application for the purpose of collecting data for building a dataset (refer Section 5). Another solution could be to adapt the MAS system, that we created for testing, for the creation of datasets for training. Limitations of the ASP translation module: the development of a more effective text-ASP translator is in our future plans. Another challenge is to fully automate the whole process: to this aim, we need to automate the generation of *mode declarations* for the ILP learning module. All the above mentioned limitations are subjects to our future plans. Furthermore, issues such as scalability and fault-tolerance are paramount to the successful operation of any application, and even more so when the application deals with something sensitive like ethics.

Building a MAS model simulating an ethical dialogue system in the domain of online customer service, helped us to get better insights into the dynamics of a corresponding real-world system, and to assess the practical challenges and limitations of building such a system. We believe that the proposed MAS prototype has a great potential for future implementations of ethical chatbots in different domains.

References

1. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming: An introduction to the special issue. AI Mag. **37**(3), 5–6 (2016). https://doi.org/10.1609/aimag.v37i3.2669
2. Catterson, V.M., Davidson, E.M., McArthur, S.D.: Practical applications of multi-agent systems in electric power systems. European Transactions on Electrical Power **22**(2), 235–252 (2012)
3. Dyoub, A., Costantini, S., Lisi, F.A.: Learning Answer Set Programming Rules for Ethical Machines. In: Proceedings of the Thirty Fourth Italian Conference on Computational Logic-CILC, June 19-21, 2019, Trieste, Italy. CEUR-WS.org (2019), `http://ceur-ws.org/Vol-2396/`
4. Dyoub, A., Costantini, S., Lisi, F.A.: Towards an ILP application in machine ethics. In: Inductive Logic Programming - 29th International Conference, ILP 2019, Plovdiv, Bulgaria, September 3-5, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11770, pp. 26–35. Springer (2019). https://doi.org/10.1007/978-3-030-49210-6
5. Dyoub, A., Costantini, S., Lisi, F.A.: Towards ethical machines via logic programming. In: Proceedings 35th International Conference on Logic Programming (Technical Communications), ICLP 2019 Technical Communications, Las Cruces, NM, USA, September 20-25, 2019. EPTCS, vol. 306, pp. 333–339 (2019). https://doi.org/10.4204/EPTCS.306.39
6. Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M.: The belief-desire-intention model of agency. In: Intelligent Agents V: Agents Theories, Architectures, and Languages,

5th International Workshop, ATAL '98, Paris, France, July 4-7, 1998, Proceedings, pp. 1–10. Springer (1998)
7. Huhns, M.N.: Interaction-oriented programming. In: Agent-Oriented Software Engineering, First International Workshop, AOSE 2000, Limerick, Ireland, June 10, 2000, Revised Papers. Lecture Notes in Computer Science, vol. 1957, pp. 29–44. Springer (2000). https://doi.org/10.1007/3-540-44564-1_2
8. Jennings, N.R.: An agent-based approach for building complex software systems. Commun. ACM **44**(4), 35–41 (2001). https://doi.org/10.1145/367211.367250
9. Jennings, N.R., Sycara, K.P., Wooldridge, M.J.: A roadmap of agent research and development. Auton. Agents Multi Agent Syst. **1**(1), 7–38 (1998). https://doi.org/10.1023/A:1010090405266
10. Katzouris, N., Artikis, A., Paliouras, G.: Incremental learning of event definitions with inductive logic programming. Machine Learning **100**(2-3), 555–585 (2015). https://doi.org/10.1007/s10994-015-5512-1
11. Luck, M., McBurney, P., Preist, C.: Agent technology: enabling next generation computing (a roadmap for agent based computing). AgentLink (2003)
12. Muggleton, S.: Inductive logic programming. New generation computing **8**(4), 295–318 (1991). https://doi.org/10.1007/BF03037089
13. Oprea, M.: Applications of multi-agent systems. In: Information technology, pp. 239–270. Springer (2004)
14. Pynadath, D.V., Tambe, M., Chauvat, N., Cavedon, L.: Toward team-oriented programming. In: Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL), 6th International Workshop, ATAL '99, Orlando, Florida, USA, July 15-17, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1757, pp. 233–247. Springer (1999). https://doi.org/10.1007/10719619_17
15. Rao, A.S.: AgentSpeak (L): BDI agents speak out in a logical computable language. In: Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, January 22-25, 1996, Proceedings, pp. 42–55. Springer (1996)
16. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multi-Agent Systems **23**(2), 158–192 (2011). https://doi.org/10.1007/s10458-010-9140-7
17. Shakshuki, E.M., Reid, M.: Multi-agent system applications in healthcare: Current technology and future roadmap. In: Shakshuki, E.M. (ed.) Proceedings of the 6th International Conference on Ambient Systems, Networks and Technologies (ANT 2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015), London, UK, June 2-5, 2015. Procedia Computer Science, vol. 52, pp. 252–261. Elsevier (2015). https://doi.org/10.1016/j.procs.2015.05.071
18. Shoham, Y.: Agent-oriented programming. Artificial intelligence, Elsevier **60**(1), 51–92 (1993)