Penalization Framework For Autonomous Agents Using Answer Set Programming

Vineel S. K. Tummala
Computer Science and Software Engineering
Miami University
Oxford, USA
tummalvs@miamioh.edu

This paper presents a framework for enforcing penalties on intelligent agents that do not comply with authorization or obligation policies in a changing environment. A framework is proposed to represent and reason about penalties in plans, and an algorithm is proposed to penalize an agent's actions based on their level of compliance with respect to authorization and obligation policies. Being aware of penalties an agent can choose a plan with a minimal total penalty, unless there is an emergency goal like saving a human's life. The paper concludes that this framework can reprimand insubordinate agents.

1 Introduction and Motivation

A framework will be presented that can enforce penalties on intelligent agents in a changing environment if autonomous agents don't obey authorization or obligation policies. Policies for an agent A acting in a changing domain/environment T are encoded as a set of conditions that describes whether an agent's actions are permitted or not, and whether the agent is obligated to perform an action or not perform it. In some cases, an autonomous agent can choose to perform an unauthorized action. In this case, the agent is forced to pay a penalty, where it can be reprimanded for its actions or lose its job for insubordination [1]. To make this possible, an architecture will be created to represent and reason about penalties in plans.

As autonomous agents continue to develop, it is necessary that proper policy representation and comprehension exist. It is essential to monitor the issues within the environment to ensure that agents act appropriately. Autonomous agents are intended to perform certain actions without any instructions or interference from a controller (pilot or driver) of the agent. The activities of an autonomous agent are often modeled after human behavior so that the agent can perform tasks that humans would. In order to do this, human behavior needs to be accurately represented. This can be successfully achieved by a logical approach, using declarative programming languages. Declarative programming languages are able to define human behavior correctly and in an understandable way. One such declarative programming language is Answer Set Programming [2],

2 Background and overview of the existing literature

In this section, background work related to the penalization framework for autonomous agents will be described. I assume that the reader is familiar with Answer Set Programming (ASP) [8, 7], a declarative programming language with roots in Logic Programming and Nonmonotonic reasoning [4], and the ASP solver CLINGO[1]. Full details about CLINGO can be found in the papers by Calimeri et al. [3] and Gebser

[1] https://potassco.org/Clingo/

S. Costantini, E. Pontelli, A. Russo, F. Toni,
R. Calegari, A. D'Avila Garcez, C. Dodaro, F. Fabiano,
S. Gaggl, A. Mileo, (Eds.): ICLP 2023
EPTCS 385, 2023, pp. 411–415, doi:10.4204/EPTCS.385.50

et al. [5]. I also assume familiarity of the reader with autonomous intelligent agents.

2.1 Transition Systems

In order for intelligent agents to be functional and autonomous, they require a proper description of the changing environment in which they act. Transition systems are a tool that is used to describe these domains. Transition systems are fundamentally directed graphs that represent a changing domain, in which the nodes represent the state of the domain, and actions that might occur in the domain are represented by arcs [1]. As a result, the domain is altered from one state to another. Transition diagrams can become huge if many fluents are being tracked, thus they are not used in practice. They can be represented in a more concise way by an action language [2] which is a high-level declarative language.

2.2 Policies

The proposed research will be focusing on agents which are aware of policies. Policies are sets of rules to follow in a given domain or environment and are also described using declarative programming languages. Given a policy, P, with respect to a given environment T, all actions performed in the said environment can be judged based on whether or not they follow the rules defined by the policy. In the case of an intelligent agent, it is important for it to be aware of all the policies in advance in order to comply with all the necessities when coming up with plans for action. The following section will define compliance to policy, look into both authorization and obligation policies.

2.2.1 Authorization

Authorization policies are policies in which an agent is either permitted (authorized) or not permitted (not authorized) to perform an action. Gelfond and Lobo [9] define Authorization Policy Language (*APL*) to describe authorization policies in a dynamic system $\Sigma = \langle A, T \rangle$, with an agent A and a transition diagram/environment T. This language is implemented through Answer Set Programming (ASP). The signature of *APL* includes a predicate for authorization policies *permitted(e)* where e is an elementary action of A. [9].

2.2.2 Obligation

An obligation policy is one in which an agent is obligated to carry out an action or not to carry it out. *AOPL* is a policy language to describe systems with an agent A and a transition diagram T over signature Σ, similar to *APL* [9]. Known as *Authorization and Obligation Policy Language*, Gelfond and Lobo [9] designed this language, to extend *APL* with obligation policies by adding a new symbol *obl(h)* where h is an elementary action of A or negation of any such action [9]. This language is implemented through ASP.

2.3 Compliance

Compliance denotes whether an action or set of actions should be taken or not in accordance with a given policy P. A set of actions is defined as strongly compliant, weakly compliant, or non-compliant. Strongly compliant means that no rules were broken and it allows/supports their execution. If there is

[2] See `https://www.diva-portal.org/smash/get/diva2:1716299/FULLTEXT01.pdf`

no sufficient information then it is called weakly compliant, and if the actions have broken one or more policies then it is called as non-compliant.

2.3.1 Authorization Compliance

Compliance in relation to authorization policies, as defined by *APL(Σ)*, can be divided into strongly compliant, weakly compliant, or not compliant [9].

2.3.2 Obligation Compliance

For obligation policies defined through *AOPL*(Σ), events are classified into *compliant* or *non-compliant* [9]. Unlike authorization policies, there are no weakly compliant events with respect to obligations.

2.4 Planning

A plan is a sequence of actions taken by an agent in order to achieve a given goal [6]. *Answer Set Planning* [10] refers to the use of Answer Set Programming to solve different types of planning problems by adding a planning module to an ASP description of a changing environment. Answer sets of the resulting program correspond to possible plans.

3 Goal of the research

The goal of the proposed research is to create a penalization framework for autonomous agents using ASP where autonomous agents are penalized if they do any job/task that they are not intended to do according to the policy.

3.1 Constructing the Framework

In this section, we present our ideas for constructing a penalization framework for autonomous agents.

3.1.1 Encoding the penalties

In order for these penalties to be imposed on the agents we will create a statement to penalize the agents. we will choose a specific scale to impose penalty points for agents; as of now the scale is 1-3, in the actual framework we will explore different possibilities.

We keep track of the values as low, medium, and high to help the agent calculate the value of the penalty while choosing the plans.

3.1.2 Calculating the total penalty

We calculate the total penalty, in the end, using the aggregate sum in CLINGO[3] (solver for ASP). Aggregates are functions that combine the values of a set of terms to produce a single value. Using this function we add up all the penalties given to an agent in an environment and we can evaluate an agent based on these penalties.

[3] `https://potassco.org/Clingo/run/`

4 Current status of the research

Collected all background and related work, and created two domains to test the penalization framework.

4.1 Collecting policies and scenarios

When we speak about penalizing autonomous agents, we need to create domains and scenarios where we evaluate autonomous agents for their actions and penalize them. Also, to achieve this we need to specify authorization and obligation policies for the domain beforehand. Conflicts may arise when Authorization and obligation policies interact or come across each other. Therefore we need to specify authorization and obligation policies precisely. First definitions and rules of both the authorization and obligation policies developed by Gelfond and Lobo [9] must be understood. After fully understanding the rules that exist, different scenarios can be tested. Scenarios will be tested to see what an agent selects if it has two non-compliant plans, and what plan it chooses. Some of the scenarios are:

4.1.1 Domain 1: Drone Delivery

Goal: Its goal is to deliver a package to a customer's doorstep, located in a suburban area with many houses.

Policy Rules: Consider an autonomous agent in this scenario as a delivery drone with the following rules to ensure safe and efficient deliveries:

- Always fly at a safe height to avoid obstacles and people on the ground.
- only land on designated landing pads or open spaces to avoid collisions and ensure safe deliveries.

4.1.2 Domain 2: Self-Driving Car

The autonomous agent in this scenario is a self-driving car, programmed with a set of rules to ensure safe and efficient transportation.

Goal: Its goal is to transport a passenger from their current location to a specified destination, which involves driving through a busy city street with heavy traffic.

Policy Rules: Consider an autonomous agent in this scenario as a self-driving car with the following rules to ensure safe and efficient deliveries:

- Always obey traffic laws and signals, including speed limits, stop signs, and traffic lights.
- Prioritize the safety of passengers and other drivers on the road.

5 Open issues and expected achievements

Open Issues

- We can't create a penalty for each and every possible action, so should we set a default value if the action is not encoded?
- Setting a maximum penalty limit after which the agent would be automatically disabled from its operations for not following the compliance. This would ensure the agents that they do not perform any operations that are not compliant.

The expected achievement is to create a framework where autonomous agent chooses plans with fewer penalties and also according to the strength of the goal.

References

[1] Justin Blount (2013): *An architecture for intentional agents*. Ph.D. thesis, doi:10.4204/EPTCS.345.23.

[2] Gerhard Brewka, Thomas Eiter & Miroslaw Truszczynski (2011): *Answer set programming at a glance*. *Commun. ACM* 54(12), pp. 92–103, doi:10.1145/2043174.2043195.

[3] Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Marco Maratea, Francesco Ricca & Torsten Schaub (2020): *ASP-Core-2 input language format*. *Theory and Practice of Logic Programming* 20(2), pp. 294–309, doi:10.1017/S1471068419000450.

[4] Thomas Eiter, Giovambattista Ianni & Thomas Krennwallner (2009): *Answer Set Programming: A Primer*. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset & Renate A. Schmidt, editors: *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures*, *Lecture Notes in Computer Science* 5689, Springer, pp. 40–110, doi:10.1007/978-3-642-03754-2_2.

[5] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub & Marius Schneider (2011): *Potassco: The Potsdam Answer Set Solving Collection*. *AI Commun.* 24(2), pp. 107–124, doi:10.3233/AIC-2011-0491.

[6] Michael Gelfond & Yulia Kahl (2014): *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, doi:10.1017/CBO9781139342124.

[7] Michael Gelfond & Vladimir Lifschitz (1988): *The Stable Model Semantics for Logic Programming*. In Robert A. Kowalski & Kenneth A. Bowen, editors: *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes)*, MIT Press, pp. 1070–1080, doi:10.2307/2275201.

[8] Michael Gelfond & Vladimir Lifschitz (1990): *Logic Programs with Classical Negation*. In David H. D. Warren & Péter Szeredi, editors: *Logic Programming, Proceedings of the Seventh International Conference, Jerusalem, Israel, June 18-20, 1990*, MIT Press, pp. 579–597, doi:10.1007/BF03037169.

[9] Michael Gelfond & Jorge Lobo (2008): *Authorization and Obligation Policies in Dynamic Systems*. In Maria Garcia de la Banda & Enrico Pontelli, editors: *Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings*, *Lecture Notes in Computer Science* 5366, Springer, pp. 22–36, doi:10.1007/978-3-540-89982-2_7.

[10] Vladimir Lifschitz (1999): *Answer Set Planning (Abstract)*. In Michael Gelfond, Nicola Leone & Gerald Pfeifer, editors: *Logic Programming and Nonmonotonic Reasoning, 5th International Conference, LPNMR'99, El Paso, Texas, USA, December 2-4, 1999, Proceedings*, *Lecture Notes in Computer Science* 1730, Springer, pp. 373–374, doi:10.1007/3-540-46767-X_28.